

Peter Buxmann
Heiner Diefenbach
Thomas Hess

Die Softwareindustrie

Ökonomische Prinzipien,
Strategien, Perspektiven

3. Auflage

Die Softwareindustrie

Peter Buxmann • Heiner Diefenbach
Thomas Hess

Die Softwareindustrie

Ökonomische Prinzipien, Strategien,
Perspektiven

3., vollständig überarbeitete und erweiterte Auflage



Springer Gabler

Peter Buxmann
Wirtschaftsinformatik | Software Business &
Information Management
Technische Universität Darmstadt
Darmstadt
Deutschland

Heiner Diefenbach
TDS AG
Neckarsulm
Deutschland

Thomas Hess
Wirtschaftsinformatik und Neue Medien
LMU München
München
Deutschland

ISBN 978-3-662-45588-3
DOI 10.1007/978-3-662-45589-0

ISBN 978-3-662-45589-0 (eBook)

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 Berlin Heidelberg 2008, 2011, 2015
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 dürfen. 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.

Gedruckt auf säurefreiem und chlorfrei gebleichtem Papier

Springer Fachmedien Wiesbaden GmbH ist Teil der Fachverlagsgruppe Springer Science+Business Media (www.springer.com)

Vorwort zur ersten Auflage

Kaum eine andere Branche hat die Gesellschaft und die Unternehmenswelt so nachhaltig verändert wie die Softwareindustrie. So ist die Softwareunterstützung von inner- sowie zwischenbetrieblichen Geschäftsprozessen heute genauso eine Selbstverständlichkeit wie das „Googeln“ nach Informationen oder die Nutzung von Navigationsgeräten. Es ist daher auch wenig überraschend, dass die Softwareindustrie gemäß IDC bei einem weltweiten jährlichen Umsatz von 230 Mrd. US-Dollar (ohne komplementäre Dienstleistungen) und mit einem Zuwachs von jährlich sieben Prozent in Europa sowie bis zu neun Prozent in Asien zu den bedeutendsten Wachstumsmärkten zählt. Dabei gilt, dass die Softwarebranche so international wie kaum eine andere ist. Dementsprechend konkurrieren Softwareunternehmen weltweit um Kunden und zunehmend auch um Mitarbeiter. Aber nicht nur aufgrund der internationalen Softwaremärkte, sondern auch wegen der besonderen Eigenschaften des Gutes Software gelten für Softwareanbieter eigene Spielregeln. Hiervon handelt dieses Buch.

Es wäre aufgrund der Besonderheiten und auch der Attraktivität der Branche zu erwarten, dass es – ähnlich wie etwa für den Industrie- und Bankensektor, für Versicherungen oder für die Medien- und Energiewirtschaft – eine spezielle Betriebswirtschaftslehre oder Lehrstühle für die Softwareindustrie aus der Wirtschaftsinformatik heraus gibt. Dies ist bisher jedoch überraschenderweise nicht der Fall. Diese Lücke war für uns ein Anlass, die Software Economics Group Darmstadt-München (www.software-economics.org) zu gründen sowie das vorliegende Buch zu schreiben.

Im ersten Kapitel werden wir nach einer kurzen Beschreibung der Anfänge der Softwareindustrie auf die verschiedenen Player in dieser Branche eingehen. Einen Schwerpunkt bildet die Untersuchung der unterschiedlichen Erlösquellen, also das Lizenz- und Servicegeschäft. Die ökonomischen Prinzipien der Softwareindustrie werden im zweiten Kapitel thematisiert. Hierzu gehören die spezifischen Eigenschaften des Gutes Software sowie die besonderen Merkmale von Softwaremärkten. Darauf aufbauend werden im dritten Kapitel Strategieempfehlungen für die Softwareindustrie abgeleitet. Dabei betrachten wir Kooperations-, Preis-, Vertriebs- und Industrialisierungsstrategien. In diesem Zusammenhang werden wir auch neue Technologien, wie das SOA-Konzept und das Model Driven Engineering, untersuchen. Dabei liegt der Schwerpunkt allerdings eher auf der Analyse der Auswirkungen auf die Softwareindustrie als auf der Darstellung technischer

Implementierungsdetails. Aufgrund des hohen Internationalisierungsgrads der Branche gehen wir im vierten Kapitel auf die Globalisierung der Softwareentwicklung ein. Dabei werden wir uns mit Offshore- sowie Open-Source-Projekten und ihren Auswirkungen auf die Softwareindustrie beschäftigen.

Bei der Erstellung des Buches konnten wir auf die Arbeiten der bereits oben ange- sprochenen Software Economics Group Darmstadt-München zurückgreifen. Im Rahmen dieses Projekts ist auch das empirische Material entstanden, das eine wichtige Grundlage verschiedener Abschnitte ist. So haben wir eine Vielzahl von Expertengesprächen geführt. Wir möchten uns sehr herzlich bei den folgenden Personen aus der Softwareindustrie dafür bedanken, dass sie ihr Wissen und ihre Erfahrungen mit uns geteilt bzw. uns auf andere Weise unterstützt haben: Dieter Berz, Rüdiger Bolt, Christian Boos, Debjit Chaudhuri, Akhil Eswaran, Alexander Gassmann, Martin Haas, Jürgen Henn, Martin Helling, Lutz Heuser, Franz Hollich, Kim Langer, Horst Kinzinger, Thomas Lünendonk, Knut Manske, Thomas Meyer, Karsten Öhler, Jens Pfennig, Jürgen Powik, Nikolai Puntikov, Ganesh Ramaoorthy, Andreas Reinicke, Stefan Ried, Morten Rolfs, Thomas Roth, Karl-Heinz Streibich, Christopher Sürie, Ferenc Szilágyi, Justin Vaughan-Brown, Joachim Voegele und Martin Winkler.

Unseren Mitarbeiterinnen und Mitarbeitern Björn Brandt, Julia Gebele, Markus Hahn, Patrick Johnscher, Sven Schade, Jochen Strube, Thomas Widjaja und Christian Wolf danken wir für ihre fachliche Unterstützung bei der Erstellung dieses Buches.

Bei Anette von Ahsen, Melanie Hüttenberger und Gerrit Pohl bedanken wir uns sehr herzlich für die vielen und umfangreichen „Bug Reports“. Ebenso dankbar sind wir Niels Thomas vom Springerverlag für die sehr angenehme Zusammenarbeit und die vielen guten Verbesserungsvorschläge.

Wir freuen uns auf Ihre Kommentare und Anmerkungen und wünschen Ihnen viele Anregungen und auch viel Spaß beim Lesen des Buchs.

im September 2007

Peter Buxmann
Heiner Diefenbach
Thomas Hess

Vorwort zur dritten Auflage

Die Softwareindustrie ist eine sich ständig verändernde Branche. Mit dieser dritten Auflage wollen wir das bewährte Konzept der beiden vorigen Auflagen fortführen und ein aktuelles Bild der Softwareindustrie zeichnen. Seit der zweiten Auflage dieses Buches in 2011 haben brancheninterne Veränderungen und der wissenschaftliche Dialog neue Erkenntnisse hervorgebracht. Um diese Entwicklungen zu berücksichtigen, haben wir insbesondere die Teile zu den Themenbereichen Cloud Computing und Geschäftsmodelle in der Softwareindustrie deutlich ausgeweitet. Darüber hinaus wurden neue Teilabschnitte verfasst, u. a. zur Abhängigkeit in IT-Outsourcing-Beziehungen, zum Fachkräftemangel in der Software- und IT-Industrie, zur Bewertung des M&A-Erfolges in der Softwareindustrie sowie zur Erschließung internationaler Märkte.

Für die fachliche Unterstützung zur Erstellung dieser dritten Auflage bedanken wir uns sehr herzlich bei unseren Mitarbeiterinnen und Mitarbeitern der Software Economics Group Darmstadt-München (www.software-economics.org). So sind viele Neuerungen durch Inhaltsbeiträge von Dr. Tobias Ackermann, Dr. Stefan Harnisch, Dr. Jasmin Kaiser, Natalie Kaltenecker, André Loske, Anton Pussep und Dr. Markus Schief entstanden.

Auch bedanken wir uns erneut sehr herzlich bei Anette von Ahsen für ihre hervorragende Unterstützung durch umfangreiche „Bug Reports“ und die dazu passenden „Patches“. Ebenso dankbar sind wir Miriam Bär und Rabea Sonnenschein für ihre außerordentliche Unterstützung bei der Manuskripterstellung. Herrn Dr. Niels Peter Thomas vom Springer-Verlag möchten wir ebenfalls danken für die erneut sehr angenehme Zusammenarbeit.

Wir freuen uns auf Ihre Kommentare und Anmerkungen und wünschen Ihnen viele Anregungen und auch viel Spaß beim Lesen dieses Buchs.

im September 2014

Peter Buxmann
Heiner Diefenbach
Thomas Hess

Inhaltsverzeichnis

Teil I Allgemeine Grundlagen

1 Die Spielregeln in der Softwareindustrie	3
1.1 Software und Softwaremärkte: Ausgewählte Besonderheiten der Softwareindustrie im Überblick	3
1.2 Die Anfänge der Softwareindustrie.	4
1.3 Typen von Softwareanbietern und Auswahlentscheidungen der Anwender	5
1.3.1 Softwareanbieter im engeren und weiteren Sinne	5
1.3.2 Die Auswahl von Software	10
1.4 Dienstleistungserlöse in der Softwareindustrie.	16
Literatur	17
2 Ökonomische Prinzipien der Softwareindustrie	19
2.1 Eigenschaften digitaler Güter	19
2.2 Netzeffekte auf Softwaremärkten: The Winner Takes it All.	20
2.2.1 Netzeffekte: Grundlagen und Definitionen	21
2.2.2 Auswirkungen von Netzeffekten auf Softwaremärkte	24
2.2.3 Struktur von Softwaremärkten	27
2.2.4 Netzeffekte als Wettbewerbsfaktor	29
2.2.5 Ein Anwendungsbeispiel: Zweiseitige Netzeffekte und Plattformstrategien in der digitalen Spieleindustrie	30
2.2.6 Grenzen der Netzeffekttheorie	34
2.3 Das Standardisierungsproblem	35
2.3.1 Ansatz und Hintergründe	35
2.3.2 Das zentrale Standardisierungsproblem als Optimierungsproblem	38
2.3.3 Das dezentrale Standardisierungsproblem – Eine spieltheoretische Darstellung	40
2.3.4 Das Standardisierungsproblem – Lessons learned	42
2.4 Transaktionskostentheorie: Auf der Suche nach den Grenzen eines Softwareunternehmens	44
2.4.1 Ansatzpunkt und Elemente der Transaktionskostentheorie	45

2.4.2	Arbeitsteilung zwischen Unternehmen aus Sicht der Transaktionskostentheorie	47
2.4.3	Strukturelle Veränderungen der Transaktionskosten: The Move to the Middle	48
2.4.4	Ausblick: Intermediäre und Transaktionskosten	50
2.5	Softwareentwicklung als Agency-Problem: Anreizkompatible Entlohnung und effiziente Kontrolle	51
2.5.1	Principal-Agent-Beziehungen: Definitionen und Grundlagen	51
2.5.2	Anreizkompatible Vergütungsschemata	53
2.5.3	Kontrollsysteme	56
	Literatur	58
3	Strategien für Softwareanbieter	61
3.1	Kooperations- und Übernahmestrategien	61
3.1.1	Kooperationen in der Softwareindustrie	62
3.1.2	Mergers & Acquisitions in der Softwareindustrie	72
3.2	Vertriebsstrategien	85
3.2.1	Gestaltung des Vertriebssystems: Organisation und Vertriebswege in der Softwareindustrie	85
3.2.2	Gestaltung der Beziehungen zu Vertriebspartnern und Key Accounts	90
3.2.3	Kennzahlensysteme als Instrument für das Vertriebscontrolling in der Softwareindustrie	92
3.2.4	Gestaltung der Verkaufsaktivitäten	96
3.2.5	Erschließung internationaler Märkte	102
3.3	Preisstrategien	107
3.3.1	Grundüberlegungen	107
3.3.2	Parameter der Preisgestaltung für Softwareprodukte	108
3.3.3	Ansätze zur Preissetzung für Individualsoftwareanbieter	122
3.4	Entwicklungsstrategien	125
3.4.1	Strukturierung der Softwareentwicklung	125
3.4.2	Personalführung in der Softwareentwicklung	129
3.4.3	Add-on: Fachkräftemangel in der Software- und IT-Industrie	132
	Literatur	135
4	Geschäftsmodelle in der Softwareindustrie	141
4.1	Die Wertschöpfungskette in der Softwareindustrie	142
4.1.1	Kernaktivitäten der Software-Wertschöpfungskette	142
4.1.2	Wertschöpfungsketten in der Softwareindustrie – 3 Fallstudien	144
4.2	Geschäftsmodelle in der Softwareindustrie – ein Framework	152
4.3	Software Business Model Tool	158
	Literatur	163

Teil II Spezielle Themen

5 Outsourcing und Offshoring der Softwareentwicklung	169
5.1 Überblick	169
5.2 Formen des Outsourcings und Offshorings	170
5.3 Motive für Outsourcing und Offshoring.....	174
5.4 Abhängigkeit in IT-Outsourcing-Beziehungen.....	178
5.4.1 Grundlagen der Analyse	178
5.4.2 Relevanz	181
5.4.3 Substituierbarkeit	182
5.4.4 Spillover	184
5.5 Standortwahl von Softwareanbietern	185
5.6 Outsourcing durch Softwareanwender	188
5.6.1 Outsourcing der Neuentwicklung von Individualsoftware.....	188
5.6.2 Outsourcing der Anpassung von Standardsoftware	191
5.6.3 Outsourcing der Weiterentwicklung und Wartung von Anwendungssoftware	194
5.6.4 Zufriedenheit der Anwender mit Onshore-, Nearshore- und Farshoreanbietern	198
5.7 Nearshoring versus Farshoring: Die Entfernung zum Kunden als Erfolgsfaktor?	199
5.7.1 Sprachliche und kulturelle Barrieren in Offshore-Projekten	199
5.7.2 Die Bedeutung persönlicher Treffen für den Projekterfolg	202
5.7.3 Herausforderungen und Chancen der Zeitverschiebung.....	203
Literatur	205
6 Plattformkonzepte	207
6.1 Überblick	207
6.2 Produktplattformen in der Softwareindustrie	207
6.2.1 Kostenstruktur plattformbasierter Softwareentwicklung	207
6.2.2 Add-on: Industrialisierung als Managementkonzept für die Softwareindustrie	210
6.3 Branchenplattformen in der Softwareindustrie.....	213
6.3.1 Offenheit einer Branchenplattform.....	213
6.3.2 Das Management der Komplementäre	216
Literatur	219
7 Cloud Computing	221
7.1 Überblick	221
7.2 Der Cloud-Markt.....	223
7.3 Die Sicherheit in der Cloud – Empirische Ergebnisse aus Anwender- und Anbieterperspektive.....	225

7.3.1	Anwendersicht	226
7.3.2	Anbietersicht	229
7.4	Software as a Service: die Anwendungsebene des Cloud Computing	231
7.4.1	SaaS aus Anwendersicht – Chancen und Risiken	233
7.4.2	SaaS aus Anbietersicht – Chancen und Risiken	240
7.4.3	Empirische Untersuchung der Preisstrategien und Geschäftsmodelle für SaaS-Anbieter	247
7.4.4	Fallstudie zum Vergleich nutzungsabhängiger und nutzungsunabhängiger Preismodelle	252
Literatur		254
8	Open Source Software	257
8.1	Überblick	257
8.2	Charakteristika von Open Source Software	258
8.3	Open-Source-Projekte: Prinzipien und Motivation der Softwareentwickler	262
8.3.1	Ablauf und Organisation von Open-Source-Projekten	262
8.3.2	Zur Motivation der Beitragenden	264
8.4	Open Source Software aus Sicht des Anwenders	266
8.5	Engagement kommerzieller Softwareanbieter	267
8.6	Quelloffene ERP-Systeme	269
Literatur		277
Weiterführende Literatur		279
Sachverzeichnis		291

Abkürzungsverzeichnis

AGB	Allgemeine Geschäftsbedingungen
ASD	Adaptive Software Development
ASP	Application Service Providing
BOT	Build Operate Transfer
BSA	Business Software Alliance
BSD	Berkeley Software Distribution
CAR	Cumulative Abnormal Return
CAAR	Cumulative Average Abnormal Return
CCC	Content Communication and Collaboration
CEO	Chief Executive Officer
CIO	Chief Information Officer
CMMI	Capability Maturity Model Integration
CRM	Customer Relationship Management
CTO	Chief Technology Officer
CVS	Concurrent Versions System
DACH	Deutschland, Österreich, Schweiz
EDI	Electronic Document Interchange
EDV	Elektronische Datenverarbeitung
ERP	Enterprise Resource Planning
GE	Geldeinheiten
GNU	GNU is not Unix
GPL	General Public License
HTTP	Hypertext Transfer Protocol
IE	Internet Explorer
IP	Intellectual Property
ISO	International Organization for Standardization
KMU	Kleine und mittelständische Unternehmen
KPI	Key Performance Indicator
LGPL	Library/Lesser General Public License
LoC	Lines of Code
M&A	Mergers & Acquisitions

MIT	Massachusetts Institute of Technology
MPL	Mozilla Public License
OEM	Original Equipment Manufacturer
OSI	Open Source Initiative
OSS	Open Source Software
PDF	Portable Document Format
PMC	Point of Marginal Cheapness
PME	Point of Marginal Expensiveness
PPS	Produktionsplanungs- und -steuerungssystem
PSM	Price Sensitivity Meter
ROI	Return on Investments
SaaS	Software as a Service
SIIA	Software & Information Industry Association
SCM	Supply Chain Management
SOA	Serviceorientierte Architektur
SQL	Structured Query Language
SW	Software
VB	Vertriebsbeauftragte(r)
VHS	Video Home System
XP	eXtreme Programming
XML	Extensible Markup Language
ZB	Zahlungsbereitschaft

Teil I

Allgemeine Grundlagen

1.1 Software und Softwaremärkte: Ausgewählte Besonderheiten der Softwareindustrie im Überblick

Die Softwareindustrie unterscheidet sich grundsätzlich von anderen Branchen. Dies ist zum einen auf die spezifischen Eigenschaften des Produkts Software, zum anderen auf die Struktur von Softwaremärkten zurückzuführen.

Eine Besonderheit von Softwareprodukten besteht darin, dass sie sich, wie jedes andere digitale Gut auch, zu geringen Kosten reproduzieren lassen. Die variablen Kosten gehen also gegen Null. Diese Kostenstruktur führt etwa dazu, dass das Lizenzgeschäft eines Softwareanbieters – zumindest auf den ersten Blick – in der Regel profitabler ist als das Servicegeschäft, wie wir noch ausführlich diskutieren werden. Darüber hinaus kann Software beliebig häufig und ohne Qualitätsverluste kopiert werden. Ist eine Kopie im Internet erst einmal im Umlauf, lassen sich Urheber- bzw. Verfügungsrechte faktisch nicht mehr durchsetzen. Dies gilt insbesondere für wenig erklärungsbedürftige Produkte auf Business-to-Consumer-Märkten. Es ist zudem relativ einfach, von einem einmal erstellten Softwareprodukt verschiedene Versionen oder Bündel zu erstellen und diese zu unterschiedlichen Preisen an verschiedene Kundengruppen zu verkaufen.

Auch für Softwaremärkte gelten einige Besonderheiten. Wie kaum eine andere Branche ist die Softwareindustrie durch eine starke Internationalisierung gekennzeichnet. Software lässt sich global verteilt entwickeln und in Sekundenschnelle zu vernachlässigbaren Kosten über das Internet vertreiben. Daraus resultiert auch ein weltweiter Wettbewerb zwischen den Softwareanbietern. Im Vergleich zu anderen Branchen spielt hierbei der Heimvorteil auf den nationalen Märkten der Anbieter in vielen Segmenten nur noch eine untergeordnete Rolle. So erzielen deutsche Softwareanbieter im Durchschnitt mehr als die Hälfte ihrer Umsätze im Ausland. Der Exportanteil am Umsatz der beiden größten deutschen Softwareanbieter SAP und Software AG betrug in 2012 circa 85 bzw. 80%

(Lünendonk 2013). Zudem führt der Netzeffektcharakter von Software dazu, dass es sich bei Softwaremärkten häufig um so genannte „Winner-takes-it-all“-Märkte handelt. Vor diesem Hintergrund lässt sich beispielsweise auch die Vielzahl von Unternehmensübernahmen erklären.

Diese und andere spezielle ökonomische Prinzipien und Spielregeln werden wir aufgreifen und ausführlich untersuchen. Sie bilden letztlich die Grundlage für die Formulierung von Strategien und Geschäftsmodellen für die Softwareindustrie.

Doch zunächst wollen wir uns in aller Kürze mit der historischen Entwicklung der Softwareindustrie beschäftigen.

1.2 Die Anfänge der Softwareindustrie

Die Softwareindustrie ist eine relativ junge Branche. Die Anfänge gehen auf die frühen fünfziger Jahre zurück, als es noch üblich war, Software und Hardware zu bündeln und gemeinsam zu verkaufen. Die Software war damals also integrierter Bestandteil der Hardware und es wurde noch ausschließlich von Programmcode gesprochen – der Begriff Software wurde erstmals im Jahr 1959 verwendet (Campbell-Kelly 1995, S. 80). In dieser Zeit entstanden in den USA die ersten kleineren Firmen, die Programmcode bzw. Software im Rahmen von individuellen Auftragsprojekten entwickelten (Hoch et al. 2000, S. 27 f.).

Eine Aufwertung der Software erfolgte 1969, als das amerikanische Justizministerium von IBM den getrennten Ausweis von Hardware und Software auf Rechnungen verlangte. In den siebziger Jahren entstand daraufhin eine Reihe von Unternehmen, die sich ausschließlich mit der Entwicklung von Software beschäftigten. Hierbei ist natürlich in erster Linie Microsoft zu nennen: Das von Bill Gates zusammen mit Paul Allen gegründete Unternehmen begann anfangs Programmiersprachen – zunächst BASIC, dann weitere, wie FORTRAN und COBOL – für verschiedene Prozessoren und Rechner zu entwickeln. Erst später entstand im Rahmen einer Zusammenarbeit mit IBM MS-DOS, das zum Standard für Betriebssysteme wurde und wesentlich zur Verbreitung des Personal Computers beitrug (Ichbiah 1993, S. 91–116). Schließlich entschied Microsoft, auch Anwendungen anzubieten und damit in Konkurrenz z. B. zu Lotus zu treten. Bereits 1983 kündigte Bill Gates in der Zeitschrift Business Week an, dass es das Ziel von Microsoft sei, alle PC-Software zukünftig aus einer Hand anzubieten (Ichbiah 1993, S. 141).

Eine parallele Erfolgsgeschichte begann etwa zeitgleich mit den Anfängen von Microsoft im badischen Walldorf, als Dietmar Hopp, Hans-Werner Hector, Hasso Plattner, Klaus Tschira und Claus Wellenreuther ein Unternehmen gründeten, das sich auf die Entwicklung von Software für betriebswirtschaftliche Funktionen und Prozesse spezialisierte – die SAP AG war geboren. Zunächst stellte die Firma Software für Großrechner her, später wurden die Anwendungen für Client-Server-Umgebungen angeboten. Heute ist die SAP das größte europäische Softwarehaus und weltweiter Marktführer im Bereich von Enterprise Resource Planning Software (ERP-Software).

Bereits an diesen beiden Beispielen lässt sich eine Besonderheit der Softwareindustrie erkennen: Am Markt setzt sich häufig nur eine Technologie bzw. ein Anbieter durch. So

verdrängte MS-DOS das Betriebssystem CPM ebenso, wie sich das Tabellenkalkulationsprogramm Excel gegen Konkurrenzprodukte wie Lotus 1-2-3 durchsetzte. Mittlerweile ist Microsoft weltweit führender Anbieter von Office-Anwendungen, Browsern und Betriebssystemen. Ebenso findet auch auf dem Markt für betriebswirtschaftliche Software aktuell eine Konsolidierung statt (siehe hierzu Abschn. 3.1.2). Wir werden auf die Besonderheiten dieser Märkte zurückkommen, wollen uns aber zunächst mit den Akteuren der Softwareindustrie beschäftigen.

1.3 Typen von Softwareanbietern und Auswahlentscheidungen der Anwender

1.3.1 Softwareanbieter im engeren und weiteren Sinne

Betrachten wir im Folgenden die verschiedenen Typen von Softwareanbietern. Hierbei soll zwischen Softwareanbietern im engeren und im weiteren Sinne unterschieden werden.

Die Aufgabe eines Softwareanbieters im engeren Sinne ist die Entwicklung von Software. Dies gilt unabhängig davon, um welche Art von Software es sich hierbei handelt. Software lässt sich nach verschiedenen Merkmalen differenzieren. Ein häufig verwendetes Kriterium ist die Nähe zur Hardware (Mertens et al. 2005). Demnach kann Software in Systemsoftware (z. B. Betriebssysteme), systemnahe Software (z. B. Datenbanksysteme oder Middleware) und Anwendungssoftware (z. B. für Textverarbeitung oder für das Rechnungswesen) unterschieden werden. Ebenfalls lässt sich zwischen Software für kommerzielle und private Anwender unterscheiden. Ein drittes Kriterium zur Klassifikation von Software, das für unsere Betrachtung wesentlich wichtiger ist, ist der Standardisierungsgrad. Als Extremformen sind hier Individualsoftware und Standardsoftware zu nennen.

Individualsoftware wird auf der Basis von spezifischen Anforderungen maßgeschneidert entwickelt, und zwar entweder von den anwendenden Unternehmen selbst – in der Regel in der IT-, in manchen Fällen aber auch in den jeweiligen Fachabteilungen – oder von einem externen Softwarehaus. Die Ausgestaltung der Verträge im Falle der Inanspruchnahme eines solchen Drittanbieters ist jedoch sehr unterschiedlich. Nach nahezu einhelliger Auffassung handelt es sich bei Projekten zur Individualsoftwareherstellung um Werkverträge (Marly 2014, S. 236 f.). Auf eine ökonomische Untersuchung von Dienst- und Werkverträgen gehen wir in Abschn. 2.5 ein. Insbesondere in Indien boomt die Branche im Bereich der Individualsoftwareentwicklung. So haben sich einige erfolgreiche Softwareanbieter mit enormen Wachstumsraten etablieren können, wie etwa Tata Consultancy Services, Wipro Technologies, Infosys Technologies sowie Cognizant Technology Solutions. Daneben existieren aber auch viele kleinere oder mittelständische Firmen, die in Hochlohnländern häufig noch national bzw. regional tätig sind. Demgegenüber sind die Individualsoftwareanbieter mit Sitz in Niedriglohnländern in den meisten Fällen global aufgestellt und stehen miteinander im Wettbewerb. Wir werden auf diese Nearshore- und

Farshore-Anbieter und die sich daraus ergebenden Auswirkungen auf die Softwareindustrie in Abschn. 5.7 näher eingehen.

Standardsoftware wird in der Regel für den Massenmarkt entwickelt. Dabei gehen die Anbieter von weitestgehend standardisierten Bedürfnissen der potenziellen Nutzer aus. Im Folgenden wollen wir kurz die Entstehung und Erfolgsfaktoren von Standardsoftware am Beispiel der SAP AG betrachten.

Das System R von SAP

Die SAP AG ist laut eigenen Angaben führender Anbieter von Unternehmenssoftware und drittgrößter unabhängiger Softwarelieferant der Welt. Weltweit beschäftigt die SAP 66.000 Mitarbeiter und hat Standorte in mehr als 50 Ländern. Die SAP-Lösungen sind bei 251.000 Kunden im Einsatz.

Kernaspekte

Der Erfolg von SAP lässt sich auf drei Kernaspekte zurückführen: die Idee der Entwicklung von Standardsoftware, das Anbieten integrierter Lösungen und Echtzeitverarbeitung. 1981 implementierte das Unternehmen diese Visionen zum ersten Mal mit SAP R/2 und legte seinen Schwerpunkt damit auf die Entwicklung betriebswirtschaftlicher Systeme. 1991 präsentierte SAP das System R/3, das ursprünglich nicht als Ablösung für R/2 gedacht war, sondern dieses als Mittelstandslösung ergänzen sollte. Der generelle Wechsel von Großrechnerlösungen zum Client-Server-Prinzip, der zu dieser Zeit im Gange war, machte R/3 aber gerade für größere Unternehmen interessant und verhalf dem System damit zum Erfolg. Mittlerweile hat SAP seine Produktpalette weiterentwickelt, worauf wir aber an anderer Stelle eingehen.

Standardsoftware

Wie bereits erwähnt, wurde Software anfangs meistens im Rahmen von Individualprojekten entwickelt. Dabei wurden die Software und gegebenenfalls auch die Hardware exakt auf die Anforderungen des Auftraggebers zugeschnitten. Demgegenüber plante SAP von Anfang an die Entwicklung eines standardisierten Systems, das sich mehrfach einsetzen lässt. SAP war eines der ersten Unternehmen, die dieses Konzept konsequent verfolgten.

Integrierte Lösung

Grundlage der Anwendung eines SAP-Systems ist eine integrierte Datenbank, auf die alle Anwendungen zugreifen können. Eine solche integrierte Datenbasis war in Unternehmen in den siebziger Jahren weitgehend unbekannt. Die Folge waren enorme Kosten aufgrund redundanter und inkonsistenter Daten. Darauf aufbauend entwickelte SAP nach und nach Module für verschiedene Funktionsbereiche, wie etwa Rechnungswesen und Controlling, Logistik oder Personalwirtschaft. Diese Pakete waren zunächst primär für industrielle Unternehmen konzipiert und konnten einzeln oder auch gemeinsam zusammen mit einer Datenbank gekauft werden.

Im Gegensatz zu Office-Anwendungen, wie sie etwa von Microsoft angeboten werden, zeigte sich, dass viele der betriebswirtschaftlichen Bereiche branchenspezi-

fische Besonderheiten aufweisen. Daher wird von der SAP mittlerweile eine Vielzahl von Branchenlösungen angeboten, auch für den zunächst weniger beachteten Dienstleistungssektor.

Echtzeitverarbeitung von Daten

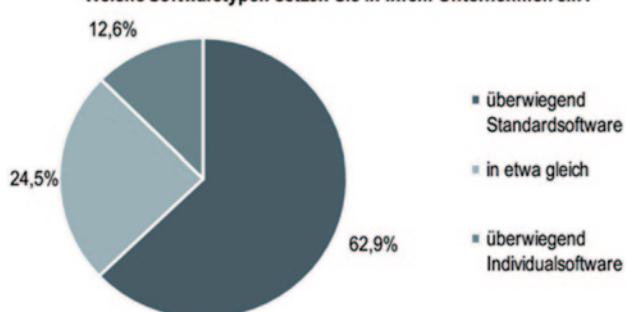
Bis Ende der siebziger Jahre war es in Unternehmen üblich, mit Lochkarten zu arbeiten, d. h., die Daten wurden über Lochkarten in den Computer eingelesen und erst später verarbeitet. Die Idee der Echtzeitverarbeitung war bei den Gründern von SAP von Beginn an ein Kernaspekt und wurde in allen Systemen umgesetzt. Dies erklärt auch das Kürzel „R“ in den Produktnamen von SAP, das für „Realtime“ steht (Leimbach 2007; www.sap.com).

Die Grenze zwischen Individual- und Standardsoftware ist fließend. So kann betriebswirtschaftliche Standardsoftware zumindest bis zu einem gewissen Grad auch an die individuellen Bedürfnisse der Anwender angepasst werden. Allerdings verschlingen Einführungsprojekte, die eine Parametrisierung bzw. ein Customizing sowie gegebenenfalls eine Erweiterung der Software umfassen, häufig Millionenbeträge. Daher ist es für den Anwender in der Regel sinnvoll, lediglich kleinere Anpassungen vorzunehmen, zumal eine komplette Veränderung der Standardsoftware, bis sie 100% auf seine Bedürfnisse zugeschnitten ist, zudem zu Problemen beim Umstieg auf Folgeversionen führen kann. Darauf hinaus bieten neuere Ansätze, wie serviceorientierte Architekturen, dem Anwender zumindest grundsätzlich die Möglichkeit, sich die besten Softwarepakete zu Teilbereichen herauszusuchen, diese auf seine Bedürfnisse anzupassen und mit Hilfe einer Integrationssoftware zu einer individuellen Anwendungslösung zusammenzuführen.

Zukünftig wird der Anteil von Standardsoftwarelösungen im Portfolio der Unternehmen tendenziell zunehmen, wie eine von uns durchgeführte Umfrage unter 498 deutschen CIOs zeigt (Buxmann et al. 2010). So nutzen 62,9% der befragten Unternehmen überwiegend Standardsoftwarelösungen, in 24,5% der Fälle setzen Unternehmen Standard- und Individualsoftware in etwa zu gleichen Teilen ein und bei 12,6% der antwortenden Unternehmen wird überwiegend Individualsoftware eingesetzt (siehe Abb. 1.1).

Abb. 1.1 Anteile
Softwaretypen

Welche Softwaretypen setzen Sie in Ihrem Unternehmen ein?



Neben der Erhebung des Status quo zur Nutzung von Standard- und Individualsoftware wurden die Unternehmen auch gefragt, wie sie die zukünftige Entwicklung bezüglich des Einsatzes dieser Softwaretypen einschätzen. 66,7% der Befragten stimmten der Aussage zu, dass in Unternehmen zukünftig tendenziell mehr Standardsoftware zu Lasten von Individualsoftware eingesetzt werden wird. 20,2% der Unternehmen sprachen sich gegen diese Aussage aus. 13,1% äußerten sich indifferent (siehe Abb. 1.2). 34,0% der Teilnehmer stimmten in diesem Zusammenhang der Aussage zu, dass zukünftig individuelle Anpassungen von Standardsoftware zurückgehen werden, knapp die Hälfte (49,0%) lehnte diese These dagegen ab. Der weitaus größte Teil (71,2%) der antwortenden Unternehmen ist der Auffassung, dass es aufgrund wachsender Anforderungen der Anwender immer wieder individuelle Lösungen für spezielle Problemstellungen geben wird, die in die gesamte IT-Landschaft zu integrieren sind.



Abb. 1.2 Einschätzung der zukünftigen Entwicklung für die Anteile der Softwaretypen

Zukünftig werden Unternehmen somit mehr Standardsoftwarelösungen einsetzen; allerdings werden weiterhin individuelle Anpassungen notwendig sein. Für Spezialprobleme, für die es (noch) keine standardisierte Anwendungssoftware auf dem Markt gibt, wird man nach Maßgabe von Wirtschaftlichkeitsüberlegungen tendenziell nach wie vor Individualsoftware entwickeln oder entwickeln lassen, die in die IT-Landschaft der Unternehmen einzubinden ist.

Neben Softwareanbietern im engeren Sinne – also den Unternehmen, die Standard- oder Individualsoftware entwickeln – gibt es Anbieter für Dienstleistungen bzw. Services in den späteren Phasen im Lebenszyklus einer Softwarelösung. Wir bezeichnen diese Unternehmen im Folgenden als Softwareanbieter im weiteren Sinne. Diese Dienstleistungen umfassen zum einen die Unterstützung von Anwendern bei der Implementierung der Softwarelösungen sowie zum anderen den Betrieb der Produkte. Abbildung 1.3 fasst unsere Klassifikation von Softwareanbietern zusammen.

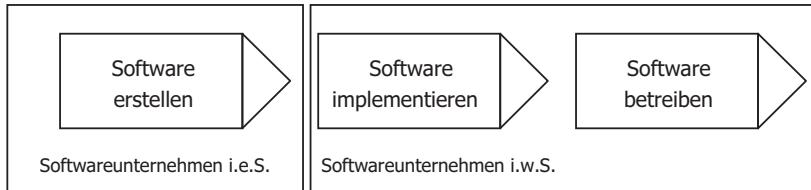


Abb. 1.3 Klassifikation von Softwareunternehmen

Insbesondere für erklärungsbedürftige Softwarelösungen, also solche, die sich nicht leicht implementieren und in ein Anwendungsumfeld integrieren lassen, existiert eine enorme Nachfrage nach Dienstleistungen. Demzufolge gibt es eine Vielzahl von Anbietern auf diesem Markt. Dabei handelt es sich um (Lünendonk 2009).

- IT-Beratungs- und Systemintegrations-Unternehmen (mit Angeboten im Bereich IT-Beratung, Entwicklung von Individualsoftware etc.),
- IT-Serviceunternehmen (mit Leistungen wie Outsourcing, ASP, Schulung etc.),
- Business Innovation/Transformation Partner (Management- und IT-Beratung sowie System-Realisierung).

Eine besonders große Rolle spielt traditionell die Unterstützung der Anwender bei SAP-Einführungsprojekten. Dies liegt daran, dass – wie bereits angesprochen – die Software im Rahmen dieser Projekte an die speziellen Bedürfnisse der Anwender angepasst wird. Entsprechende Kenntnisse liegen in den Anwendungsunternehmen in vielen Fällen nicht vor oder es stehen dort nicht genügend Mitarbeiter mit einem solchen Know-how zur Verfügung. Diese Einführungsprojekte sind grundlegend für den Nutzen, den die Software für die Kunden stiftet, da sie direkt in die inner- sowie zwischenbetrieblichen Prozesse eingreift. In diesem Zusammenhang sind häufig auch kleinere Programmierarbeiten, etwa die Entwicklung von Schnittstellen zwischen heterogenen Systemen, zu erledigen.

Dabei setzt dieses Geschäft wie kaum ein anderes eine Vertrauensbasis zwischen den Kunden und den Dienstleistungsanbietern voraus. Hoch et al. (2000, S. 160) benutzen in diesem Zusammenhang sogar den Begriff des „Glaubens“: Der Kunde muss daran glauben, dass das Dienstleistungsunternehmen bzw. die für das Projekt eingesetzten Mitarbeiter ein hervorragendes Know-how haben und in der Lage sind, ihr Versprechen zu halten, die Probleme der Kunden zu lösen. Denn während ein Anwender die Möglichkeit hat, die Software eines Standardsoftwareherstellers zu testen, ist dies im Beratungsgeschäft nicht möglich. Dem Kunden bleibt häufig nichts anderes übrig, als seine Entscheidung auf die Referenzen der Beratungshäuser und Systemintegratoren zu stützen. Daher ist es auch das vorrangige Ziel der Marketingaktivitäten von diesen Dienstleistungsunternehmen, Vertrauen in ihre Kompetenz aufzubauen. Hierzu gehören insbesondere die folgenden Maßnahmen (Hoch et al. 2000, S. 162–178):

- Sponsoring von IT-Konferenzen,
- Diskussionszirkel mit hochrangigen Vertretern der IT- und Softwareindustrie,
- Veröffentlichungen in Fachmagazinen und wissenschaftlichen Zeitschriften sowie
- Anzeigen und TV-Spots.

Aufgrund der Komplexität von Einführungs- und Integrationsprojekten ist die Auswahl des Dienstleistungsunternehmens für den Kunden von zentraler Bedeutung. Dies gilt insbesondere auch, weil in den meisten Fällen das Budget überschritten wird bzw. viele dieser Projekte komplett scheitern. So wurden nach einer Untersuchung der Standish Group International, Inc. lediglich 32 % der betrachteten IT-Projekte innerhalb der Zeit- und Budgetvorgaben erfolgreich abgeschlossen (The Standish Group International, Inc. 2009, S. 2). 44 % überschritten diese Planwerte erheblich und 24 % wurden überhaupt nicht zu Ende geführt. Bei den als komplex eingestuften Projekten wurden Durchlaufzeit und Budget um mehr als das Doppelte überschritten.

Der Betrieb von IT-Lösungen gehört seit Jahren zum Outsourcing-Geschäft (siehe hierzu auch Kap. 5). Hier gibt es eine Vielzahl von Playern in einem weltweit umkämpften Markt, die wir auch zu den Softwareanbietern i. w. S. zählen.

1.3.2 Die Auswahl von Software

Grundlage der Entwicklung und Gestaltung von Geschäftsmodellen für Softwareanbieter im engeren sowie im weiteren Sinne sind die Auswahlentscheidungen der Kunden. Die folgenden Abschnitte sollen aufzeigen, welche Prozesse und Entscheidungsfaktoren bei der Auswahl und Bewertung alternativer Softwarelösungen in Unternehmen (Abschn. 1.3.2.1) und Haushalten (Abschn. 1.3.2.2) existieren.

1.3.2.1 Softwareauswahl in Unternehmen

In der Regel durchlaufen Unternehmen mehrere sequenzielle Phasen bis zur endgültigen Auswahlentscheidung (siehe Abb. 1.4). Hierbei wird der Lösungsraum der zur Verfügung stehenden Alternativen schrittweise eingeengt. Typische Kontextvariablen sind Unternehmensaspekte, wie z. B. Branchenzugehörigkeit, Größe oder zu unterstützende Funktionen, bzw. Umfeldvariablen, wie etwa Technologiestandards. Aus den Rahmenbedingungen heraus werden Ziele definiert, die mit der Anschaffung der Software verfolgt werden sollen. Der nächste Schritt stellt in der Regel die Definition von Kriterien dar. Kriterien sind relevante Merkmale für die Beschreibung eines Softwaresystems und die Grundlage für eine Bewertung.



Abb. 1.4 Softwareauswahl in Unternehmen

Mit Hilfe von Marktübersichten werden potenzielle Alternativen ermittelt. Typisch ist das sich anschließende zweistufige Bewertungs- und Auswahlverfahren. Bei der Grobauswahl werden die Alternativen anhand so genannter K.O.-Kriterien bewertet. Ihre Erfüllung ist notwendige Voraussetzung für eine detailliertere Untersuchung. Für die verbliebenen Alternativen werden auf Grundlage detaillierter Anforderungsdefinitionen (Pflichtenhefte), die aus den Kriterien abgeleitet werden, entsprechende Angebote eingeholt bzw. Informationen angefordert. Basierend auf den auf diese Weise gewonnenen Informationen wird eine Feinauswahl vorgenommen, die in einen Entscheidungsvorschlag mündet. Einige Ansätze berücksichtigen ferner die an die Entscheidung anschließenden Phasen, wie Vertragsverhandlungen und funktionale Testläufe zur Bestimmung des Anpassungsaufwandes.

Im Folgenden soll detaillierter auf die Entwicklung eines Zielsystems, die Kriterienableitung und -gewichtung sowie auf die während der Grob- und Feinauswahl relevanten Aktivitäten eingegangen werden.

Die Formulierung der Ziele, die mit der Einführung bzw. dem Einsatz einer Software in einem bestimmten Bereich erreicht werden sollen, stellt in der Regel den Ausgangspunkt der Analyse und den Maßstab zur Beurteilung der Alternativen dar. Bei der Ableitung der Ziele können unterschiedliche Verfahren zum Einsatz kommen: Zum einen werden Ziele top-down aus den obersten Unternehmenszielen bzw. aus der IT-Strategie abgeleitet. Das kann in der Weise geschehen, dass oberste Unternehmensziele, wie beispielsweise die Gewinnsteigerung, auf das konkrete Entscheidungsproblem „Auswahl einer Standardsoftware für ...“ übertragen werden. Eine Variante dieser Zielableitung stellt eine Konkretisierung der obersten Ziele durch Unterziele im Sinne einer Zweck-Mittel-Beziehung bezogen auf das Entscheidungsproblem dar. Beispiele hierfür sind die Senkung der Durchlaufzeit, die Senkung von Lagerbeständen oder die Verkürzung von Reaktionszeiten.

Ein anderer Ansatz geht entgegengesetzt zur oben vorgestellten Strategie bottom-up vor. Diese Ziele haben einen stark inhaltlich-funktionalen Fokus mit Bezug zum konkreten Entscheidungsproblem bzw. zu existierenden Strukturen (wie z. B. die bestehende IT-Architektur). Beispiele sind Ziele in Form allgemeiner Softwaremerkmale, wie Datenbank- oder Hardwareherstellerunabhängigkeit, Kompatibilität mit bestehenden Versionen oder spezifische anwendungsbezogene Funktionalitäten, wie beispielsweise eine automatische Teilenummernvergabe mit Prüfziffer oder die Verbreiterung der Informationsbasis. Diese Aspekte stellen oftmals eher bereits Anforderungen dar und nicht Ziele im engeren Sinn, die um ihrer selbst willen verfolgt werden. In der Praxis wird deshalb meist ein kombiniertes Verfahren aus top-down und bottom-up-Vorgehensweise zur Zielbestimmung als Voraussetzung der Kriterienableitung angewendet. Allerdings wird auf das erforderliche Gegenstromprinzip im Sinne eines Abgleichens und Anpassens oftmals zugunsten einer vereinfachten Aufzählung verzichtet.

Werden Kriterien aus dem Zielsystem abgeleitet, so stellen sie in der Regel die unterste Ebene eines solchen Systems dar. Die Auswirkung einer bestimmten Systemeigenschaft auf die angestrebten Ziele ist damit transparent. Im Sinne der Zielhierarchie werden die Auswahlkriterien häufig in software-, implementierungs- und anbieterbezogene Merkmale

untergliedert. Neben den eigentlichen Leistungsmerkmalen der Software spielen bei komplexeren Systemen insbesondere Implementierungskriterien eine wesentliche Rolle, da sie neben den Anschaffungskosten erhebliche Auswirkungen auf die Gesamtaufwendungen für die Software haben können. Die Implementierung und Einführung komplexerer Software-Systeme werden oftmals von Softwareanbietern bzw. von Implementierungspartnern begleitet, deren Expertise und Professionalität im Vorfeld der Softwareauswahl in der Regel auch zu evaluieren ist. Neben rein rationalen Kriterien spielen darüber hinaus in der Praxis auch häufig politische Kriterien (z. B. Absprachen, Klüngeli, Machtspiele) eine nicht zu unterschätzende Rolle, die den Auswahlprozess erheblich beeinflussen können (Howcroft und Light 2006).

Tabelle 1.1 zeigt für das Beispiel der Auswahl von ERP-Systemen, welche konkreten Auswahlkriterien für die Dimensionen „Software“, „Implementierung“ und „Anbieter“ in der Praxis herangezogen werden (Keil und Tiwana 2006; Jadhav und Sonar 2009).

Tab. 1.1 Typische Auswahlkriterien bei der Auswahl von ERP-Systemen

Auswahlkriterien (Software)	Auswahlkriterien (Implementierung/Anbieter)
Funktionalität	Integrationsfähigkeit in bestehende IT-Architektur
Kosten	Implementierungszeit/Anpassungsaufwand
Benutzerfreundlichkeit	Support durch Anbieter
Zuverlässigkeit	Reputation des Anbieters

Hinsichtlich der Gewichtung der Einzelkriterien im Rahmen der Gesamtentscheidung wird in der Regel davon ausgegangen, dass die Kriterien vorhanden bzw. erfüllt sein sollen. Andererseits ist auch denkbar, dass die Erfüllung eines Kriteriums in Abstufungen möglich ist. In einem solchen Fall ist die Angabe eines angestrebten Ziellniveaus bzw. die Beurteilung von unterschiedlichen Ziellniveaus erforderlich. So ist allein aus einem Kriterium „Datenbankunabhängigkeit“ nicht ersichtlich, ob hierzu die Unterstützung von zwei, drei oder zehn unterschiedlichen Datenbankmanagementsystemen erforderlich ist. Dies ist insbesondere zur Vermeidung von Überbewertungen von Bedeutung, wenn etwa bezüglich eines Ziels lediglich ein satisfizierendes Ziellniveau angestrebt wird. Auf der anderen Seite liegen häufig auch Ziele vor, bei denen ein bestimmtes Anspruchsniveau mindestens erreicht werden muss, damit eine Alternative überhaupt in Betracht gezogen bzw. detaillierter untersucht wird. Solche K.O.- oder Killer-Kriterien tragen zwar einerseits dazu bei, den Auswahlprozess effizient zu gestalten. Andererseits bringen sie aber auch die Gefahr mit sich, dass Alternativen aufgrund einer auch nur geringen Unterschreitung des Mindestanspruchsniveaus eines Ziels ausgesondert werden.

Zur Gewichtung der Kriterien wird typischerweise ein holistischer Ansatz angewendet, wie er beispielsweise aus der Nutzwertanalyse bekannt ist. Dabei erfolgt die Vergabe der Gewichte durch einen direkten Vergleich der Kriterien. So wird z. B. in empirischen Studien zur Auswahl von ERP-Systemen berichtet, dass die Kriterien Funktionalität, Zuver-

lässigkeit und Kosten am meisten ins Gewicht fallen, während die Benutzerfreundlichkeit und der Implementierungsaufwand als weniger wichtig bewertet werden (Keil und Tiwana 2006). Bei diesem Ansatz erscheint die Vergabe sehr differenzierter Gewichtungen jedoch problematisch. Insbesondere bei komplexen Problemen, zu denen die Softwareauswahl u. a. aufgrund der großen Anzahl von Bewertungskriterien gehört, besteht allenfalls die Möglichkeit der Rangfolgenbildung in ordinaler Skalierung. Exaktere Angaben unterstellen (Schein-) Genauigkeiten, die in den seltensten Fällen tatsächlich die Präferenz des Entscheiders wiedergeben.

Marktüberblick, Alternativenvorauswahl und -bewertung vollziehen sich in der Praxis häufig interdependent und miteinander vernetzt. Schon die Berücksichtigung bzw. die Nichtberücksichtigung eines Systems beim Marktüberblick stellt eine Auswahl dar. Dieser erste Auswahlschritt ist aufgrund der großen Marktbreite geprägt durch heuristische Ansätze, wie z. B.:

- Berücksichtigung von Systemen, die die Mitbewerber einsetzen,
- Berücksichtigung von Systemen, die gegenwärtig in der Fachpresse große Beachtung finden (z. B. Google Apps Premier, SAP ERP u. a.),
- Berücksichtigung von Systemen, mit denen Mitarbeiter bereits Erfahrungen haben,
- Berücksichtigung von Systemen, bei deren Auswahl sich ein Berater einen Anschlussauftrag im Rahmen der Implementierung erhofft,
- Berücksichtigung von Systemen aufgrund einer zufälligen Auswahl, z. B. durch Messbesuche.

Wie bereits dargestellt, bilden Bewertung und Auswahl einen gekoppelten Prozess, wobei die Anzahl der Alternativen mittels K.O.-Kriterien verringert werden kann. Dabei werden typischerweise die folgenden Verfahren angewendet:

- Auswertung von Systembeschreibungen,
- Auswertung bearbeiteter Ausschreibungsunterlagen bzw. Anforderungskataloge durch die potenziellen Anbieter sowie
- Präsentationen des Systems bzw. konkrete Funktionspräsentation auf Basis definierter Anwendungsszenarien.

Bei der Bewertung bestimmter Kriterien wird dabei häufig eine dreistufige Skala mit den Ausprägungen „erfüllt“, „teilweise erfüllt“, „nicht erfüllt“ zugrunde gelegt. Schließlich besteht das Bedürfnis, die Ergebnisse der Einzelbewertung zu einer übergreifenden Bewertung zusammenzufassen.

Da die Eignung eines Systems nicht nur aus seiner funktionalen Übereinstimmung mit den gestellten Anforderungen resultiert, sondern auch von den Kosten des Systems abhängt, werden diese häufig in der Weise in die Bewertung einbezogen, dass ein Kosten-Nutzen-Koeffizient gebildet wird. Auf Basis des Vergleichs entsprechender quantitativer und qualitativer Bewertungen wird schließlich die Auswahlentscheidung getroffen.

1.3.2.2 Softwareauswahl durch Konsumenten

Während es sich bei Auswahlentscheidungen in Unternehmen in der Regel um Gruppenentscheidungen handelt, werden Konsumentenentscheidungen meistens alleine getroffen. Ähnlich wie bei Auswahlentscheidungen im Unternehmenskontext durchläuft ein Konsument bei der Auswahl von Produkten im Allgemeinen und Software im Speziellen typischerweise vier Phasen (Blackwell et al. 2003), die in Abb. 1.5 illustriert werden.

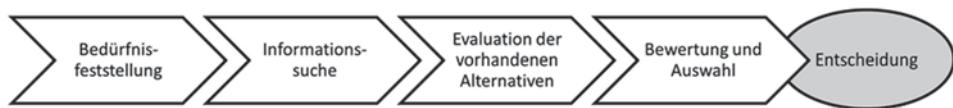


Abb. 1.5 Prozess zur Auswahl von Software durch Konsumenten

Nachdem ein Konsument sich des Bedürfnisses bewusst wird, eine neue Software kaufen zu wollen (Phase 1: Bedürfnisfeststellung), sucht er anschließend nach geeigneten Informationen über alternative Produkte, die das Bedürfnis befriedigen sollen. Dabei identifiziert der Konsument zunächst – mehr oder weniger bewusst – wesentliche subjektive und objektive Kriterien für seine Entscheidungsfindung (Phase 2: Informationssuche). Nach der Informationssammlung verarbeitet der Konsument die vorhandenen Daten und evaluierter jede Alternative vor dem Hintergrund seiner persönlichen Zielkriterien (Phase 3: Evaluation der Alternativen). Schließlich vergleicht und bewertet der Konsument die vorhandenen Alternativen und führt daraufhin eine Auswahl- bzw. Kaufentscheidung durch (Phase 4: Bewertung und Auswahl).

Im Gegensatz zu Auswahlentscheidungen im Unternehmen spielen bei Auswahlentscheidungen von Konsumenten neben rein objektiven Kriterien (wie Kosten oder Funktionalität einer Software) insbesondere individuelle bzw. interne (wie z. B. die Persönlichkeit, Emotionen, Wahrnehmungen oder der Lebensstil des Konsumenten) und situative bzw. externe Faktoren (z. B. Kultur oder Mund-zu-Mund-Propaganda) eine Rolle. Um aus Anbietersicht besser verstehen zu können, welche wesentlichen Kriterien Konsumenten bei der Software-Auswahl heranziehen, sollen diese nachfolgend vertiefend beleuchtet werden.

Wie oben bereits angeführt, spielen bei Konsumentenentscheidungen neben rein rationalen Auswahlkriterien insbesondere weichere Faktoren eine Rolle. Ein umfassendes Modell, das diesen Besonderheiten speziell in digitalen Produktwelten Rechnung trägt, ist das Modell der Erfahrungswerte digitaler Güter nach Mathwick et al. (2001; siehe Abb. 1.6). Es veranschaulicht, dass digitale Produkte (wie z. B. Spielesoftware) für Konsumenten einen Erfahrungswert besitzen, der vor dem eigentlichen Kauf (bewusst oder unbewusst) evaluiert wird. Der Erfahrungswert setzt sich dabei aus utilitaristischen und hedonischen Merkmalen zusammen.

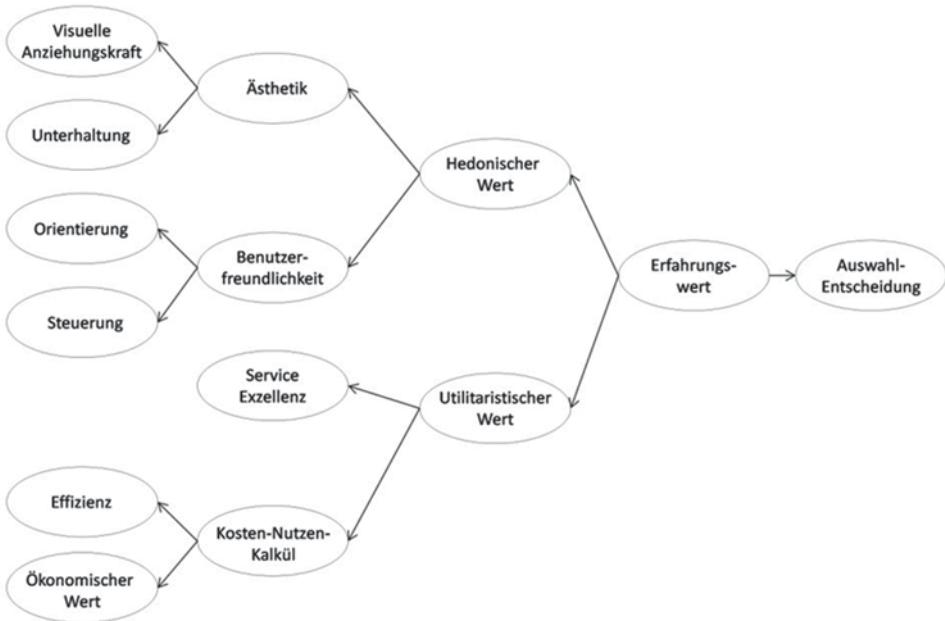


Abb. 1.6 Einflussfaktoren des Erfahrungswertes digitaler Güter. (in Anlehnung an Mathwick et al. (2001))

Utilitaristische Merkmale sind überwiegend objektiver Natur und entsprechen typischerweise Kosten-Nutzen-Kalkülen. In die Kosten-Nutzen-Betrachtung fließen wiederum weitgehend rationale Bewertungskriterien, wie z. B. die Funktionalität, Kompatibilität oder die Anschaffungskosten der Software, ein. Der mit der Software einhergehende Service beschreibt auf der anderen Seite, welche Unterstützungs- und Garantieleistungen (z. B. Hotline oder Gewährleistungsansprüche) der Nutzer beim Erwerb der Software erhält.

Hedonische Merkmale sprechen dagegen die oben angeführten persönlichen und situativen Bedürfnisse von Konsumenten an. So spielen ästhetische Vorlieben von Konsumenten bei der Auswahl eine große Rolle wie auch die Benutzerfreundlichkeit der Software. Beide Faktoren umfassen konkretere Subkriterien, wie z. B. die visuelle Anziehungskraft (etwa Farbgestaltung, 3D-Effekte) oder die Art und Weise der Steuerung der Software (z. B. mit Maus, Tastatur, Controller), die wiederum spezifische, individuelle Kaufmotive des Konsumenten ansprechen. Insgesamt lassen sich über die Beeinflussung der Subkriterien die darüber liegenden hedonischen und utilitaristischen Werte und somit der gesamte Erfahrungswert einer Software beeinflussen, der wiederum einen Einfluss auf die Software-Auswahl und Kaufentscheidung des Konsumenten ausübt. Die Subkriterien lassen sich in diesem Sinne mehr oder weniger stark vonseiten eines Softwareanbieters beeinflussen.

1.4 Dienstleistungserlöse in der Softwareindustrie

Neben den Lizenzennahmen erzielen Softwareanbieter in zunehmendem Maße Dienstleistungserlöse. Diese Erlösquelle ist dabei nicht allein den Individualsoftwareherstellern sowie den Softwareanbietern im weiteren Sinne vorbehalten. Auch für Standardsoftwarehersteller bieten sich hier verschiedene Möglichkeiten, Umsätze zu generieren, etwa indem sie selbst Services anbieten, wie z. B. Beratungs- sowie Wartungsdienstleistungen.

Während die Beratungsdienstleistungen meistens nach Beratertagen (und gelegentlich auch erfolgsabhängig) abgerechnet werden, sehen die Erlösmodelle für den Bereich Wartungsservices in der Regel vor, dass der Anwender jährlich einen Prozentsatz der Lizenzgebühren an den Softwareanbieter zahlt. Dieser Prozentsatz variiert von Anbieter zu Anbieter – eine typische Größe liegt bei ca. 20 % pro Jahr. Damit wird unmittelbar deutlich, von welcher Bedeutung diese Erlösquelle für die Softwareanbieter ist: Wenn wir davon ausgehen, dass eine bestimmte Softwarelösung im Durchschnitt sieben bis zehn Jahre genutzt wird, bevor sie durch eine neue Version oder – seltener – durch ein Alternativprodukt ersetzt wird, zeigt sich zum einen, dass die Wartungserlöse die Lizenzröhre in der Regel übersteigen; dies gilt auch nach Diskontierung der Erlöse. Zum anderen ergibt sich aus diesem Geschäftsmodell auch, dass Wartungserlöse relativ konstant über den Zeitverlauf anfallen und – im Gegensatz zu den Lizenzennahmen – eine relativ gut planbare Erlösquelle für die Softwareanbieter sind. Abbildung 1.7 zeigt, dass die meisten der großen Standardsoftwareanbieter ihren Anteil des Servicegeschäfts am Gesamtumsatz in den letzten Jahren recht konstant halten konnten.

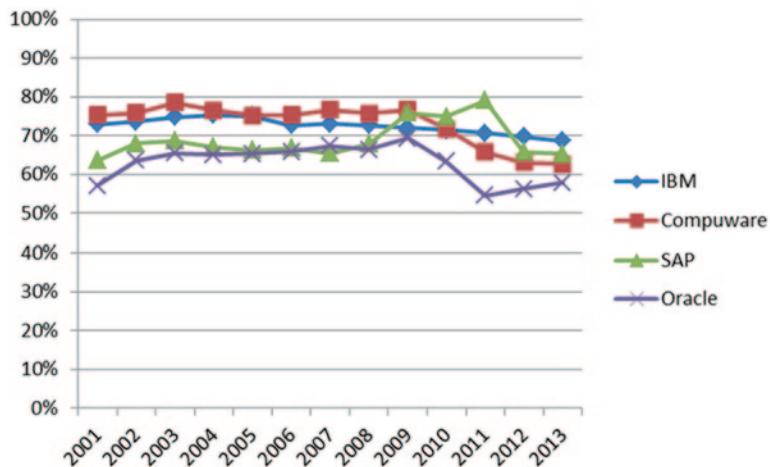


Abb. 1.7 Anteil des Servicegeschäfts am Umsatz großer Standardsoftwarehersteller. (erweiterte eigene Berechnungen auf Basis von Cusumano (2004, S. 37))

Dass die Serviceumsätze die Lizenzeinnahmen in vielen Fällen deutlich übersteigen, bedeutet jedoch nicht, dass das Servicegeschäft grundsätzlich attraktiver als der Verkauf von Lizenzen ist. Betrachten wir die Profitabilität der verschiedenen Erlösquellen, so liegt diese für das Lizenzgeschäft im Regelfall bei etwa 90–100 % (Cusumano 2004, S. 4 f.). Variable Kosten, wie zum Beispiel für Datenträger, Handbücher, Dienstleistungsanbieter oder Verpackung, sind meist niedrig – beim Vertrieb über das Internet sind sie sogar vernachlässigbar gering. Demgegenüber liegen die Deckungsbeiträge für Beratungs- und Wartungsdienstleistungen deutlich darunter.

Von großer Bedeutung ist die Profitabilität der unterschiedlichen Geschäftsbereiche für Investoren, die sich in vielen Fällen eher an Renditen als an absoluten Werten orientieren. Damit sind Softwareanbieter, die sich überwiegend auf den Produkt- bzw. Lizenzverkauf konzentrieren, häufig für Investoren attraktiver als Softwareanbieter im weiteren Sinne. Dies liegt insbesondere daran, dass reine Produktanbieter schneller wachsen können als Unternehmen, die einen Großteil ihrer Umsätze mit Services erzielen. Daher haben viele Softwareunternehmen auch keine Beratungssparte. Andererseits haben Softwareanbieter, die einen großen Teil ihres Umsatzes mit Services erzielen, den Vorteil, dass sie – wie bereits oben erwähnt – relativ konstante Erlöse erzielen, was gerade in schlechteren wirtschaftlichen Zeiten mit wenig Neugeschäft von hoher Bedeutung ist.

Dabei stehen die in diesem Abschnitt untersuchten Lizenz- und Dienstleistungserlöse natürlich nicht unabhängig nebeneinander. Vielmehr werden die Höhe der Lizenerlöse sowie die Preisstrategien der Anbieter die Dienstleistungseinnahmen maßgeblich mitbestimmen. So kann ein Geschäftsmodell in der Softwareindustrie etwa darin bestehen, Software zu niedrigen Preisen zu verkaufen oder gar zu verschenken. Dies ist häufig eine viel versprechende und zum Teil sogar notwendige Strategie, um frühzeitig Marktanteile zu gewinnen oder einem etablierten Wettbewerber Marktanteile abzunehmen. Unabhängig davon kann es auch interessant sein, komplementäre Dienstleistungen zu der günstig abgegebenen oder verschenkten Software anzubieten und die Umsätze ausschließlich oder überwiegend mit Services zu generieren.

Solche Geschäftsmodelle sind insbesondere für die Unternehmen anwendbar, die digitale Güter, wie etwa Software oder Musik, anbieten. Über die Erlösgenerierung hinaus hat das Gut Software einige weitere spezifische Eigenschaften, die es von anderen Produkten unterscheidet. Auf die ökonomischen Hintergründe und die Folgen für Softwaremärkte wollen wir im folgenden zweiten Kapitel eingehen.

Literatur

- Blackwell RD, Miniard PW, Engel JF (2003) Consumer behavior. Harcourt, Orlando
- Buxmann P, Brandt B, von Ahsen A, Hess T (2010) Outsourcing der Entwicklung und Anpassung von Anwendungssoftware: analyse der Kundenzufriedenheit auf Basis einer empirischen Untersuchung. Darmstädter Arbeitspapier
- Campbell-Kelly M (1995) Development and structure of the international software industry 1950–1990. Bus Econ Hist 24:73–110

- Cusumano MA (2004) *The business of software: what every manager, programmer and entrepreneur must know to succeed in good times and bad*. Simon & Schuster, New York
- Hoch DJ, Roeding CR, Purkert G, Lindner SK (2000) Erfolgreiche Software-Unternehmen. Die Spielregeln der New Economy. Hanser, München
- Howcroft D, Light B (2006) Reflections on issues of power in packaged software selection. *Inf Syst J* 16:215–235
- Ichbiah D (1993) *Die microsoft story: Bill Gates und das erfolgreichste Softwareunternehmen der Welt*. Campus, München
- Jadhav AS, Sonar RM (2009) Evaluating and selecting software packages: a review. *Inf Softw Technol* 51:555–563
- Keil M, Tiwana A (2006) Relative importance of evaluation criteria for enterprise systems: a conjoint study. *Inf Syst J* 16:237–262
- Leimbach T (2007) Vom Programmierbüro zum globalen Softwareproduzenten. Die Erfolgsfaktoren der SAP von der Gründung bis zum R/3-Boom. *Z Unternehmensgeschichte* 52:5–34
- Lünendonk T (2009) Führende IT-Beratungs-, IT-Service- und Standard-Software-Unternehmen in Deutschland. Bad Wörishofen
- Lünendonk T (2013) TOP 25 der Standard-Software-Unternehmen in Deutschland 2012. http://luenendonk.de/wp-content/uploads/2013/05/LUE_Liste_u_PI_2013_Standard_Software_f160520131.pdf
- Marly J (2014) *Praxishandbuch Softwarerecht*, 6 Aufl. Beck, München
- Mathwick C, Malhotra N, Rigdon E (2001) Experiential value: conceptualization, measurement and application in the catalog and internet shopping environment. *J Retailing* 77:39–56
- Mertens P, Große-Wilde J, Wilkens I (2005) Die (Aus-)Wanderung der Softwareproduktion – Eine Zwischenbilanz, Bd 38. Institut für Informatik der Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen-Nürnberg
- The Standish Group International, Inc. (2009) New Standish Group report shows more project failing and less successful projects. http://www1.standishgroup.com/newsroom/chaos_2009.php. Zugegriffen: 23. April 2009, Massachusetts

Ökonomische Prinzipien der Softwareindustrie

2

Nachdem wir im ersten Kapitel einige grundlegende Spielregeln der Softwareindustrie dargestellt haben, wenden wir uns im Folgenden den ökonomischen Prinzipien dieser Branche zu. Ausgangspunkt sind die Eigenschaften digitaler Güter (Abschn. 2.1). Im nächsten Schritt untersuchen wir Netzeffekte auf Softwaremärkten (Abschn. 2.2) und das damit eng verbundene Standardisierungsproblem (Abschn. 2.3). Darüber hinaus beschäftigen wir uns mit den Aspekten der Transaktionskosten- (Abschn. 2.4) und Principal-Agent-Theorie (Abschn. 2.5), die für die Softwareindustrie von besonderer Bedeutung sind.

2.1 Eigenschaften digitaler Güter

Zu Beginn des ersten Kapitels haben wir bereits einige ökonomische Eigenschaften von digitalen Gütern und Softwareprodukten vorgestellt. Diese Überlegungen sollen nun vertieft werden. Ein wesentliches Merkmal digitaler Güter besteht darin, dass die Erstellung der First Copy in der Regel zu hohen Kosten führt, die Reproduktion jedoch zu sehr geringen variablen Kosten möglich ist. Sehen wir uns diesen Zusammenhang für die Softwareindustrie einmal näher an. Die Entwicklung einer Software (genauer gesagt: des Quellcodes als First Copy) erfordert häufig erst einmal hohe Investitionen. Dabei ist jedoch keineswegs gewährleistet, dass ein Entwicklungsvorhaben auch tatsächlich zum Erfolg führt. Ein Individualsoftwareanbieter ist daher geneigt, die Finanzierung der Entwicklungskosten sowie das Risiko ganz oder zumindest teilweise auf seinen Auftraggeber abzuwälzen (siehe hierzu auch Abschn. 2.5). Einem Standardsoftwareanbieter bietet sich diese Option hingegen nicht; er trägt das komplette Risiko, kann im Erfolgsfall aber auch überproportional profitieren. Bei den Entwicklungskosten handelt es sich darüber hinaus um so genannte „sunk costs“, d. h. um Kosten, die unwiderruflich entstanden sind, nicht mehr beeinflusst werden können und damit auch nicht mehr entscheidungsrelevant sind.

Bereits an dieser Stelle wollen wir jedoch schon darauf hinweisen, dass die Theorie digitaler Güter die Realität in ihren Annahmen hinsichtlich der gegen Null tendierenden variablen Kosten stark und in vielen Fällen wohl auch zu stark vereinfacht. So gehen die variablen Kosten lediglich für Softwarelizenzen gegen Null. Nicht vernachlässigbar sind diese Kosten jedoch für die Dienstleistungen, die rund um die Softwareprodukte angeboten werden, beispielsweise für Beratung, Wartung und Support. Wir werden insbesondere im dritten Kapitel auf diese Zusammenhänge zurückkommen.

Eine weitere wichtige Eigenschaft digitaler Güter besteht darin, dass sie einfach und ohne Qualitätsverluste kopiert werden können. Aus dieser einfachen Kopierbarkeit resultieren unter anderem die niedrigen Kosten der Reproduktion. Kopien digitaler Güter werden auch als perfekt bezeichnet, da zwischen dem Original und dem Duplikat keinerlei Unterschiede mehr bestehen.

Die daraus resultierenden Probleme sind etwa aus der Musikindustrie bekannt. Auf verschiedenen Peer-to-Peer-Plattformen werden Musikstücke zum Teil illegal zum kostenlosen Download angeboten. Diese Problematik betrifft auch die Softwareindustrie und hier insbesondere jene Anbieter, deren Produkte zur Nutzung nicht oder nur wenig angepasst werden müssen und die auch für private Anwender interessant sind. So werden beispielsweise Office-Anwendungen und Computerspiele häufig illegal kopiert bzw. über Filesharing-Netze ausgetauscht. In mehreren Studien wurde versucht, die Umsatzeinbußen der Softwarefirmen durch Raubkopien zu schätzen, beispielsweise in der jährlich erscheinenden „Piracy Study“ des Industrieverbands BSA (Business Software Alliance). Die Methodik und die diesen Studien zugrunde liegenden Annahmen sind allerdings umstritten. So wird vielfach einfach unterstellt, dass jede illegal bezogene Kopie zu einem Umsatzverlust führt, also jeder Raubkopierer das entsprechende Softwareprodukt ansonsten auch gekauft hätte. Dies führt zu immensen (rechnerischen) Umsatzeinbußen, die in dieser Größenordnung wenig realistisch erscheinen, obgleich es unstrittig ist, dass die entsprechenden Softwareanbieter tatsächlich Verluste durch die Anfertigung illegaler Kopien hinnehmen müssen.

Zum Schutz des geistigen Eigentums bzw. zum Verhindern illegaler Kopien können Anbieter digitaler Güter grundsätzlich Digital-Rights-Management-Systeme einsetzen (Hess und Ünlü 2004). Diese stellen mittels Hard- oder Softwareimplementierungen entsprechende Schutzverfahren bereit. Dabei kann es sich etwa um Zugangs- und Nutzungs-kontrollen, den Schutz der Authentizität und Integrität, die Identifikation über Metadaten, die Anwendung eines Kopierschutzes oder ein bestimmtes Bezahlsystem handeln.

2.2 Netzeffekte auf Softwaremärkten: The Winner Takes it All

In diesem Abschnitt beschäftigen wir uns mit Netzeffekten, die auf Softwaremärkten eine wesentliche Rolle spielen. Denn neben der Funktionalität einer Software hat insbesondere auch deren gegenwärtige und zukünftig erwartete Verbreitung einen großen Einfluss auf den Nutzen für die Anwender. Dieser Zusammenhang wird im Rahmen der Theorie der

positiven Netzeffekte beschrieben und analysiert. Der erste Teil dieses Abschnitts enthält wesentliche Grundlagen und Definitionen (Abschn. 2.2.1). Danach beschreiben wir, warum sich nicht immer die besten Standards und die besten Softwarelösungen auf Netzeffektmärkten durchsetzen und speziell kleine Softwarefirmen und Startup-Unternehmen einen schweren Stand haben (Abschn. 2.2.2). In diesem Zusammenhang ziehen wir zum einen den so genannten Pinguineffekt als Erklärungsmodell heran. Zum anderen wird ein Modell von Arthur (1989) vorgestellt, das aufzeigt, wie kleine zufällige Ereignisse den Erfolg von Standards und Softwarelösungen auf Netzeffektmärkten bestimmen können. Im nächsten Schritt untersuchen wir, warum auf Netzeffektmärkten häufig die „Winner-takes-it-all-Regel“ gilt, was letztlich die Begründung für eine Vielzahl von Unternehmensübernahmen in der Softwarebranche darstellt (Abschn. 2.2.3). Darauf aufbauend wollen wir zeigen, wie Softwareanbieter die Existenz von Netzeffekten als Wettbewerbsfaktor ausnutzen können (Abschn. 2.2.4). Danach beschreiben wir in Abschn. 2.2.5 mit digitalen Spielen ein Anwendungsbeispiel, mit dem sich sowohl Plattformstrategien als auch zweiseitige Netzeffekte erklären lassen. Der Abschnitt schließt mit einigen Überlegungen zu den Grenzen und Erweiterungsmöglichkeiten der Netzeffekttheorie (Abschn. 2.2.6).

2.2.1 Netzeffekte: Grundlagen und Definitionen

Michael Katz und Carl Shapiro definieren das Konzept der Netzeffekte folgendermaßen: „The utility that a given user derives from the good depends upon the number of other users who are in the same network as he or she“ (Katz und Shapiro 1985, S. 424). Netzeffekte liegen also vor, wenn sich der Nutzen eines Gutes für einen Konsumenten dadurch erhöht, dass andere Konsumenten das Gut ebenfalls nutzen. Je größer das Netzwerk dabei ist, umso besser ist dies in der Regel für die Anwender. Dabei wird zwischen direkten und indirekten Netzeffekten unterschieden.

Direkte Netzeffekte entstehen, da die Anwender durch die gemeinsame Nutzung von Softwarestandards oder allgemeiner Technologien einfacher und damit kostengünstiger miteinander kommunizieren können. Das klassische Beispiel für direkte Netzeffekte ist das Telefon: Je mehr Personen ein Telefon besitzen, umso vorteilhafter ist diese Technologie für die Nutzer. Das gleiche Prinzip gilt etwa auch für XML-Vokabulare, wie xCBL, deren Nutzung ebenfalls umso vorteilhafter ist, je mehr andere Anwender das betreffende Vokabular nutzen. Auch im Bereich von ERP-Systemen spielen Netzeffekte auf zwischenbetrieblicher Ebene eine zunehmend wichtige Rolle. Durch die Nutzung standardisierter Formate wird beispielsweise der Austausch von Geschäftsdokumenten zwischen verschiedenen ERP-Systemen vereinfacht. Aus diesem Grund üben etwa in der Automobilindustrie die starken Partner in der Wertschöpfungskette häufig Druck auf kleinere Unternehmen aus, ein kompatibles und oftmals sogar identisches ERP-System zu verwenden. Ein weitergehender Schritt ist eine Prozessstandardisierung über Unternehmensgrenzen hinweg.

Indirekte Netzeffekte resultieren demgegenüber aus der Abhängigkeit zwischen dem Konsum eines Basisgutes und dem Konsum komplementärer Güter und Dienstleistungen.

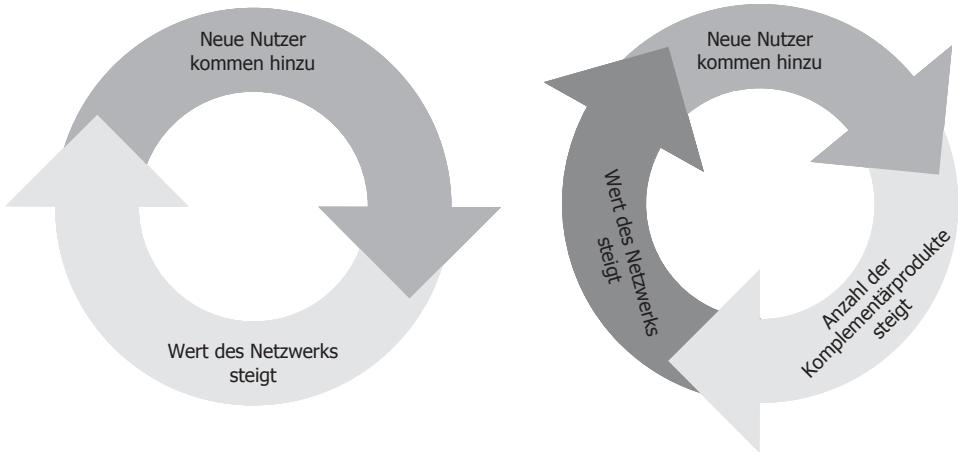


Abb. 2.1 Der Kreislauf der positiven Feedbacks – direkt und indirekt. (modifiziert nach Bansler und Havn 2002, S. 819)

Sie treten also dann auf, wenn die größere Verbreitung eines Gutes ein größeres Angebot an Zusatzgütern und Diensten nach sich zieht und damit wiederum der Nutzen des Basisgutes erhöht wird. Solche indirekten Netzeffekte treten beispielsweise auf bei Standardsoftware sowie komplementären Beratungsleistungen, bei Betriebssystemen mit dazu passender Anwendungssoftware oder wenn wir an die Verfügbarkeit von Experten und/oder Tools für Programmiersprachen denken.

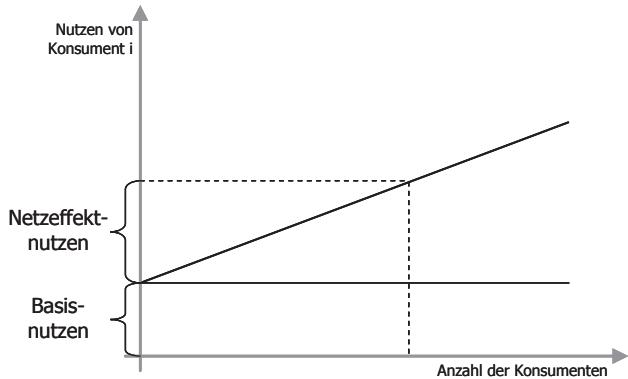
Netzeffekte führen zu nachfrageseitigen Skaleneffekten und zu so genannten positiven Feedbacks bzw. Increasing Returns. Shapiro und Varian fassen es wie folgt zusammen: „Positive Feedback makes the strong get stronger and the weak get weaker“ (Shapiro und Varian 1998, S. 175). Dieser sich positiv verstärkende Kreislauf ist in der folgenden Abb. 2.1 sowohl für direkte als auch für indirekte Netzeffekte dargestellt.

Diese Netzeffekte sind ein starkes Argument für Anwender, auf Softwareprodukte mit hoher Verbreitung zu setzen bzw. sich für Anbieter zu entscheiden, die ihnen starke Netzeffekte anbieten können.

Darüber hinaus können weitere Punkte für die Wahl eines großen Anbieters sprechen: An erster Stelle ist hier die Investitionssicherheit zu nennen, die insbesondere vor dem Hintergrund hoher Switching Costs (siehe hierzu auch Abschn. 2.2.4) eine zentrale Rolle spielt. Zudem gilt wohl auch, dass niemand seinen Job verlieren wird, wenn er bei der Auswahl der Softwarelösungen auf den Marktführer, etwa SAP oder Microsoft, setzt. Das Scheitern der Einführung einer Open Source Software dürfte demgegenüber für einen Entscheidungsträger wesentlich schwieriger zu begründen sein und eher zu beruflich negativen Konsequenzen führen.

Von diesem Netzeffektnutzen wollen wir im Weiteren den Basisnutzen unterscheiden, den eine Software stiften kann. Dabei handelt es sich um den Nutzen, den eine Software unabhängig von der Nutzung durch andere Anwender bietet. Ein Beispiel für ein Gut, das

Abb. 2.2 Der Zusammenhang von Basis- und Netzeffektnutzen



sowohl einen Netzeffekt- als auch einen Basisnutzen stiftet, ist etwa ein Tabellenkalkulationsprogramm. Der Basisnutzen resultiert aus den angebotenen Funktionalitäten, der Netzeffektnutzen ergibt sich demgegenüber aus den Möglichkeiten, mit anderen Anwendern Dateien auszutauschen (direkter Netzeffekt) oder sich Tipps für den Umgang mit der Software zu holen (indirekter Netzeffekt). Dagegen ist ein E-Mail-System ein Beispiel für eine Software, die ausschließlich einen Netzeffektnutzen stiftet, denn was kann man schon als einziger Nutzer mit einem solchen System anfangen?

Abbildung 2.2 zeigt die Nutzenfunktion eines Anwenders. Dabei ergibt sich der Gesamtnutzen aus der Summe von Basis- und Netzeffektnutzen. In dieser Darstellung wird ein linearer positiver Zusammenhang zwischen dem Netzeffektnutzen und der Anzahl der Anwender angenommen.

Zur Messung der Stärke des Netzeffektnutzens im Vergleich zum Basisnutzen führen wir einen Netzeffektfaktor Q ein. Dabei bezeichnet im Weiteren c den Netzeffektnutzen und b den Basisnutzen einer Software. Den Netzeffektfaktor definieren wir nun wie folgt:

$$Q = \frac{c}{c + b}$$

Damit ist Q zwischen Null und Eins normiert. Je höher der Wert von Q ist, umso größer ist die Rolle, die Netzeffekte im Vergleich zum Basisnutzen spielen. Beispielsweise würde der Netzeffektfaktor einer Standardsoftware wie Microsoft Excel einen Wert zwischen Null und Eins annehmen. Ein EDI-Standard hingegen stiftet keinen Basisnutzen; insofern würde der Netzeffektfaktor in diesem Fall genau Eins werden. Wir werden auf diesen Netzeffektfaktor im Zusammenhang mit der Bewertung von Preisstrategien zurückkommen (siehe hierzu Abschn. 3.3 dieses Buches).

Im Folgenden wollen wir nun untersuchen, welche Auswirkungen die Existenz von Netzeffekten auf Softwaremärkte hat.

2.2.2 Auswirkungen von Netzeffekten auf Softwaremärkte

Beginnen wir mit einem einfachen Beispiel: Angenommen, einer neu gegründeten Firma würde es gelingen, eine Software zu entwickeln, die bessere Funktionalitäten hat als das Office-Paket von Microsoft. Die Software bietet den Kunden also einen höheren Basisnutzen. Würde sich dieses Startup-Unternehmen im Markt etablieren können? Vermutlich würden wir nicht viel Geld auf den Erfolg dieser Firma wetten. Die Schwierigkeit besteht für das kleine Unternehmen darin, dass der Marktführer – in diesem Fall Microsoft – seinen Kunden bereits den Vorteil hoher Netzeffekte bieten kann.

Aus gesamtwirtschaftlicher Sicht bestünde die optimale Lösung – unter der Bedingung, dass die Kosten für die Umstellung nicht zu hoch sind – darin, dass sich alle Anwender für die neue Software entscheiden würden. Denn sie stiftet, wie dargestellt, den höheren Basisnutzen und wenn alle Anwender einen Wechsel vornähmen, würden sich nach einer gewissen Zeit auch die entsprechenden Netzeffekte einstellen. Dennoch ist es unwahrscheinlich, dass ein solcher Wechsel tatsächlich stattfinden wird, da die Existenz dieser Netzeffekte zu einem „Lock-in“ in eine aus technischer Sicht nicht optimale Technologie führen kann (siehe hierzu auch David (1985) sowie Liebowitz und Margolis (1994)). Damit hat ein Unternehmen, das auf einem Netzeffektmarkt über eine installierte Basis verfügt, einen erheblichen Wettbewerbsvorteil, der von Konkurrenten nur schwer aufgeholt werden kann. In unserem Beispiel ist es der Microsoft-Konzern, der diesen Wettbewerbsvorteil hat.

2.2.2.1 Der Pinguineffekt

Dass die installierte Basis in vielen Fällen den Wechsel zu einem technisch überlegenen Standard verhindert, ist nach Farrell und Saloner auf Informations- und daraus resultierende Abstimmungsprobleme zurückzuführen (Farrell und Saloner 1985). Sie argumentieren wie folgt: Die Summe der Nutzen aller Marktteilnehmer könnte erhöht werden, wenn diese sich für einen Übergang zu dem neuen (technisch überlegenen) Standard entschieden – in unserem Beispiel für die Programme der neu gegründeten Softwarefirma. Die Anwender sind jedoch unsicher, ob ein solcher Übergang tatsächlich stattfindet. Das Problem besteht für einen potenziellen Umsteiger bei unvollkommener Information darin, dass die anderen Marktteilnehmer möglicherweise nicht folgen und der Zugewinn an Basisnutzen durch die Anwendung des neuen Standards den entgangenen Netzeffektnutzen nicht kompensieren kann. Die Unsicherheit bezüglich der Reaktion der anderen Marktteilnehmer kann dazu führen, dass die Unternehmen in ihrem bisherigen Zustand verharren. Dieses Abstimmungsproblem wird auch als Pinguineffekt bezeichnet, wobei der Namensgebung folgende nette Analogie zugrunde liegt: Hungrige Pinguine stehen am Rande einer Eisscholle. Aus Angst vor Raubfischen hoffen sie, dass andere Pinguine zuerst ins Wasser springen, um das damit verbundene Risiko – nämlich Opfer eines Raubfisches zu werden – auszuloten. Sobald einige Pinguine den Sprung gewagt haben, hat sich die Gefahr für die anderen Pinguine verringert und die „Trittbrettpinguine“ folgen nach (Farrell und Saloner 1987).

Aufgrund dieses Pinguineffektes sehen sich insbesondere Softwareunternehmen einem Startup-Problem ausgesetzt. Auch wenn natürlich jede Firma, die neu in einen Markt einsteigt, mit Schwierigkeiten zu kämpfen hat, gilt dies für Unternehmen auf Netzeffektmärkten in besonderer Weise. Das Startup-Unternehmen steht hier nicht nur vor der Aufgabe, das Produkt selbst überzeugend anzupreisen, sondern auch glaubhaft zu versichern, dass es sich flächendeckend durchsetzen und Netzeffekte generieren wird. Die Auswirkung dieses Startup-Problems aufgrund des Pinguineffekts wird in der angelsächsischen Literatur auch mit *excess inertia* (d. h. eine Unterversorgung mit Standards) bezeichnet. Daneben kann es grundsätzlich auch zu einer Überversorgung (*excess momentum*) mit Standards kommen.

Dieser Pinguineffekt ist ein wesentlicher Grund dafür, dass sich in vielen Fällen ein aus technischer Sicht nicht-optimaler Standard durchgesetzt hat; hierzu gibt es eine Vielzahl von Beispielen:

- Der Videostandard VHS war lange Zeit marktbeherrschend, obwohl Betamax aus technischer Sicht besser einzuschätzen war.
- OSI-Protokolle konnten sich trotz hoher technischer Qualität nicht flächendeckend auf allen Ebenen durchsetzen. Stattdessen dominieren heute die Internetprotokolle den Markt.
- Der Tastaturanordnung im so genannten QWERTY-Design wurde nachgesagt, weniger effizient als andere – wie zum Beispiel die später nach ergonomischen Gesichtspunkten entwickelte Dvorak-Tastatur – zu sein. Eine Umstellung auf eine möglicherweise effizientere Tastenanordnung wird aufgrund der Existenz von sowohl direkten als auch indirekten Netzeffekten verhindert.

Wir lernen daraus, dass sobald sich ein Standard erst einmal durchgesetzt hat, er nicht mehr so leicht aus dem Markt gedrängt werden kann.

2.2.2.2 Diffusionsprozesse: Das Modell von Arthur

Welcher Standard sich letztlich durchsetzt, kann auch durch zufällige kleine Ereignisse beeinflusst werden (Arthur 1989). Der Begriff Pfadabhängigkeit beschreibt diese Auswirkungen von frühen und gegebenenfalls zufälligen Ereignissen im Diffusionsprozess auf die spätere Marktstruktur (David 1985, S. 322). Das im Folgenden dargestellte Modell zeigt auch anschaulich, wie der Wettbewerb zwischen zwei Technologien stattfindet, die beide Netzeffekte erzeugen.

Als Ausgangspunkt dienen uns hierbei zwei Technologien A und B. Darüber hinaus existieren zwei Anwendertypen, die Arthur als R-Akteure bzw. S-Akteure bezeichnet. Der Gesamtnutzen einer Technologie ergibt sich dabei aus der Summe des Basis- und des Netzeffektnutzens. Der Basisnutzen wird in dem Modell durch die folgenden Parameter abgebildet:

- a_r Basisnutzen der Technologie A für R-Akteure
- a_s Basisnutzen der Technologie A für S-Akteure
- b_r Basisnutzen der Technologie B für R-Akteure
- b_s Basisnutzen der Technologie B für S-Akteure

Dabei soll gelten, dass die Technologie A für R-Akteure einen höheren Basisnutzen hat als die Technologie B ($a_r > b_r$). Umgekehrt stiftet Technologie B für die S-Akteure einen höheren Basisnutzen als Technologie A ($a_s < b_s$).

Der Netzeffektnutzen wächst, wie in Abb. 2.2 bereits dargestellt, annahmegemäß linear mit der Anzahl der Nutzer. Dabei bezeichnet n_a die Anzahl der Nutzer von Technologie A und n_b entsprechend die Anzahl der Anwender von Technologie B.

Arthur zeigt nun in seinem Modell auf der Basis von Entscheidungen der Akteure, wie sich eine Technologie am Markt durchsetzt. Dazu entwickelt er ein Simulationsmodell, in dem sich zufällig ein R- oder ein S-Akteur zwischen den beiden Technologien entscheidet. Dabei wählen die Akteure die Technologie aus, die ihnen den höchsten Gesamtnutzen stiftet. Der Gesamtnutzen der Technologie A beträgt für die R-Akteure ($a_r + r \times n_a$), der entsprechende Gesamtnutzen von Technologie B ($b_r + r \times n_b$). Der Parameter r steht hierbei für den Grenznutzen jedes weiteren Anwenders. Die S-Agenten entscheiden annahmegemäß nach dem gleichen Kalkül. Abbildung 2.3 zeigt die Ergebnisse der Anwendung des Simulationsmodells.

In dem in der Abbildung weiß dargestellten Bereich wählen R-Akteure Technologie A und S-Akteure Technologie B, d. h. jeweils die Technologie, die ihnen den höchsten Basisnutzen stiftet. Wenn sich jedoch beispielsweise viele S-Agenten für Technologie B entscheiden, so steigt entsprechend der Netzeffektnutzen von Technologie B an. Dies führt

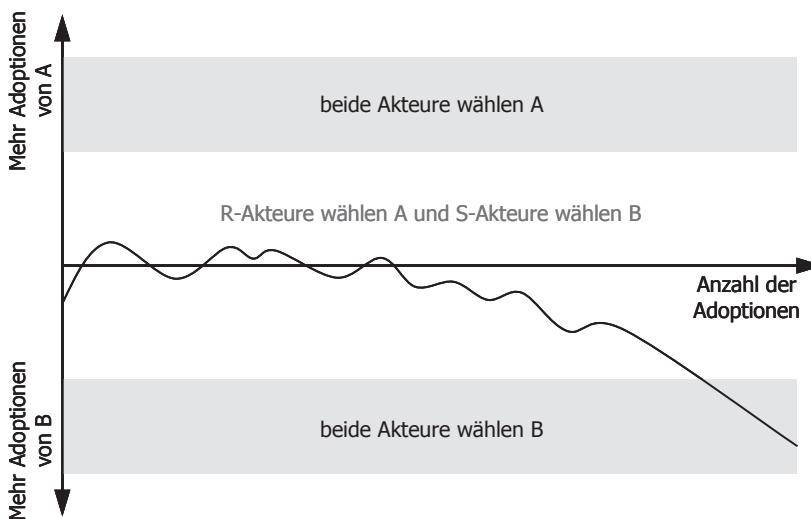


Abb. 2.3 Pfadabhängigkeit aufgrund von „Increasing Returns“. (Arthur 1989, S. 120)

ab einer gewissen Nutzeranzahl dazu, dass auch R-Agenten sich nun für die Technologie B entscheiden, da die höheren Netzeffekte dieser Technologie den für die R-Akteure geringeren Basisnutzen übercompensieren.

Mit Hilfe dieses einfachen Modells lässt sich anschaulich zeigen, dass zum einen den so genannten „Early Adopters“ eine entscheidende Rolle bei dem Kampf um die Marktanteile zukommt. Zum anderen wird klar, dass es häufig zufällige Ereignisse sind, die die Verbreitung einer Technologie auf Netzeffektmärkten bestimmen.

2.2.3 Struktur von Softwaremärkten

Seit langem haben Ökonomen erkannt, dass Netzeffekte oftmals zu Monopolen führen. Daher werden Märkte, die starke positive Netzeffekte und Feedbacks aufweisen, häufig als „tippy“ bezeichnet: „When two or more firms compete for a market where there is strong positive feedback, only one may emerge as a winner. It's unlikely that all will survive.“ (Shapiro und Varian 1998, S. 176). Verschiedene Standards oder Technologien existieren häufig also nicht nebeneinander, sondern es setzt sich ein dominanter Standard bzw. eine dominante Technologie durch (Besen und Farrell 1994, S. 118). Es handelt sich deshalb um so genannte „Winner-takes-it-all-Märkte“.

Diese theoretischen Überlegungen werden durch Beobachtungen heutiger Softwaremärkte gestützt. Betrachten wir z. B. Märkte für Standardsoftwarelösungen: Hier ist es auffällig, dass die Anzahl der Anbieter deutlich zurückgegangen ist. So gab es vor einigen Jahren beispielsweise im Bereich von Browsern oder Office-Lösungen noch Alternativen zu Microsoftprodukten, wie etwa den Netscape Navigator, Word Perfect im Bereich Textverarbeitung oder 1-2-3 zur Tabellenkalkulation. Mittlerweile existiert auf diesen Märkten im Wesentlichen nur noch eine sehr starke Konkurrenz für Microsoft: die Open-Source-Gemeinde.

Der Browerkrieg

Der so genannte Browerkrieg entwickelte sich 1995 zwischen Microsoft und der Firma Netscape. Netscape ist mit seinem Internet Browser Netscape Navigator bekannt geworden und besaß vor 1995 einen Marktanteil von über 80 %. Als Microsoft schließlich die wachsende Bedeutung des Internets erkannte und daraufhin sein Konkurrenzprodukt, den Internet Explorer (IE), entwickelte, begann der „Krieg“ zwischen den beiden Firmen. In dessen Verlauf gab es zwei besonders interessante Aspekte, die im Folgenden näher beschrieben werden: zum einen die Bedeutung von Open-Source-Produkten als Konkurrenz zu kommerzieller Software, zum anderen die Auswirkungen von Produktbündelungen für ein Unternehmen.

Phase I: Die Anfänge des WWW

Einer der wichtigsten Internetbroweser, der zu Beginn der rasanten Entwicklung des Internets Anfang der 90er Jahre auf den Markt kam, war MOSAIC. Er besaß als erster

Browser eine grafische Oberfläche, mit der es möglich war, im Internet zu „surfen“. Ende 1993 wurde dann die Firma Netscape gegründet, die bereits 1994 die erste Version des Browsers Netscape Navigator auf den Markt brachte. Bis zum Jahre 1995 hatte Netscape auf dem Browsermarkt schließlich eine Monopolstellung aufgebaut.

Phase II: 1995–1998

Als Microsoft entschied, in den Internetmarkt einzusteigen, wurde die erste Version des Internet Explorers entwickelt. Das Unternehmen hatte dabei zwei entscheidende Vorteile gegenüber Netscape: Zum einen standen Microsoft wesentlich mehr finanzielle Mittel für die Entwicklung der Browsersoftware zur Verfügung. Zum anderen hatte Microsoft die Möglichkeit der Produktbündelung und begann, den Internet Explorer in seine Programme einzubinden. Ab Windows 98 war der Internet Explorer dann fest in das Betriebssystem integriert (nach dem Motto: „Was einmal installiert ist, benutzen die Leute auch“). Nachdem Windows auf ca. 95 % aller neu verkauften PCs installiert war, zahlte sich die Bündelungsstrategie für Microsoft auch tatsächlich bald aus. Die Marktanteile des Internet Explorers stiegen in der Folgezeit von ursprünglich weniger als 3 % auf über 95 %. Schließlich musste sich Netscape geschlagen geben, als sein Marktanteil 1998 auf unter 4 % gesunken war. Netscape veröffentlichte daraufhin den Quellcode des Navigators und machte ihn damit zu dem Open-Source-Projekt „Mozilla“.

Phase III: 2004 bis heute

Diese Phase wird auch als die zweite Runde des Browserkrieges bezeichnet. 2004 wurden zunehmend Sicherheitslücken des Internet Explorers bekannt, was Mozilla die Chance bot, wieder Marktanteile zu gewinnen. Die Mozilla Community veröffentlichte im November 2004 den Firefox 1.0 (ein abgespeckter Browser). Die Gründe für die daraufhin sehr schnelle Verbreitung des Firefox liegen nicht nur in den Sicherheitsmängeln des Internet Explorers. Der Firefox bietet vielmehr auch eine Reihe praktischer Funktionalitäten (Tabbed Browsing, Find-as-you-type, Download Manager etc.), die von den Open-Source-Entwicklern integriert wurden. Microsoft konnte die veraltete Technologie seines Internet Explorers 6.0 nicht schnell genug aktualisieren, um zu verhindern, dass der moderne Open Source Browser Firefox ihm wichtige Marktanteile streitig machte. Erst mit seiner Version 7.0 (in der viele der neuen Funktionen beim Firefox abgeschaut sind) konnte sich der Internet Explorer wieder mit einem positiven Image auf dem Markt platzieren. In 2013 schneidet der Internet Explorer mit einem weltweiten Marktanteil von 30,7 % besser ab als Firefox mit 21,2 %.

Quellen: heise-online (2002): *Nicht Trustworthy – IE gefährdet Rechner und Netz.* <http://www.heise.de/ct/02/25/100/>; Netplanet: *Die Geschichte des Webbrowser.* <http://www.netplanet.org/www/browser.shtml>; heise-online (2013): *Marktanteile: Chrome und Safari legen zu, Internet Explorer und Firefox verlieren.*

<http://www.heise.de/newsticker/meldung/Marktanteile-Chrome-und-Safari-legen-zu-Internet-Explorer-und-Firefox-verlieren-1775342.html>

Eine monopolartige Struktur, wie sie etwa auf dem Markt für Office-Software existiert, ist für den Anwender häufig problematisch. Dabei steht Microsoft keineswegs im Kreuzfeuer der Kritik, weil das Unternehmen – wie entsprechend der traditionellen Mikroökonomie anzunehmen wäre – einen (höheren) so genannten Cournot'schen Preis durchsetzen wird. Vielmehr hat die Europäische Kommission gegen Microsoft zahlreiche Bußgelder wegen Verletzung von Wettbewerbsgesetzen verhängt und dem Konzern unter Androhung weiterer Strafen Auflagen in Bezug auf die Offenlegung von Schnittstellenspezifikationen und die Entbündelung von Produkten gemacht. Beide Forderungen sind insbesondere im Sinne von anderen Softwareanbietern, um es diesen leichter zu machen, kompatible Produkte zum Microsoft-Standard zu entwickeln und anzubieten.

2.2.4 Netzeffekte als Wettbewerbsfaktor

Unsere bisherigen Ausführungen legen nahe, dass Netzeffekte für einen Anbieter ein entscheidender Wettbewerbsvorteil sein können. Ist eine Softwarelösung erst einmal weit verbreitet, führt dies zu einem Lock-in-Effekt. Potenzielle Konkurrenten stehen deshalb vor einem Startup-Problem, das umso größer ist, je höher der Netzeffektfaktor ist.

Dabei ist bei der Analyse der Wettbewerbssituation zu beachten, dass für einen Anwender der Wechsel seiner Softwarelösungen in der Regel mit hohen Switching Costs verbunden ist. Dies gilt insbesondere etwa für ERP-Systeme, weshalb ein Wechsel in der Praxis relativ selten zu beobachten ist. Doch was ist der Grund für diese hohen Switching Costs? Gerade auf dem ERP-Markt gilt, dass diese Software auch die Geschäftsprozesse der Anwender abbildet und gestaltet. Ein Wechsel des Anbieters würde deshalb auch erhebliche Kosten für die Organisationsänderungen nach sich ziehen. Insbesondere bei Softwaresstandards mit einem hohen Netzeffektfaktor besteht zudem die Unsicherheit, inwieweit auch andere Anwender zu einem anderen neuen Standard wechseln – wir erinnern uns an den in Abschn. 2.2.2.1 angesprochenen Pinguineffekt. Dabei steigen die Switching Costs tendenziell sogar, je länger eine ERP-Software im Einsatz ist, da sie damit auch immer besser in das Nutzungsumfeld integriert wird. Umfangreiche individuelle Anpassungen des Systems erhöhen die Switching Costs zusätzlich.

Wie kann ein Softwareanbieter trotz der Existenz eines solchen Lock-in-Effekts Anwender zu einem Systemwechsel bewegen? Eine Möglichkeit besteht darin, den Wechsel der Anwender zu subventionieren. So hat beispielsweise die SAP ein „Safe Passage Program“ eingeführt, das den Nutzern einen Anreiz bietet, etwa von Siebel auf SAP-Lösungen umzusteigen. SAP-Neukunden können auf diese Weise ihre installierten Lösungen mehr oder weniger bilanz- bzw. kostenneutral in eine SAP-Implementierung transferieren. Hier übernimmt die SAP zum Teil interne und externe Projektkosten, Lizenzkosten sowie möglicherweise Teile der Hardwareanschaffungen bzw. Schulungsaufwände. Das Instrument der Safe-Passage-Initiative kann im Einsatz gegen direkte Wettbewerber sehr effektiv sein, da der Kunde die Chance erhält, nicht gewünschte bzw. ungeliebte Softwareinstallationen abzulösen und die Entscheidungsbarriere zu diesem Schritt deutlich reduziert ist.

Auf verschiedene alternative Möglichkeiten für die Softwareanbieter, die Anwender mit Niedrigpreisstrategien zu locken, gehen wir in Abschn. 3.3 ein.

Insbesondere für große Anbieter, die ihren Kunden bereits Netzeffekte anbieten können, kann es zur Erhaltung der Wettbewerbsvorteile durchaus lukrativ sein, die eigenen Schnittstellenspezifikationen nicht vollständig offen zu legen. Auf diese Weise haben es andere Anbieter schwerer, kompatible Produkte anzubieten. Es ist jedoch zu beachten, dass diese Strategie der teilweisen Inkompatibilisierung für Anbieter in der heutigen Zeit nur noch schwer gegenüber den Kunden durchsetzbar ist. Die Anwender haben längst erkannt, dass offene Standards die Chance auf eine größere Herstellerunabhängigkeit eröffnen und vor allem auch die IT-Integration auf inner- sowie zwischenbetrieblicher Ebene erleichtern können.

Völlig anders sieht die Entscheidung in Bezug auf die Kompatibilisierung der Produkte aus der Perspektive von kleineren Softwarehäusern oder von Anbietern aus, die mit ihren Lösungen noch nicht im Markt sind. Diese Softwareanbieter haben gar keine Wahl als entweder vollständig auf offene Standards zu setzen oder zu den großen Anbietern kompatible Produkte zu entwickeln. Ein anschauliches Beispiel bietet der Fall von Business Objects, einem Anbieter auf dem Markt für Business-Intelligence-Software:

Business Objects wurde 1990 von zwei Oracle-Angestellten gegründet. Das Unternehmen begann mit einer Anwendung, die Oracle-Kunden ohne SQL-Kenntnisse das intuitive Erstellen von Datenbankabfragen ermöglichte. In den ersten Jahren konzentrierte sich die Firma vollständig auf Oracle-Kunden; die eigene Anwendung unterstützte nur Oracle-Datenbanken. Damit wurde eine Nischenstrategie verfolgt und andere Datenbanken wurden vorerst außen vor gelassen.

Durch den Fokus auf Oracle-Kunden konnte Business Objects mit Oracle eine enge Partnerschaft eingehen. Business-Objects-Produkte wurden von Oracle als Komplementärprodukte betrachtet; sie waren keine Konkurrenz, sondern unterstützten den Absatz der eigenen Datenbankanwendung. Später ging Business Objects weitere Partnerschaften ein, etwa mit IBM, und wurde zu einem bekannten Business-Intelligence-Anbieter. 2007 wurde das Unternehmen von der SAP übernommen.

Das Oracle-Datenbanksystem bildete also zunächst die Plattform für die von Business Objects angebotenen Produkte bzw. Services. Dieses Modell ist auch in vielen anderen Bereichen der Softwareindustrie zu finden, etwa im Bereich der Spielesoftware, auf den wir nachfolgend eingehen.

2.2.5 Ein Anwendungsbeispiel: Zweiseitige Netzeffekte und Plattformstrategien in der digitalen Spieleindustrie

In diesem Abschnitt wollen wir eine Industrie vorstellen, die zwischen Software- und Medienbranche angesiedelt ist: die digitale Spieleindustrie. An diesem Markt sollen insbesondere die Auswirkungen zweiseitiger Netzeffekte sowie die Bedeutung von Plattformstrategien aufgezeigt werden. Wir konzentrieren uns dabei auf Konsolenspiele, da

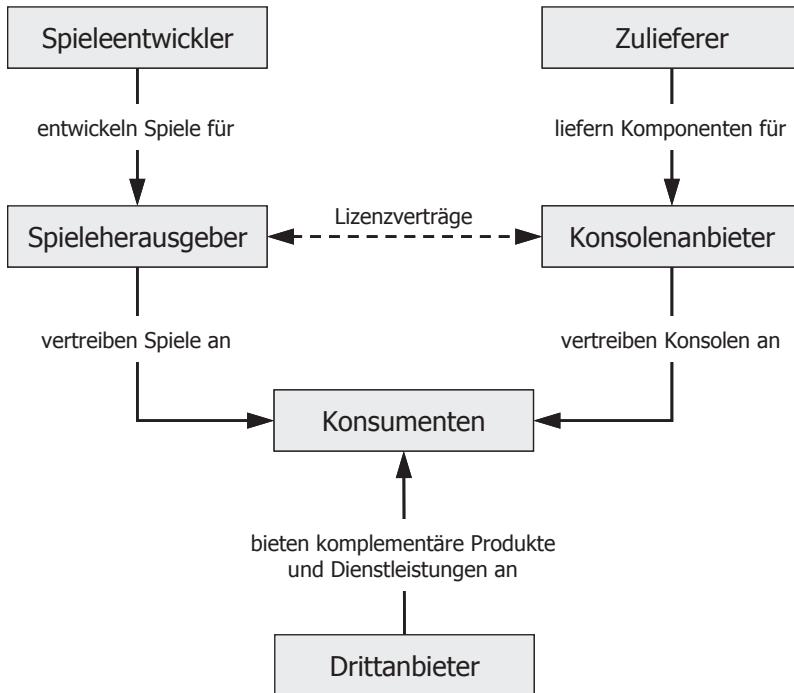


Abb. 2.4 Wertschöpfungsstruktur in der digitalen Spieleindustrie

sich an diesem Fall die vielfältigen Wirkungen von Netzeffekten besonders gut zeigen lassen.

2.2.5.1 Die digitale Spieleindustrie – Ein Überblick

Der Bereich der digitalen Spiele hat sich inzwischen zu einem beachtlichen Markt der Unterhaltungsindustrie entwickelt. So beläuft sich der weltweite Umsatz auf circa 79 Mrd. \$ in 2012 mit steigender Tendenz (Gartner 2013a). Dabei kommen die größten Softwareanbieter aus Japan und USA (Statista 2014a). In den USA scheint darüber hinaus eine ungebrochen hohe Nachfrage nach Computerspielen zu existieren, so spielten laut einer Untersuchung der Entertainment Software Association in 2013 rund 58 % der US-Amerikaner Videospiele (ESA 2013). Auch in Deutschland hat man das Potenzial der digitalen Spieleindustrie erkannt, nach UK ist Deutschland inzwischen der wichtigste Standort in Europa für die Entwicklung von digitalen Spielen (Martin-Jung 2012).

Zunächst wollen wir die Akteure und Wertschöpfungskette der digitalen Spieleindustrie betrachten. Bei den Akteuren handelt es sich um Konsolenanbieter, Zulieferer, Spieleentwickler, Spieleherausgeber, Konsumenten und Drittanbieter. Die Wertschöpfungsstruktur der digitalen Spieleindustrie ist in Abb. 2.4 dargestellt.

Konsolenanbieter entwickeln und vertreiben Konsole, die die Grundlage für die Nutzung von digitalen Spielen darstellen. Wir werden diese Konsole daher im Weiteren auch als Plattformen bezeichnen. Die derzeit dominierenden Konsolenanbieter sind Microsoft,

Nintendo und Sony. Sie werden von Zulieferern unterstützt, die bestimmte Komponenten für die Fertigung der Konsolen liefern; so werden beispielsweise spezialisierte Chips für Sound und Grafik, aber auch CPUs von Herstellern wie nVidia, AMD, IBM und Intel produziert.

Die Spieleherausgeber, wie z. B. Electronic Arts oder Activision, vervielfältigen und vermarkten Spiele für diese Konsolen. Sie nehmen dabei die Rolle eines Intermediärs zwischen den Spieleentwicklern und den Konsolenanbietern ein.

Entwickelt werden diese Spiele von internen oder eigenständigen Spieleentwicklern bzw. Entwicklungsstudios, wie z. B. Pyro Studios, Rockstar North oder Deck13. Die Entwickler werden dabei entweder vom Spieleherausgeber mit der Programmierung eines bestimmten Spiels beauftragt und damit auch finanziert oder sie treten mit einem Prototyp an die Spieleherausgeber heran. Eine wesentliche Besonderheit dieses Marktes besteht darin, dass Spieleherausgeber eine entsprechende Lizenz vom Konsolenhersteller benötigen, um ein Spiel für eine bestimmte Konsole auf den Markt bringen zu dürfen. Das Modell sieht vor, dass der Konsolenanbieter einen Anteil der Umsatzerlöse der Spieleherausgeber erhält und damit am Erfolg des Spiels beteiligt wird.

Drittanbieter (z. B. Hersteller von Peripheriegeräten, Zeitschriftenverlage oder Videotheken) bieten für Konsolen komplementäre Produkte und Dienstleistungen an, beispielsweise Spieldomänen oder Joysticks und Speicherkarten. Die Konsumenten erwerben über entsprechende Distributionskanäle (Kaufhäuser oder Internethändler) Konsolen, Spiele und andere komplementäre Produkte und Dienstleistungen.

2.2.5.2 Zweiseitige Netzeffekte auf digitalen Spieldomänen

Auch für Plattform- bzw. Konsolenanbieter gilt häufig das zuvor dargestellte Winner-takes-it-all-Prinzip. So hat auch auf diesem Markt eine Konsolidierung stattgefunden. Während es bei den früheren Generationen noch mehr Konsolenanbieter gab, hat sich die Anzahl dieser Anbieter auf die drei Player Microsoft, Nintendo und Sony reduziert. Der Markt ist stark umkämpft, was häufig auch als „Konsolenkrieg“ bezeichnet wird. So bringen die Hersteller in der Regel ca. alle fünf Jahre eine neue Gerätegeneration auf den Markt. Abbildung 2.5 zeigt die wichtigsten Konsolen im europäischen Markt im Überblick. Einige Konsolen – wie Ataris Jaguar oder Segas Dreamcast – konnten sich trotz ihrer technischen Überlegenheit nicht am Markt behaupten. Andere – wie Nintendo NES oder Atari 2600 (von 1977–1986) – konnten demgegenüber jahrelang gegen ihre Konkurrenten bestehen, obwohl neuartige weiterentwickelte Konsolen in den Markt drängten. Als Begründung für die Marktdominanz von technisch unterlegenen Systemen kann die bereits vorgestellte Netzeffekttheorie herangezogen werden.

Der Erfolg einer Konsole bzw. Plattform hängt nicht nur von der technischen Leistungsfähigkeit und dem Preis, sondern auch von der verfügbaren Spieldaten ab. Es handelt sich also um einen Markt mit indirekten Netzeffekten, da der Nutzen und die Verbreitung einer Konsole maßgeblich von der Verfügbarkeit von attraktiven Spielen, bei denen es sich um Komplementärgüter handelt, abhängen (siehe hierzu auch Abschn. 2.2.1).

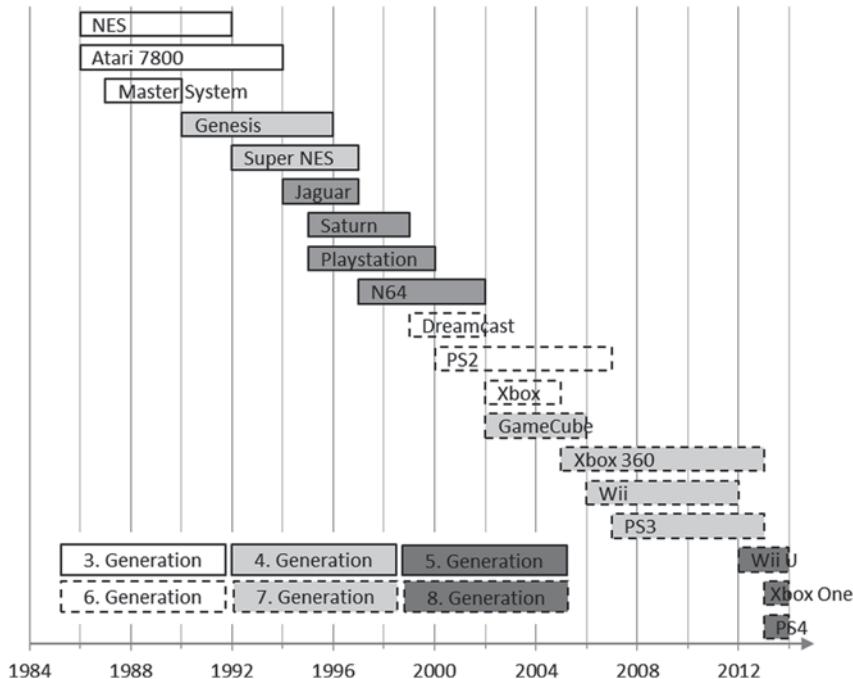


Abb. 2.5 Konsolengenerationen in Europa. (überarbeitet aus Wikipedia)

Am Beispiel der Wertschöpfungsstruktur der digitalen Spieleindustrie können wir aber auch eine weitere Form von Netzeffekten erläutern, die als zweiseitige Netzeffekte bezeichnet werden. Hierbei bilden sich um eine Plattform herum mehrere Gruppen. Dabei ist der Nutzen, den die Plattform der einen Gruppe stiftet, umso größer, je mehr Mitglieder die andere Gruppe hat. Armstrong nennt hierfür das folgende anschauliche Beispiel: „For instance, a heterosexual dating agency or nightclub can only do well if it succeeds in attracting business from both men and women“ (Armstrong 2006, S. 1). Übertragen auf den digitalen Spielemarkt handelt es sich bei den Gruppen um Spieleherausgeber und Konsumenten, die sich um die Plattform der Konsolen herum bilden. Dabei profitieren die Spieleherausgeber davon, wenn viele Konsumenten eine Konsole besitzen, weil dadurch für Spiele dieser Konsole die Nachfrage gesteigert wird. Umgekehrt profitieren die Konsumenten davon, wenn viele Spieleanbieter für eine Konsole Spiele entwickeln, da dadurch das Spieleangebot und damit der indirekte Netzeffektnutzen für die Konsole steigt.

Wir haben uns in diesem Abschnitt auf den Teilbereich der Konsolenspiele beschränkt, um das Konzept der zweiseitigen Netzeffekte an diesem Beispiel darstellen zu können. Darüber hinaus gibt es weitere Bereiche der digitalen Spieleindustrie, wie etwa Mobile Games, Browserspiele oder Massive Multiplayer Online Games, denen große Zukunftschancen prophezeit werden (Gartner 2013a).

Im folgenden Abschnitt wollen wir die Grenzen der Netzeffekttheorie sowie potenzielle Erweiterungsmöglichkeiten beschreiben.

2.2.6 Grenzen der Netzeffekttheorie

Auch wenn die Netzeffekttheorie einige Einsichten in Bezug auf die Erklärung der Verbreitung von Standards hervorgebracht hat, so ist ihre Anwendbarkeit dennoch begrenzt. Daher werden u. a. die nachfolgenden Erweiterungsmöglichkeiten vorgeschlagen (Weitzel et al. 2000):

Homogene Netzeffekte und Kosten der Netzgröße Eine wesentliche Erweiterungsmöglichkeit traditioneller Modelle der Netzeffekttheorie betrifft Annahmen über die Abhängigkeit des Netzeffektnutzens von der Anzahl der Anwender. Häufig wird hierbei, wie in den letzten Abschnitten, eine lineare Nutzenfunktion angenommen, d. h. jeder weitere Nutzer führt zu einer konstanten Nutzen erhöhung für die Anwender. Alternativ sind aber auch degressiv oder progressiv steigende Nutzenfunktionen denkbar. Im ersten Fall stiftet der n -te Nutzer einen geringeren Nutzen als der $n - 1$ -te Nutzer; im zweiten Fall gilt, dass der n -te Nutzer einen höheren Nutzen stiftet als der $n - 1$ -te Nutzer. Ebenso existieren in unterschiedlichen Modellen verschiedene Annahmen in Bezug auf die Kostenfunktionen in Abhängigkeit von der Netzgröße. Alternative Verläufe sind in Abb. 2.6 dargestellt.

Alle in der Abbildung dargestellten Nutzenfunktionen vernachlässigen jedoch – unabhängig vom Verlauf – die Individualität der Kommunikationsbeziehungen. Eine solche Individualisierung ist aus unserer Sicht jedoch essentiell, wenn über den Einsatz von Standards entschieden wird, wie etwa im Fall von EDI. Es ist klar, dass es beispielsweise für einen Automobilhersteller von hoher Relevanz ist, welche Standards die eigenen Zulieferer verwenden. Demgegenüber wird es dem Unternehmen relativ gleichgültig sein, welche Standards in Unternehmen verwendet werden, mit denen keine Geschäftsbeziehungen bestehen und auch zukünftig unwahrscheinlich sind.

Art des Netzeffektnutzens Der Netzeffektnutzen wird in den bisherigen Modellen überwiegend abstrakt behandelt und nicht spezifiziert. Damit ist das Problem verbunden, dass auf eine Konkretisierung des Nutzens verzichtet wird. Es stellt sich somit die Frage, worin denn konkret der Nutzen der entsprechenden Netzeffekte besteht.

Normative Implikationen Ein Großteil der Literatur betrachtet Standardisierungsentscheidungen aus volkswirtschaftlicher Sicht, insbesondere in Bezug auf die Konsequenzen der Existenz von Netzeffekten für die Markteffizienz. Auf der Anwenderseite verfolgen die meisten Beiträge das Ziel, Standardisierungsentscheidungen von Nutzern zu erklären. Sie verzichten jedoch weitgehend darauf, konkrete Handlungsempfehlungen zu geben.

An den genannten Grenzen der Netzeffekttheorie setzt das Standardisierungsproblem an. Erstens wird eine differenzierte, akteursspezifische Betrachtung von Kosten und Nutzen von Netzeffektgütern modelliert, zweitens wird der Netzeffektnutzen konkretisiert und drittens sollen Handlungsempfehlungen für Anwender gegeben werden, in welchem Maß Standardisierung für sie optimal ist.

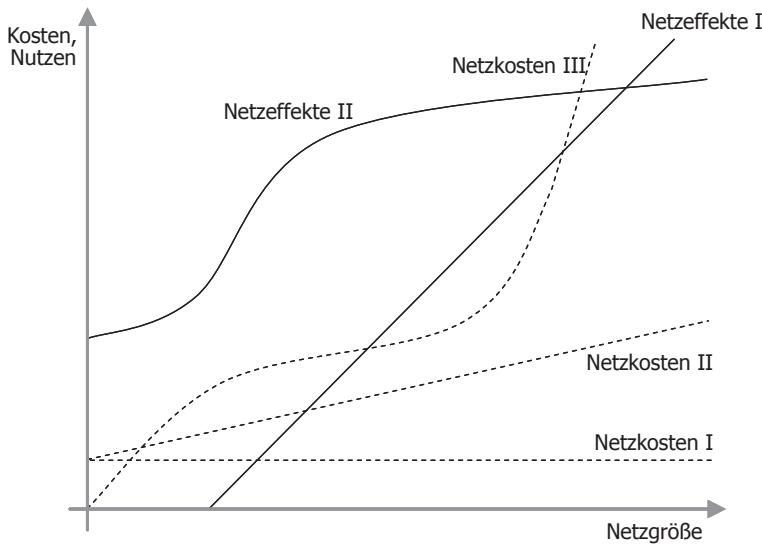


Abb. 2.6 Mögliche Kosten- und Nutzenverläufe in Abhängigkeit von der Netzgröße. (Weitzel 2004, S. 30)

2.3 Das Standardisierungsproblem

2.3.1 Ansatz und Hintergründe

In vielen Unternehmen oder Unternehmensverbünden sind die Anwendungssysteme historisch und zum Teil unkoordiniert gewachsen. Dies führt zu heterogenen IT-Landschaften, die den Informationsaustausch zwischen verschiedenen Bereichen durch Inkompatibilitäten erschweren. Häufig wird als Daumenregel davon ausgegangen, dass die Kosten für die Entwicklung und Wartung von Schnittstellen zwischen den inkompatiblen Teilsystemen bis zu 50% des gesamten IT-Budgets betragen.

Der Einsatz von Standards stellt eine wesentliche Maßnahme zur Reduzierung dieser Integrationskosten dar. Beispiele sind die unternehmensübergreifende Anwendung von EDI-Standards oder eine SOA-Plattform, die eine reibungslose Integration von Services unterschiedlicher Anbieter auf der Basis von Web-Service-Standards ermöglichen soll.

Eine Alternative zur Verwendung von Standards ist die Nutzung von Konvertern. Diese führt jedoch in der Regel zu höheren Kosten als die Verwendung von Standards, da die Integration von n Funktionen oder Bereichen zur Notwendigkeit des Einsatzes von bis zu $n \cdot (n - 1)$ Konvertern führt (Wüstner 2005). Zudem sinkt durch den flächendeckenden Einsatz solcher Konvertierungslösungen in der Regel die Flexibilität der gesamten IT-Landschaft. Wir werden diesen Weg daher bei unseren weiteren Überlegungen nicht näher betrachten.

Das im Folgenden dargestellte Modell untersucht die Entscheidung der Anwender über den Einsatz von Standards zur Koppelung von Applikationen. Die Antizipation dieser Ent-

scheidungen auf Modellbasis kann jedoch auch in Strategieempfehlungen für die Anbieter münden (Buxmann 2002). Dabei gehen wir, wie auch in dem Modell von Arthur, zunächst davon aus, dass ein Softwarestandard einem Anwender einen Basisnutzen und einen Netzeffektnutzen stiftet. Der Basisnutzen drückt etwa die Funktionalität eines ERP-Systems aus. Zum Vergleich der Basisnutzen mehrerer Softwarelösungen können grundsätzlich die bekannten Methoden zur Wirtschaftlichkeitsanalyse von Informations- und Kommunikationssystemen, wie etwa die Nutzwertanalyse oder die Kapitalwertmethode, verwendet werden. Wie kann nun der Netzeffektnutzen, der in der Netzeffekttheorie weitestgehend abstrakt behandelt wird, konkretisiert werden? Wir gehen im Weiteren davon aus, dass die gemeinsame Nutzung von Softwarestandards zu einer Einsparung von Informationskosten führt. Dabei bezeichnen Informationskosten alle Kosten, die für den Austausch von Informationen anfallen. So ermöglicht beispielsweise die Nutzung eines gemeinsamen EDI-Standards Kostenreduzierungen durch Einsparung von Personalkosten, Porto, Papier etc. Die gemeinsame Nutzung der gleichen Office-Lösung spart in der Regel Kosten, Zeit und auch Nerven beim Austausch von Dokumenten. Ebenso können Informationskosten eingespart werden, wenn Geschäftsprozesse über verschiedene Bereiche hinweg von einer Standardsoftware unterstützt werden.

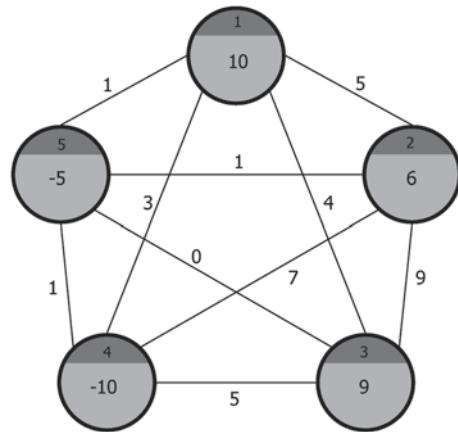
Auf der anderen Seite fallen jedoch Kosten für die Anschaffung und Implementierung der Softwarestandards an, die im Folgenden als Standardisierungskosten bezeichnet werden.

Wir vereinfachen die Überlegungen im Weiteren insofern, als wir von einer statischen Betrachtung ausgehen. Das bedeutet, dass in unserem Modell etwa die Kosten für die Einführung eines Softwarestandards einen bestimmten Wert annehmen. Das stellt natürlich eine Vereinfachung dar, weil die mit einer solchen Investition verbundenen Kosten über die Jahre in unterschiedlichen Höhen anfallen. So haben Anwender einmalige Lizenzkosten und zudem jährlich anfallende Wartungs- und Servicekosten zu tragen. Dennoch ist diese Vereinfachung für unsere Zwecke aus zwei Gründen unproblematisch: Zum einen können wir den Wert der Standardisierungskosten als abgezinster Wert der über die Jahre anfallenden Zahlungen interpretieren; zum anderen ändert auch eine dynamische Betrachtung, die explizit verschiedene Zeitpunkte betrachtet, nichts an den im Weiteren abgeleiteten Hauptaussagen.

Grundlage der Formulierung des Standardisierungsproblems ist ein Graph, wie er in Abb. 2.7 dargestellt ist. Dabei werden die Akteure als Knoten und die Beziehungen zwischen den Akteuren als Kanten modelliert. Das Modell ist allgemein gehalten, so dass die Knoten je nach Anwendungsfall sowohl Unternehmen, z. B. in einer Supply Chain, als auch Organisationseinheiten innerhalb eines Unternehmens repräsentieren können.

Dabei werden der Basisnutzen und die Standardisierungskosten knotenbezogen modelliert; die einzusparenden Informationskosten fallen an den Kanten an. Wenn wir im einfachsten Fall davon ausgehen, dass nur ein Standard zur Disposition steht, jeder Akteur also eine Binärentscheidung zu treffen hat, ob er einen gegebenen Standard einführt oder nicht, so können wir den Basisnutzen von Knoten i mit b_i und die Standardisierungskosten von Knoten i mit a_i ($i=1,2,\dots,n$) bezeichnen. Würde nun die Entscheidung über die Einführung eines Softwarestandards nur knotenbezogen und isoliert getroffen werden, so

Abb. 2.7 Beispiel für das Standardisierungsproblem



wäre lediglich für jeden Knoten die Differenz aus Basisnutzen und Standardisierungskosten zu ermitteln. Ist dieser Nettobasisnutzen größer als Null, ist eine Investition in den Softwarestandard sinnvoll, andernfalls entsprechend nicht.

Eine solche isolierte Betrachtung ist für die praktische Anwendung jedoch nicht sinnvoll, da gerade die Kosten für die IT-Integration in der Regel sehr hoch sind. In unserem Modell werden die Informationskosten entlang der Kante von Knoten i zu Knoten j mit c_{ij} bezeichnet. Diese können eingespart werden, wenn beide Akteure den gemeinsamen Standard implementieren, also beispielsweise den gleichen EDI-Standard, die gleiche Standardsoftware oder das gleiche Kommunikationsprotokoll. Es ist natürlich unrealistisch anzunehmen, dass Informationskosten durch die gemeinsame Nutzung von Standards komplett wegfallen. Das stellt für die Modellierung jedoch kein Problem dar, da die Kantenwerte auch als die Differenz zwischen den Informationskosten ohne und mit gemeinsamer Nutzung der Softwarestandards interpretiert werden können.

Die Abb. 2.7 zeigt ein einfaches Beispiel. In den Knoten ist der jeweilige Nettobasisnutzen dargestellt, die Kanten repräsentieren die Kommunikationskosten, welche eingespart werden können. Anhand dieses Beispiels lässt sich zeigen, dass die Kommunikationsbeziehungen zwischen bestimmten Akteuren im Gegensatz zur klassischen Netzeffekttheorie individuell modelliert werden können. So wird der hohe Kommunikationsbedarf zwischen den Knoten 2 und 3 durch Kantenkosten in Höhe von 9 abgebildet. Diese können durch die gemeinsame Nutzung eines Standards eingespart werden. Demgegenüber existiert zwischen Knoten 5 und den anderen ein geringerer Kommunikationsbedarf, so dass durch die Nutzung des gemeinsamen Standards auch nur relativ geringe Informationskosten eingespart werden können.

Zunächst ist es – wie oben dargestellt – optimal, wenn alle jene Knoten den Standard einsetzen, für die der Nettobasisnutzen positiv ist. In diesem Fall sind das die Knoten 1, 2 und 3. Aber auch für einen Knoten mit einem negativen Nettobasisnutzen kann die Standardisierung vorteilhaft sein. Dies ist genau dann der Fall, wenn dieser Knoten enge Kommunikationsbeziehungen mit anderen hat, also entsprechend hohe Informationskos-

ten eingespart werden können. In unserem Beispiel gilt dies für Knoten 4. Obwohl der Nettobasisnutzen für diesen Knoten negativ ist, lohnt sich die Standardisierung aus Gesamtsicht, da insgesamt Informationskosten in Höhe von 15 eingespart werden können, wenn die Knoten 1, 2 und 3 den Standard ebenfalls einführen. Demgegenüber ist eine Standardisierung von Knoten 5 wenig sinnvoll. Die einsparbaren Informationskosten in Höhe von 3 sind geringer als der negative Nettobasisnutzen von -5.

2.3.2 Das zentrale Standardisierungsproblem als Optimierungsproblem

Formal können wir das Standardisierungsproblem als Optimierungskalkül wie folgt darstellen:

$$\max F(x) = \sum_{i=1}^n (b_i - a_i)x_i - \sum_{i=1}^n \sum_{\substack{j=1 \\ j \neq i}}^n c_{ij}(1-x_i x_j)$$

u. d. N.

$$x_i \in \{0,1\}$$

Die Entscheidungsvariable x_i nimmt den Wert 1 an, wenn Akteur i den Standard einsetzt. Nur in diesem Fall kann der Basisnutzen b_i realisiert werden und auch nur dann fallen Standardisierungskosten a_i an. Die Informationskosten c_{ij} können nur gespart werden, wenn sowohl Akteur i als auch Akteur j den Standard einsetzen. Die Zielfunktion zeigt auch, dass das Modell eine Optimierung für den gesamten Graphen anstrebt. Eine Anwendung könnte z. B. so aussehen, dass eine zentrale Instanz, etwa der Chief Information Officer und sein Stab, nach einer optimalen Lösung für das gesamte Unternehmen sucht. Dies kann durchaus bedeuten, dass einzelne Akteure schlechter gestellt werden.

Das obige Modell umfasst nur Situationen, in denen genau ein Standard zur Wahl steht. Im Gegensatz dazu betrachtet das erweiterte Standardisierungsproblem auch Situationen, in denen der Entscheider zwischen mehreren Standards wählen kann – auf die mathematische Modellierung wollen wir in diesem Buch verzichten (Domschke und Wagner 2005).

Zur Veranschaulichung des Modells, zur Optimierung sowie zur Durchführung von Simulationen haben wir einen Prototyp entwickelt. Abbildung 2.8 zeigt beispielhaft die optimale Standardisierungslösung aus zentraler Sicht für ein vollständiges Kommunikationsnetz mit sieben Akteuren bei drei verfügbaren Standards. In diesem Beispiel ist es optimal, mehrere Standards im Netz zu implementieren und sogar einzelne Akteure mit mehr als einem Standard auszustatten.

Mit Hilfe des Prototyps können jedoch auch beliebige Kommunikationsnetze unterschiedlichster Strukturen erzeugt und unter diversen Kostenverteilungen untersucht wer-

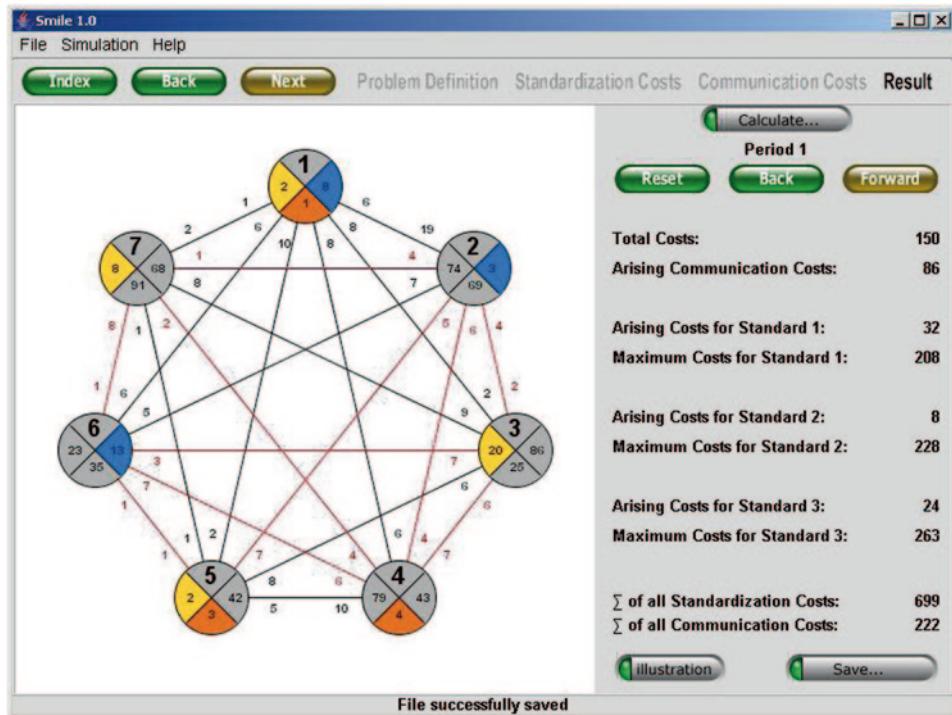


Abb. 2.8 Lösung des Standardisierungsproblems für sieben Akteure. (Schade und Buxmann 2005)

den. So lassen sich beispielsweise die Auswirkungen der Netztopologie auf die Vorteilhaftigkeit bestimmter Standardisierungslösungen, wie etwa Teilstandardisierung oder Komplettstandardisierung, untersuchen; Abb. 2.9 zeigt hierfür ein Beispiel.

Dieses Modell haben wir sowohl analytisch als auch simulativ untersucht (Buxmann und Schade 2007). Auf diese Weise lassen sich die folgenden drei elementaren Aussagen herleiten:

1. Es ist am häufigsten optimal, das komplette Netz entweder vollständig oder gar nicht zu standardisieren.
2. Je größer das Netz ist, desto mehr lohnt sich eine Komplettstandardisierung.
3. Stehen mehrere Softwarestandards zur Auswahl, so ist es nur in seltenen Fällen optimal, mehrere dieser Standards einzusetzen.

Das bedeutet auch, dass Best-of-Breed-Lösungen für den Anwender meist keine ideale Lösung darstellen.

Aus der Sicht eines größeren Softwareanbieters ist es somit im Regelfall optimal, Lösungen für alle Bereiche (Knoten) eines Netzes anzubieten. Die einheitliche Lösung wird für den Anwender in den meisten Fällen vorteilhaft sein. Dies gilt insbesondere bei steigender Unternehmensgröße des Kunden – im Modell ausgedrückt durch die Anzahl der

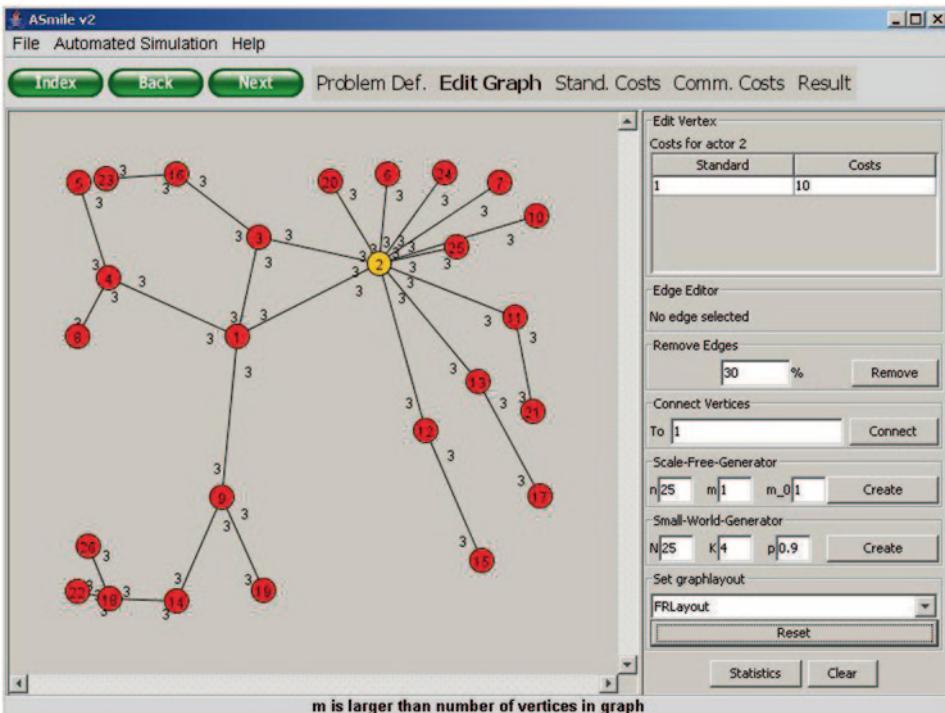


Abb. 2.9 Untersuchung verschiedener Netztopologien

Knoten n. Eine Ausnahme bildet der (in der Praxis jedoch relativ selten anzutreffende) Fall, dass ein Bereich (Knoten) geringe oder gar keine Informationsbeziehungen zu anderen Bereichen (Knoten) hat. Dann kann es für den Kunden optimal sein, verschiedene Softwarestandards im Sinne eines Best-of-Breed-Ansatzes einzusetzen.

Für kleinere Softwarehersteller oder Nischenanbieter gilt natürlich, dass sie ihre Produkte mit kompatiblen Standards ausstatten müssen, um für einen Anwender attraktiv zu sein, der Informationskosten beispielsweise im Sinne von Integrations- und Konvertierungskosten einsparen möchte.

2.3.3 Das dezentrale Standardisierungsproblem – Eine spieltheoretische Darstellung

Die bisherigen Überlegungen unterstellen die Existenz einer zentralen Instanz, die für alle Knoten die Entscheidung über den Einsatz von Standards trifft. Dies würde bedeuten, dass etwa ein CIO gemeinsam mit seinem Team für alle Organisationseinheiten den entsprechenden Softwarestandard auswählen (und durchsetzen) kann. Auf zwischenbetrieblicher Ebene wäre davon auszugehen, dass das stärkste Unternehmen einer Supply Chain den Partnern den Softwarestandard vorschreibt. Was passiert aber, wenn jeder Knoten eigen-

		Akteur j	
		standardisiert	standardisiert nicht
Akteur i	standardisiert	u_i / u_j	$u_i - c_{ij} / -c_{ji}$
	standardisiert nicht	$-c_{ij} / u_i - c_{ji}$	$-c_{ij} / -c_{ji}$

Abb. 2.10 Spieltheoretische Darstellung des Standardisierungsproblems

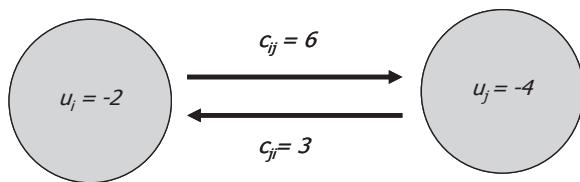
ständig über den Einsatz der Standards entscheidet (Buxmann et al. 1999; Weitzel 2004 sowie Heinrich et al. 2006)?

Zur Beschreibung dieser dezentralen Problemstellung bedienen wir uns eines spieltheoretischen Ansatzes (Weitzel 2004). Dazu vereinfachen wir die Problemstellung, indem wir im Weiteren vom Ein-Standard-Problem ausgehen und nur zwei Akteure i und j betrachten. Dies ist völlig ausreichend, um die Unterschiede einer zentralen oder dezentralen Entscheidung und die daraus resultierenden Konsequenzen darzustellen. Jeder dieser Akteure trifft eine Entscheidung, ob er den gegebenen Standard implementiert oder nicht. Wir führen nun gerichtete Kanten zwischen den Akteuren ein, mit deren Hilfe wir abbilden können, dass die Einführung eines Software-Standards möglicherweise einem Beteiligten mehr nutzt als einem anderen. Formal ausgedrückt: Falls beide Akteure den (gegebenen) Standard implementieren, spart Akteur i (j) Informationskosten in Höhe von c_{ij} (c_{ji}).

Diese Problemstellung lässt sich auf Basis der nicht-kooperativen Spieltheorie allgemein gemäß Abb. 2.10 darstellen:

Diese Matrix ist wie folgt zu lesen: In den Quadranten sind die Ergebnisse der vier möglichen Strategiekombinationen eingetragen, die sich aus den jeweils zwei Alternativen (Standardisierung oder Nicht-Standardisierung) der Akteure i und j ergeben. Beginnen wir links oben: Wenn beide Akteure i und j sich für den Einsatz des Softwarestandards entscheiden, können beide die anfallenden Informationskosten einsparen. Übrig bleibt der Nettobasisnutzen für die Akteure i und j , den wir im Weiteren mit u_i bzw. u_j bezeichnen. Im Quadranten rechts oben führt nur Akteur i den Standard ein. Dies führt zu dem Ergebnis, dass Akteur i nun ein Ergebnis von $u_i - c_{ij}$ erzielt – zum einen den Nettobasisnutzen für die Einführung des Softwarestandards, von dem die Informationskosten abzuziehen sind, die er nicht einsparen kann, da sich Akteur j gegen die Einführung entschieden hat. Akteur j macht in diesem Fall einen Verlust in Höhe der Informationskosten c_{ji} . Für den Quadranten links unten gilt das Gleiche nur mit umgekehrten Akteuren. Der Quadrant rechts unten beschreibt den Fall, dass keiner der beiden Akteure den Standard einführt. Somit fallen keine Standardisierungskosten an, und jeder der beiden Akteure macht einen Verlust in Höhe der Informationskosten c_{ij} bzw. c_{ji} .

Abb. 2.11 Beispielhafte Konstellation für dezentrale Standardisierungsentscheidungen



Im Rahmen einer solchen spieltheoretischen Analyse wird vereinfacht ausgedrückt von einem Nash-Gleichgewicht gesprochen, wenn keiner der Spieler einen Anreiz hat, als Einziger von seiner Entscheidung (in diesem Fall für oder gegen die Einführung des Softwarestandards) abzuweichen. Das Nash-Gleichgewicht ist damit eine Strategiekombination, die sich nicht aus sich selbst heraus zerstört, sondern stabil bleibt.

Diese spieltheoretische Matrix kann nun mit verschiedenen Parameterkonstellationen auf die sich einstellenden Gleichgewichte untersucht werden: Besonders interessant und lehrreich ist für uns der Fall, in dem es aus Gesamtsicht der beiden Akteure optimal wäre, wenn beide den Standard einsetzen würden, aber nur einer der Akteure bei einer dezentralen Betrachtung vom Einsatz des Standards profitiert. Betrachten wir das folgende Beispiel (Abb. 2.11):

Man sieht auf den ersten Blick, dass es aus Gesamtsicht optimal ist, wenn beide Akteure standardisieren. Auch wenn der Nettobasisnutzen negativ ist, die Standardisierungskosten also höher sind als der Bruttobasisnutzen, lohnt sich die Standardisierung, da insgesamt Informationskosten in Höhe von 9 GE eingespart werden können. Demgegenüber macht der negative Nettobasisnutzen lediglich -6 GE aus. Insgesamt ließen sich also drei Geldeinheiten sparen. Das Problem besteht nun darin, dass keine zentrale Instanz existiert, welche die Entscheidung für beide Akteure trifft und die aus Gesamtsicht optimale Lösung sucht. Wird jedoch die dezentrale Perspektive der beiden Akteure eingenommen, so sieht man, dass nur Akteur i von der gemeinsamen Einführung des Software-Standards profitiert: Die Einsparungen an Informationskosten in Höhe von 6 GE sind höher als sein negativer Nettobasisnutzen. Anders sieht es für Akteur j aus, für den die Einführung des Standards nicht sinnvoll ist, weil sein negativer Nettobasisnutzen höher ist als die Informationskosten, die durch die Einführung des Standards gespart werden können.

Abbildung 2.12 zeigt die aus diesem Beispiel resultierende spieltheoretische Matrix.

Diese Matrix ist nun insofern interessant, als sie zeigt, dass die Standardisierungslösung (Quadrant links oben) nicht als Nash-Gleichgewicht in Frage kommt, obwohl diese Lösung aus Gesamtsicht optimal ist. Offensichtlich hat Akteur j einen Anreiz, von dieser Lösung abzuweichen, um 1 GE einzusparen.

2.3.4 Das Standardisierungsproblem – Lessons learned

Der Vergleich zwischen dem dezentralen und dem zentralen Modell verdeutlicht, dass dezentrale Entscheidungen über den Einsatz von Softwarestandards tendenziell zu einem

Abb. 2.12 Spieltheoretische Abbildung des Beispiels

		Akteur <i>j</i>	
		standardisiert	standardisiert nicht
Akteur <i>i</i>	standardisiert	-2 / -4	-8 / -3
	standardisiert nicht	-6 / -7	-6 / -3

geringeren – und häufig aus Gesamtsicht suboptimalen – Standardisierungsgrad führen als zentrale Entscheidungen. Das zurückhaltende Verhalten der Akteure bei der Einführung von Standards beim dezentralen Standardisierungsproblem kann zum einen auf den in Abschn. 2.2.2.1 dargestellten Pinguineffekt zurückgeführt werden. Zum anderen profitieren verschiedene Akteure unterschiedlich stark von der gemeinsamen Einführung der Standards.

Um auch bei dezentralen Entscheidungsstrukturen zu einem aus Gesamtsicht optimalen Ergebnis zu kommen, könnten finanzielle Anreize für diejenigen Akteure sinnvoll sein, die durch die Implementierung des Standards schlechter gestellt werden (Heinrich et al. 2006). So könnte bei dem in Abb. 2.11 dargestellten Beispiel Akteur *i* einen Ausgleichsbetrag an Akteur *j* zahlen, um diesen zum Mitmachen zu bewegen. Dies ist aus zwei Gründen problematisch: Zum einen stellt sich die Frage, wie hoch diese Ausgleichszahlung sein sollte. Zum anderen wird es kaum möglich sein, mit einer Vielzahl von Geschäftspartnern Verhandlungen über mögliche Ausgleichszahlungen zu führen und dabei eine Verteilung zu finden, mit der alle Akteure einverstanden sind (siehe hierzu auch Abschn. 3.1.1 zu Kooperationen in der Softwareindustrie). In der Praxis ist häufig der Fall zu finden, dass das stärkere Unternehmen den kleineren den Standard quasi diktiert. Im Rahmen der Verhandlung über EDI-Standards wurde hier der Ausdruck „EDI or die“ kreiert. Dass es in vielen Fällen jedoch auch im Rahmen der Einführung solcher Standards kooperativ zwischen den Geschäftspartnern zugeht, konnte im Rahmen einer empirischen Untersuchung gezeigt werden. Demnach unterstützten in der Automobilindustrie in ca. 70% der Fälle die großen Hersteller die kleineren Partner, insbesondere Zulieferer, bei der Einführung von Softwarelösungen (Buxmann et al. 2005).

Darüber hinaus zeigt das Standardisierungsproblem, dass es für den Anwender in der Regel am besten ist, lediglich auf einen Standard zu setzen. Daraus folgt, dass interne Netzeffekte (beispielsweise innerhalb eines Unternehmens oder innerhalb einer Supply Chain) die Stellung großer Softwareanbieter stärken. Der Versuch, gerade in einer großen Organisation eine Vielzahl unterschiedlicher Systeme zu etablieren, wird in vielen Fällen zu hohen Informationskosten führen, etwa im Sinne von Integrations- und Konvertierungskosten. Oftmals wird es daher aus Gesamtsicht optimal sein, eine integrierte Lösung

eines Herstellers zu nutzen – möglicherweise ergänzt um kompatible Produkte kleinerer Softwarehersteller und Nischenanbieter. Best-of-Breed-Lösungen sind für den Anwender lediglich bei sehr geringem Informationsaustausch und daraus resultierend niedrigen Informationskosten zwischen den Teilsystemen vorteilhaft – eine Konstellation, die in der Praxis nur selten zu finden ist. Eine interessante Frage für Anwender und Anbieter lautet, inwieweit service-orientierte Architekturen und Plattformen durch die Nutzung offener Standards einen Beitrag dazu leisten können, die Kommunikationskosten zu reduzieren und damit Best-of-Breed-Lösungen attraktiver zu machen.

Auch aus der Sicht eines Anbieters wird es häufig nicht sinnvoll sein, alle möglichen Standards in seine Lösungen einzubauen (Plattner 2007, S. 3). Vielmehr gilt es, auf den richtigen Standard zu setzen, mit dem der Kunde Interoperabilität zwischen den Plattformen verschiedener Hersteller erreichen kann. Vor diesem Hintergrund ist es natürlich auch für die Anbieter von großer Bedeutung, sich an der Entwicklung von Standards zu beteiligen, um ihre eigenen strategischen Interessen durchzusetzen. Dabei treten bei der Entwicklung von Standards jedoch immer wieder Probleme auf, die u. a. auf die folgenden Gründe zurückzuführen sind (Plattner 2007, S. 3):

- Anbieter vertreten in Standardisierungsprozessen häufig unterschiedliche ökonomische Interessen (auf die Hintergründe sind wir bereits in Abschn. 2.2.4 eingegangen).
- Standardisierungsprozesse werden in vielen Fällen durch typische Komiteearbeit (Plattner spricht von „committee thinking“) verlangsamt.
- Die Mitarbeit in Standardisierungsgremien ist für die Mitarbeiter häufig nur eine zusätzliche Aufgabe, die sie parallel zu ihrem Hauptjob zu erledigen haben. Dementsprechend sieht auch die Priorisierung ihrer Tätigkeiten aus.

2.4 Transaktionskostentheorie: Auf der Suche nach den Grenzen eines Softwareunternehmens

Die Positionierung in der Wertschöpfungskette ist für ein Unternehmen von größter Wichtigkeit. Dabei geht es im Kern um die klassische Make-or-Buy-Entscheidung, also um die Frage, was ein Unternehmen selbst produziert bzw. entwickelt und was von anderen Unternehmen eingekauft wird. Die Transaktionskostentheorie liefert interessante Einsichten zu diesen Fragen. In Abschn. 2.4.1 stellen wir Ansatzpunkt und Elemente der Transaktionskostentheorie vor und entwickeln in Abschn. 2.4.2 ein grundlegendes Kalkül für die ökonomisch effiziente Arbeitsteilung zwischen Unternehmen. Zudem zeigen wir Anwendungsfelder der Transaktionskostentheorie in Softwareunternehmen auf. Strukturelle Veränderungen der Transaktionskosten, wie sie aktuell insbesondere von neuen Kommunikationstechnologien ausgehen, präsentieren wir in Abschn. 2.4.3. Das Kapitel schließt mit ergänzenden Überlegungen zur transaktionskostenreduzierenden Wirkung von Intermediären in Abschn. 2.4.4.

2.4.1 Ansatzpunkt und Elemente der Transaktionskostentheorie

Zwischen den Akteuren einer arbeitsteilig organisierten Wirtschaft bestehen zahlreiche und vielfältige Austauschbeziehungen. Diese bilden den Ausgangspunkt transaktionskostentheoretischer Überlegungen (Williamson 1985; Picot und Meier 1992). Im Mittelpunkt steht dabei aber nicht der Tausch von Gütern und Dienstleistungen an sich, sondern vielmehr die dem Tausch vorgelagerte Übertragung von Verfügungsrechten. Diese wird als Transaktion bezeichnet. Stellt ein Softwareanbieter seinem Kunden die für ihn entwickelte Software zur Verfügung, dann ist dies genauso eine Transaktion wie die Übernahme eines von einem Dienstleister in einem Niedriglohnland erstellten Moduls oder die Weitergabe eines Fachkonzepts von Abteilung A zu Abteilung B innerhalb eines Unternehmens. Um Missverständnissen vorzubeugen: Verwendet wird der Begriff der Transaktion in der Softwarebranche auch in Datenbanksystemen und bei der Spezifikation von Anwendungssystemen. Mit beiden hat unser ökonomisch geprägter Transaktionsbegriff aber gar nichts zu tun.

Transaktionen verursachen Kosten – eine Erkenntnis, die übrigens in der wirtschaftswissenschaftlichen Theorie über sehr lange Zeit ausgeklammert wurde. Vornehmlich handelt es sich dabei um Kosten für Information und Kommunikation, die bei der Anbahnung, Vereinbarung, Abwicklung, Kontrolle und Anpassung wechselseitiger Leistungsbeziehungen auftreten. Illustrieren wir dies anhand eines Beispiels. Ein Individualsoftwareanbieter wird u. a. auf Messen, Konferenzen und in Fachzeitschriften präsent sein, um auf sein Leistungsangebot aufmerksam zu machen. Genauso wird der Auftraggeber sich einen Überblick über den Markt verschaffen. In beiden Unternehmen fallen also nicht unerhebliche Anbahnungskosten an. Ebenfalls recht kostenintensiv werden die Verhandlungen über den Leistungsumfang und den Preis der zu erstellenden Software sein, die der Vereinbarungsphase zuzurechnen sind. Genauso werden aber auch Kosten für die Kontrolle (z. B. für Tests) und gegebenenfalls für eine spätere Anpassung (z. B. bei Wandel der Anforderungen) anfallen. In Summe ist damit die Bereitstellung einer Individualsoftware ein transaktionskostenintensiver Vorgang.

Genau hier setzt die Transaktionskostentheorie an. Sie will aufzeigen, in welcher Organisationsform sich eine Transaktion in einer gegebenen Umwelt am kostengünstigsten abwickeln lässt: Gesucht ist also die transaktionskostenminimale Organisationsform. Anwenden lässt sich dieser Ansatz sowohl auf die Arbeitsteilung zwischen Unternehmen als auch innerhalb eines Unternehmens. Auf die Arbeitsteilung zwischen Unternehmen werden wir nachfolgend noch näher eingehen. Vorab wollen wir aber noch die wesentlichen Determinanten der Entstehung von Transaktionskosten sowie die dem Transaktionskostenkalkül zu Grunde liegenden Annahmen kurz herausarbeiten.

Die Transaktionskostentheorie sieht in dem Umfang transaktionsspezifischer Investitionen (verkürzt häufig als „Spezifität“ bezeichnet) die wesentlichste Determinante für die Kosten einer Transaktion. Die dahinter stehende Überlegung ist relativ einfach. Zwar sollten transaktionsspezifische Investitionen die Produktionskosten senken, gleichzeitig wird aber durch hohe transaktionsspezifische Investitionen zwischen den Transaktions-

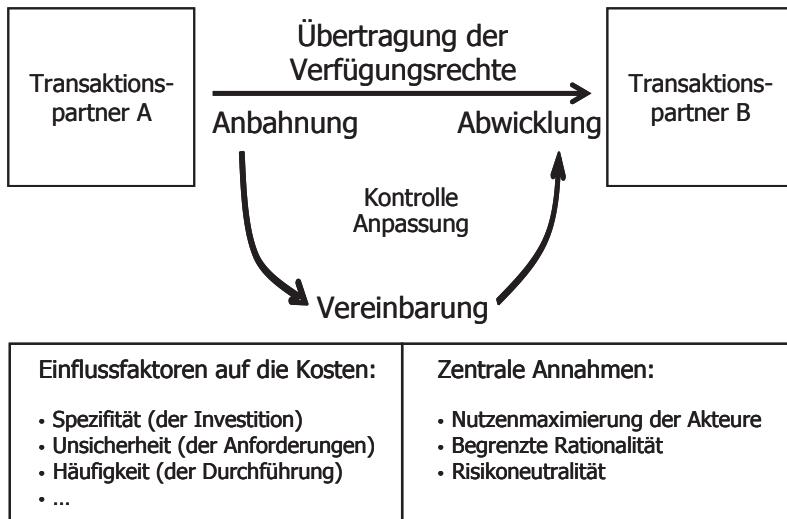


Abb. 2.13 Grundkonstrukt der Transaktionskostentheorie

partnern eine besondere Abhängigkeit begründet. Denn nachdem die Investition getätigt ist, kann der Transaktionspartner nur unter Inkaufnahme von Nachteilen gewechselt bzw. müssen „Nachverhandlungen“ mit dem bisherigen Transaktionspartner in Kauf genommen werden. Derartige Abhängigkeiten bestehen beispielsweise bei einer umfangreichen Auslagerung von IT-Services. Transaktionsspezifische Investitionen führen damit tendenziell erst einmal zu steigenden Transaktionskosten. Darüber hinaus wird die Höhe der Transaktionskosten vom Grad der Unsicherheit der Transaktion, der Häufigkeit der Transaktionsdurchführung, der strategischen Bedeutung der Transaktion und der Transaktionsatmosphäre beeinflusst.

Bezüglich des Verhaltens der Akteure geht die Transaktionskostentheorie davon aus, dass diese opportunistisch, begrenzt rational und risikoneutral handeln. Zusammen mit anderen wesentlichen Merkmalen sind diese Annahmen in Abb. 2.13 noch einmal zusammenfassend dargestellt.

Opportunistisch handelnde Akteure verfolgen im Zweifelsfall ausschließlich ihre eigenen Interessen. Hat also einer der Partner an einer Transaktion die Chance, sein Gegenüber zu den besagten „Nachverhandlungen“ zu bewegen, dann wird er dies im Regelfall auch tun. Bestimmte Informationen, etwa im Hinblick auf die Kompatibilität zu anderen Lösungen, werden vermutlich nicht immer weitergegeben. Auch halten sich opportunistisch agierende Akteure nicht unbedingt an Zusagen, wenn andere dies nicht überprüfen können (z. B. zur Qualität eines Softwaremoduls). Denn gerade in der Softwareindustrie lässt sich die Qualität eines Produkts häufig erst in der konkreten Anwendungssituation umfassend beurteilen, was einen erheblichen opportunistischen Spielraum eröffnet. Begrenzte Rationalität bedeutet hierbei, dass die Transaktionspartner nicht über alle grundsätzlich verfügbaren Informationen selbst verfügen (z. B. zur Zahlungsfähigkeit eines Kunden).

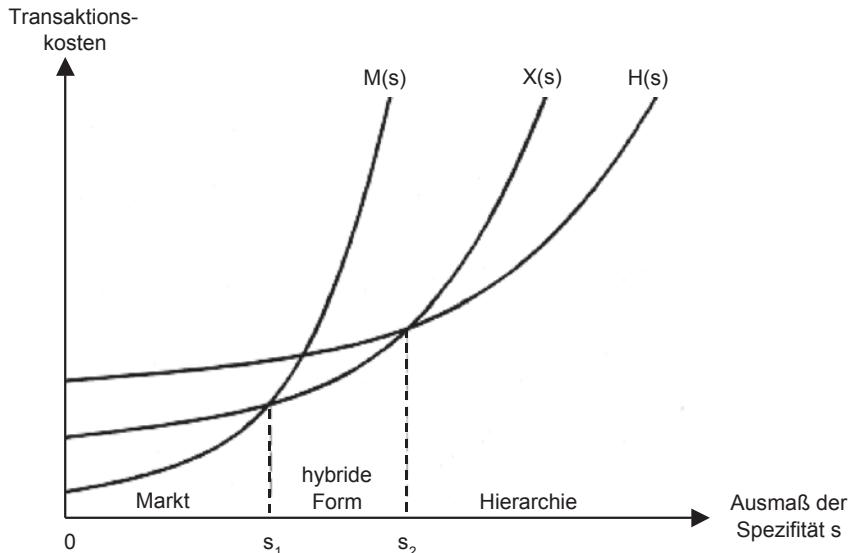


Abb. 2.14 Transaktionskostenverläufe in Abhängigkeit von der Spezifität. (Williamson 1991, S. 284)

2.4.2 Arbeitsteilung zwischen Unternehmen aus Sicht der Transaktionskostentheorie

Der wohl bekannteste Zweig der Transaktionskostentheorie beschäftigt sich mit den Grenzen eines Unternehmens. Zu diesem Zweck wird bewusst vereinfachend zwischen Markt und Unternehmen sowie – was erst später dazu gekommen ist – Kooperationen als Mischform von Markt und Hierarchie unterschieden. Charakteristisch für diese drei Grundtypen sind die jeweiligen Koordinationsmechanismen: der Preis im Markt, die Anordnung im Unternehmen und eine Mischung aus Preis und Anordnung in der Kooperation.

Wenn Transaktionen nicht mit spezifischen Investitionen verbunden, also unspezifisch sind, schlägt die Transaktionskostentheorie den Markt als effizientesten Koordinationsmechanismus vor. Sind die Transaktionspartner dagegen über hohe transaktionskosten-spezifische Investitionen stark voneinander abhängig, dann steigt für sie der Anreiz, die Abhängigkeit des jeweils anderen opportunistisch auszunutzen. Mittels der Hierarchie lässt sich dieser opportunistische Spielraum jedoch wirksam begrenzen. Bei hoher Spezifität ist daher die Hierarchie das effizienteste Koordinationsinstrument. Konsequenterweise ist die Kooperation bei mittlerer Spezifität das zu wählende Koordinationsinstrument, weil dort Markt und Hierarchie miteinander kombiniert werden.

Die Transaktionskostentheorie hat diese Überlegungen noch etwas weiterentwickelt, was in Abb. 2.14 verdeutlicht wird. Nach dieser Darstellung ist der Markt bei einer Spezifität bis zu s_1 die effizienteste Koordinationsform. Bei der Spezifität zwischen s_1 und s_2 ist dagegen die Kooperation und bei einer Spezifität größer als s_2 ist das Unternehmen als Koordinationsform zu wählen.

Abbildung 2.14 zeigt anschaulich die Höhe der Transaktionskosten für unterschiedliche Organisationsformen in Abhängigkeit von der Spezifität. Dabei ist der exakte Kurvenverlauf allerdings weder analytisch noch empirisch herleitbar, weshalb diese Darstellung in der Literatur auch nicht unumstritten ist.

Ursprünglich verfolgte die Transaktionskostentheorie eine eher gesamtwirtschaftliche Fragestellung, indem sie die Entstehung von Unternehmen und Kooperationen als Alternativen zum Markt erklären wollte. Mittlerweile ist sie zu einem betriebswirtschaftlichen Standardinstrument geworden, um die Grenzen eines einzelnen Unternehmens zu bestimmen. Typischerweise geht es dabei um die Frage nach dem In- und Outsourcing einzelner Aufgaben und den Beziehungen zum beauftragten Unternehmen. In der Softwareindustrie hat dieses Thema in seinen verschiedenen Ausprägungen eine sehr große Bedeutung, angefangen beim Outsourcing an spezialisierte Unternehmen bis hin zu Offshore-Anbietern. Wir werden es daher in Abschn. 5.1 näher betrachten.

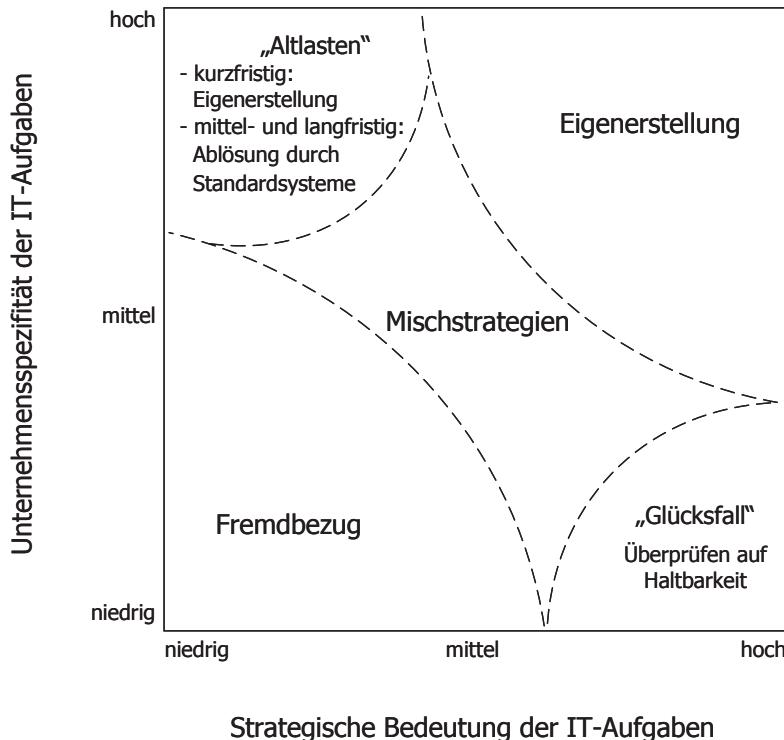
Speziell in Softwareunternehmen eröffnet die Transaktionskostentheorie aber auch interessante Einsichten in die Überlegungen des Endkunden, speziell des kommerziellen Anwenders. Im Gegensatz zum eben skizzierten Themenfeld des Sourcings eigener Aufgaben ist dieses Themenfeld in der Literatur schon seit Jahren hinlänglich bearbeitet worden. Entwickelt wurden u. a. auch einfache Managementinstrumente, die Entscheidungen über die Auslagerung von IT-Aufgaben unterstützen sollen. Die Abb. 2.15 zeigt einen entsprechenden Vorschlag von Picot und Meier.

Picot und Meier greifen neben der Spezifität auch die strategische Bedeutung von IT-Aufgaben auf, die wir oben bei der Nennung der Determinanten der Transaktionskosten am Rande bereits erwähnt haben. Im Kern empfehlen sie, nur IT-Aufgaben mit einer eher geringen Spezifität und einer ebenso geringen strategischen Bedeutung vollständig auszulagern. In vielen Unternehmen liegt der Betrieb des Netzwerks genau in diesem Quadranten, was wiederum erklärt, warum Outsourcing häufig hier startet.

2.4.3 Strukturelle Veränderungen der Transaktionskosten: The Move to the Middle

Wir haben bereits dargestellt, dass die Transaktionskostentheorie die Suche nach der kostengünstigsten Organisationsform für eine gegebene Umwelt unterstützt. In der Realität – gerade in der Softwarebranche – ist die Umwelt aber keineswegs stabil. Insbesondere neue Kommunikationstechnologien sowie die Öffnung neuer Länder mit beachtlichem IT-Know-how und relativ geringen Lohnkosten verändern diese Umwelt massiv. Wir skizzieren daher nachfolgend den Einfluss dieser Veränderungen auf die Transaktionskosten und damit auf die Wahl der kostengünstigsten Organisationsform.

Es besteht heute weitestgehend Einigkeit darüber, dass neue Kommunikationstechnologien wie das Internet und die darauf aufbauenden Dienste zu einer Reduzierung der variablen Transaktionskosten und damit zu einer Absenkung und Verflachung der Kurve in Abb. 2.14 führen (Bauer und Stickel 1998). Als Konsequenz verschiebt sich der Schnitt-



Strategische Bedeutung der IT-Aufgaben

Abb. 2.15 Entscheidungsmatrix für die Auslagerung von IT-Aufgaben. (Picot und Meier 1992, S. 21)

punkt für den Übergang zur jeweils nächsten Koordinationsform nach rechts. Dies bedeutet, dass erst bei einem höheren Grad an Spezifität der Wechsel zu Kooperationen bzw. zur Hierarchie ökonomisch sinnvoll ist. Unter dem Schlagwort „Move-to-the-Market“ wird diese Entwicklung recht plakativ bezeichnet. Die Kernaussage besteht darin, dass eine unternehmensinterne Koordination zu Gunsten von Kooperationen und Märkten in Folge neuer Kommunikationstechnologien an Bedeutung verliert und sich die Wertschöpfungsketten damit fragmentieren. Für die Softwarebranche würde dies bedeuten, dass eine Vielzahl spezialisierter Unternehmen zu erwarten ist – eine Vision, die durch das neue serviceorientierte Paradigma möglicherweise unterstützt wird.

Es ist jedoch auch zu beobachten, dass viele Unternehmen heute intensiver zusammenarbeiten, häufig allerdings mit einer geringeren Anzahl von Partnern. Diese Kooperationsstrategie erfordert transaktionsspezifische Investitionen, beispielsweise für die Vereinbarung und Implementierung spezieller Varianten für den Datenaustausch. In der Logik der Transaktionskostentheorie führt dies zu einer Erhöhung der Spezifität und damit zu einer Gegenbewegung zu „Move-to-the-Market“ (Clemons et al. 1993). In Summe führen beide Trends zusammen damit zu einer Vorteilhaftigkeit von Kooperationen, was unter dem Schlagwort „Move-to-the-Middle“ diskutiert wird.

Ebenfalls bedeutend sind die bereits angesprochenen neuen Möglichkeiten der Auslagerung von IT-Dienstleistungen in Niedriglohnländer. Klassisch-ökonomisch betrachtet kommen damit neue Anbieter ins Spiel, die den Preis am Markt der Endverbraucher tendenziell reduzieren bzw. erstmals die Teilvergabe von Aufgaben an Dritte interessant machen. Gerade an diesem Punkt kommen die Transaktionskosten ins Spiel. Ein Softwareanbieter wird beispielsweise bestimmte Dienstleistungen nur dann an ein Softwarehaus in Indien vergeben, wenn verminderte Entwicklungskosten als Folge massiver Lohnunterschiede nicht durch erhöhte Transaktionskosten überkompensiert werden. Die Quellen für erhöhte Transaktionskosten sind vielfältig: von der aufwändigeren Spezifikation der Software und gegebenenfalls erforderlichen Anpassungen bis hin zu zusätzlichen Kosten für die Kontrolle des Dienstleisters. Schon diese einfachen Überlegungen verdeutlichen, dass die Auslagerung von Teilen der Softwareentwicklung keineswegs automatisch die Gesamtkosten senkt. Zu Fehlentscheidungen kommt es auch deshalb, weil aus der Kostenrechnung eines Unternehmens typischerweise nur die Produktions-, aber nicht die Transaktionskosten ablesbar sind. Wir werden in Abschn. 5.1 dieses Buches auf diese Themenstellung zurückkommen.

2.4.4 Ausblick: Intermediäre und Transaktionskosten

Bezugspunkt der Transaktionskostentheorie sind Transaktionen, die wir zuvor als Übertragung von Verfügungsrechten definiert haben. Ausgehend von der Transaktion als Analyseeinheit liefert die Transaktionskostentheorie insbesondere in Abhängigkeit von dem Umfang transaktionsspezifischer Investitionen Hinweise auf die ökonomisch adäquate Organisationsform. Daneben wird die Transaktion aber immer wieder auch als Konstrukt herangezogen, wenn es um die ökonomische Sinnhaftigkeit von Intermediären geht.

Intermediäre finden sich in der Softwarebranche bisher überwiegend in Form von Absatzmittlern, die gerade im Geschäft mit Privatkunden als Zwischenhändler zwischen Produzenten und Nutzern einer Software geschaltet sind. Mit der Zerlegung klassischer Standardsoftwarepakete in Module und mit dem Einzug von Services werden Intermediäre in den nächsten Jahren aber auch in anderen Feldern der Branche ein bisher unbekanntes Gewicht erlangen. Wir stellen daher nachfolgend die Grundidee der Intermediationstheorie vor, auch wenn diese nicht unmittelbar der Transaktionskostentheorie zuzurechnen ist. Aufgreifen werden wir die Intermediationstheorie dann u. a. im Abschnitt zur Distributionspolitik sowie zur Industrialisierung.

Intermediäre sind Unternehmen, die keine Produkte oder Dienstleistungen selber herstellen, sondern vielmehr ein Wirtschaftsgut bzw. Bündel von Wirtschaftsgütern einem Nachfrager bereitstellen. Das wohl bekannteste Beispiel sind die bereits erwähnten Einzelhändler, die dem Verbraucher eine Vielzahl von Produkten unterschiedlichster Hersteller an einem für ihn gut erreichbaren Ort anbieten. Intermediäre sind aber z. B. auch Verlage oder Fernsehsender, die sehr häufig Inhalte nicht selber erstellen, sondern diese von freiberuflichen Autoren oder spezialisierten Produktionsgesellschaften aufkaufen und

den Präferenzen der Kunden entsprechend in Form einer Tageszeitung oder eines Fernsehkanals anbieten. Für die Softwarebranche hatten wir bereits eine Reihe von Beispielen genannt. Intermediäre haben immer dann eine Existenzberechtigung, wenn die Abwicklung einer Transaktion über sie kostengünstiger ist als eine direkte Abwicklung zwischen Anbieter und Nachfrager.

Mit Hilfe des Kalküls von Baligh und Richartz lässt sich die Wirkung eines Intermediärs zumindest bezüglich der Suchphase einer Transaktion abschätzen (Baligh und Richartz 1967). Gegeben sei dafür ein Markt mit m Anbietern und n Nachfragern. Möchte sich ein Anbieter nun einen Überblick über das Angebot verschaffen, so muss er alle m Anbieter befragen. Da dies für jeden der n Nachfrager gilt, ergeben sich $m \times n$ Kontakte. Ist dagegen ein Intermediär eingeschaltet, dann braucht jeder Anbieter und jeder Nachfrager seine Informationen nur bei einem Intermediär zu hinterlegen, woraus sich $m + n$ Kontakte ergeben. Schon bei recht kleinem n und m ist damit ein Intermediär hinsichtlich der Kontaktzahl günstiger – und damit ökonomisch sinnvoll.

2.5 Softwareentwicklung als Agency-Problem: Anreizkompatible Entlohnung und effiziente Kontrolle

Bei der Entwicklung von Individualsoftware, aber auch im Rahmen der Implementierung, Wartung und beim Betrieb von Standardsoftwarelösungen, arbeiten Softwareunternehmen und Kunden eng zusammen. Durch diese enge Zusammenarbeit entstehen mehrschichtige Abhängigkeiten, die in der Ökonomie als Principal-Agent-Probleme bekannt sind. Im nachfolgenden Kapitel greifen wir diese Idee auf (Abschn. 2.5.1), zeigen die praktische Anwendung mittels Entlohnungsschemata (Abschn. 2.5.2) und Kontrollsystmen (Abschn. 2.5.3).

2.5.1 Principal-Agent-Beziehungen: Definitionen und Grundlagen

Im Mittelpunkt der Principal-Agent-Theorie (Spence und Zeckhauser 1971; Ross 1973; Jensen und Meckling 1976) steht die arbeitsteilige Beziehung zwischen Principal und Agent. Eine derartige Beziehung entsteht, wenn ein Auftraggeber (Principal) einem Ausführenden (Agent) zur Realisierung seiner Interessen Entscheidungs- sowie Ausführungsbefugnisse überträgt und ihm dafür eine Entlohnung bietet.

Für unsere Zwecke ist insbesondere die Beziehung zwischen einem Softwareunternehmen als Agent und einem beauftragenden Unternehmen als Principal wichtig, in der das Softwareunternehmen die Entwicklung, die Bereitstellung, die Implementierung oder den Betrieb einer Software übernimmt und dafür entlohnt wird. Ebenfalls relevant in unserem Kontext ist aber auch eine zweite Principal-Agent-Beziehung: Vergibt ein Softwareunternehmen einen Teil des Entwicklungsprozesses an ein anderes Unternehmen, etwa die Entwicklung eines Softwaremoduls an ein Unternehmen in einem Land mit geringeren Lohn-

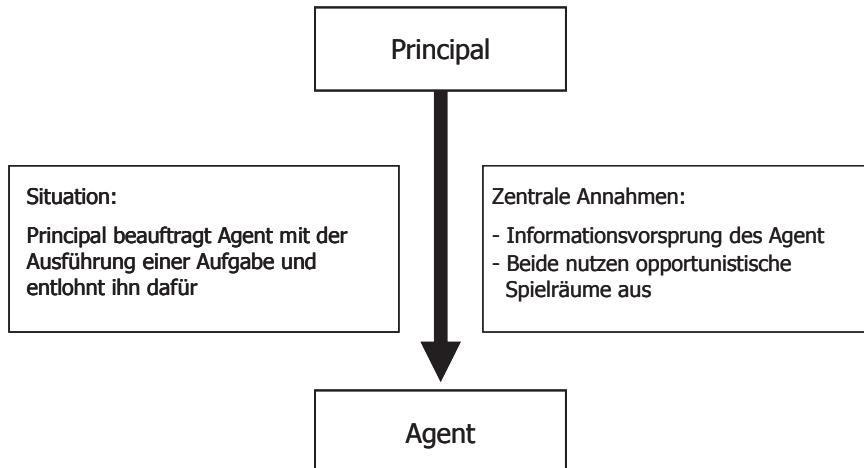


Abb. 2.16 Principal-Agent-Beziehung

kosten, dann wird das Softwareunternehmen zum Principal und der Anbieter im Niedriglohnland nimmt die Rolle eines Agent ein (siehe hierzu auch Kap. 5 dieses Buches).

Die Principal-Agent-Theorie beruht auf zwei zentralen Annahmen. Zum einen wird davon ausgegangen, dass sowohl Principal als auch Agent nach der Maximierung ihres individuellen Nutzens streben und sich hierfür bietende „opportunistische“ Spielräume (z. B. durch den überhöhten Ausweis des Arbeitseinsatzes bei der Programmierung) systematisch ausnutzen. Zum anderen wird angenommen, dass der Agent einen Informationsvorsprung hinsichtlich der Bearbeitung der ihm gestellten Aufgabe besitzt. So weiß er beispielsweise in der Regel besser als der Principal, wie viele Personenmonate für ein Entwicklungsprojekt aufzuwenden sind, und wird seine wahren Interessen (z. B. die langfristige Bindung des Principals an sein Unternehmen) häufig nicht offen legen. Principal und Agent handeln damit vor dem Hintergrund der ihnen vorliegenden (unvollständigen) Informationen rational, was als begrenzte Rationalität bezeichnet wird. Abbildung 2.16 zeigt dieses grundlegende Modell im Überblick.

Opportunistisches Verhalten und begrenzte Rationalität in der Beziehung zwischen Principal und Agent können sich in drei Problemfeldern niederschlagen, die gleichzeitig auch die zeitliche Struktur einer solchen Beziehung beschreiben.

Hidden Characteristics treten vor Vertragsabschluss auf. Der Principal kennt die Qualitätsmerkmale des Agent sowie der von ihm angebotenen Leistung zunächst nicht bzw. nur unvollständig. Für den Principal besteht somit die Gefahr, dass er einen ungeeigneten Agent auswählt. Genau dies ist der Fall, wenn ein Kunde ein Softwareunternehmen beauftragt, dem die fachliche Kompetenz für eine ganz spezielle Lösung letztlich dann doch fehlt.

Hidden Action beschreibt die Gefahr, dass der Principal die Handlungen des Agent nicht beobachten bzw. nicht beurteilen kann. In beiden Fällen kennt der Principal zwar das Ergebnis, kann aber nicht einschätzen, inwieweit dieses Ergebnis auf die Anstrengung

des Agent oder auf exogene Faktoren zurückzuführen ist. Der Agent hat daher prinzipiell die Möglichkeit, diesen Spielraum opportunistisch zu nutzen. So kann ein beauftragendes Unternehmen beispielsweise nur schwer abschätzen, welche Qualitätssicherungstests mit welchen Ergebnissen durchgeführt worden sind.

Hidden Intention bezeichnet die Gefahr für den Principal, dass er in Abhängigkeit vom Agent gerät, weil er auf dessen Leistung angewiesen ist. Genauso wie Hidden Action tritt Hidden Intention erst nach Vertragsabschluss auf. Der Agent kann dies ausnutzen, um sich einen Vorteil zu verschaffen. Schließt ein Kunde z. B. einen Outsourcing-Vertrag ab, so gerät er zwangsläufig in Abhängigkeit vom Outsourcing-Anbieter.

Hidden Characteristics, Hidden Action und Hidden Intention führen zu so genannten Agency-Kosten beim Principal. Um diese Kosten zu reduzieren, stehen ihm insbesondere Entlohnungsschemata und Kontrollsysteme zur Verfügung. Beide Ansätze beschreiben wir nachfolgend, wodurch wir einen Einblick sowohl in die quantitative als auch in die qualitative Forschung zur Principal-Agent-Theorie geben.

2.5.2 Anreizkompatible Vergütungsschemata

Werk- oder Dienstverträge sind die Basis für die Entlohnung erbrachter Leistungen. Die Ausgestaltung von Vergütungsschemata richtet sich danach, welcher Vertragstyp vorliegt. Nachfolgend sind Vergütungsschemata für beide Varianten beschrieben (Picot und Ertsey 2004).

Entlohnung auf der Basis eines Werkvertrags Mit einem Werkvertrag verpflichtet sich der Auftragnehmer, eine vorab definierte Leistung (ein Werk) mit klar definierten Merkmalen zu vereinbarten Konditionen zu erbringen. Bei der Entlohnung geht es daher „nur“ noch um den Preis. Grundsätzlich besteht die Möglichkeit, den Auftragnehmer entweder durch die Übernahme der tatsächlich angefallenen Kosten plus einer Marge („Cost plus“) oder über einen Festpreis zu entlohnen.

Bei einem Cost-plus-Vertrag trägt alleine der Auftraggeber das Kostenrisiko des Projekts. Für den Auftragnehmer besteht in dieser Konstellation prinzipiell die Möglichkeit, höhere Kosten als die tatsächlich angefallenen abzurechnen, was auch durch Offenlegungsvorschriften nur zum Teil relativiert werden kann. Ein Cost-plus-Vertrag liegt insbesondere nahe, wenn die Projektspezifikation relativ unvollständig ist und der Auftragnehmer seine Produktionskosten kaum einschätzen kann. Ist eine der beiden genannten Bedingungen aber nicht erfüllt, dann bietet sich ein Festpreis-Vertrag an. Hierbei zahlt der Auftraggeber nach Projektabschluss einen zuvor verabredeten festen Preis. Das Kostenrisiko des Projekts liegt damit alleine beim Auftragnehmer. Liegt dieser am Ende des Projekts mit seinen Kosten unter dem vereinbarten Preis, erzielt er einen (zusätzlichen) Gewinn. Übersteigen seine Kosten nach Projektabschluss hingegen den vereinbarten Preis, reduziert sich seine Marge entsprechend, im schlechtesten Fall muss er sogar draufzahlen. Sicherlich hat der Auftragnehmer bei einem Festpreisprojekt den Anreiz, das Projekt

effizient durchzuführen. Gleichwohl hat er aber auch kein Interesse, eventuell erreichte Einsparungen an den Auftraggeber weiterzugeben. Wir kommen auf die Methoden zur Aufwandsschätzung und Ansätze zur Preisfindung bei Individualsoftwareprojekten in Abschn. 3.3.3 zurück.

Beide vorgestellten Varianten des Vertragsabschlusses sind bezüglich des Risikos einseitig, da in jedem Fall das Risiko vollständig auf einen der beiden Akteure abgewälzt wird. Damit eröffnen sich den jeweils anderen Akteuren entsprechende opportunistische Spielräume. Das nachfolgende Modell zeigt für Cost-plus-Verträge, wie sich Projektrisiken verteilen lassen. Dazu nehmen wir an, dass ein Auftragnehmer insgesamt die Entlohnung e erhält. Diese setzt sich aus einer fixen Vergütung e_{\min} und einem Anteil α an der Plan-Ist-Abweichung der Entwicklungskosten $k_i - k_p$ zusammen, wobei k_p für die geplanten und k_i für die tatsächlich entstandenen Projektkosten steht. Auftraggeber und Auftragnehmer teilen sich damit die zusätzlichen Kosten. Liegt k_i unter dem im Vertrag festzulegenden Minimalwert k_{\min} , dann bekommt der Auftragnehmer trotzdem die Minimalvergütung e_{\min} – er realisiert damit zusätzliche Deckungsbeiträge. Liegt k_i dagegen über k_{\max} , dann erhält der Auftragnehmer trotzdem lediglich die vereinbarte Maximalentlohnung e_{\max} – seine Marge reduziert sich bzw. er muss sogar draufzahlen. Es ergibt sich folgendes Entlohnungsschema, das die Risiken für beide Seiten begrenzt:

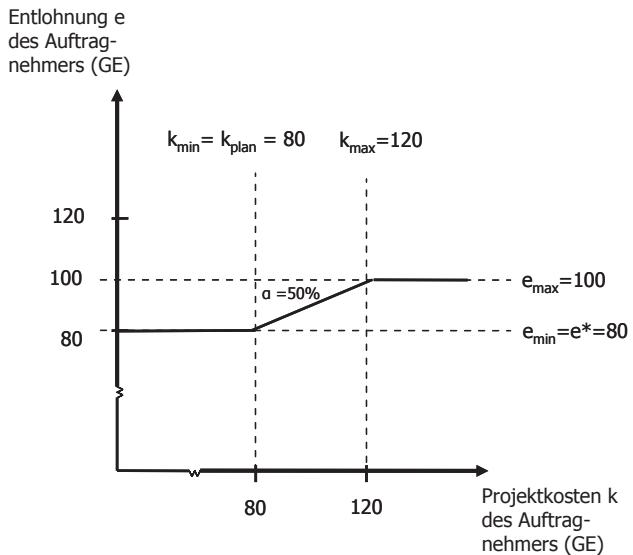
$$e = \begin{cases} e_{\min} & \text{für } k_i < k_{\min} \\ e_{\min} + \alpha \cdot (k_i - k_p) & \text{für } k_{\min} \leq k_i \leq k_{\max} \\ e_{\max} & \text{für } k_i > k_{\max} \end{cases}$$

Abbildung 2.17 illustriert dieses Entlohnungsschema an einem Beispiel, in dem k_p und auch k_{\min} 80 Geldeinheiten (GE) betragen.

Darüber hinaus gehende Entwicklungskosten bis zu einer Höhe von 120 GE werden von Auftraggeber und Auftragnehmer zu gleichen Teilen übernommen, d. h., bei $k_i=100$ GE erhält der Auftragnehmer eine Entlohnung von $80 \text{ GE} + 0,5 \cdot (100 \text{ GE} - 80 \text{ GE}) = 90 \text{ GE}$. Projektkosten über 120 GE gehen alleine zu Lasten des Auftragnehmers. Liegen die Projektkosten dagegen unter 80 GE, profitiert der Auftragnehmer alleine von den erwirtschafteten Effizienzvorteilen.

Entlohnung auf der Basis eines Dienstvertrags Bei einem Dienstvertrag verpflichtet sich der Auftragnehmer zu einer bestimmten Arbeitsleistung, wie etwa einer Modellier- oder Programmierungstätigkeit oder der Bereitstellung eines Services. Wird ein fixer Preis für die Bereitstellung der Arbeitsleistung vereinbart, hat der Anbieter den Anreiz, seine Leistung kostengünstig zu erbringen und möglicherweise weniger genau auf deren Qualität zu achten – schließlich hat er sich ja „nur“ verpflichtet, eine Arbeitsleistung bereit zu stellen. Als Gegenmittel bieten sich anreizkompatible Entlohnungsschemata an, die die Entlohnung in Verbindung mit der erbrachten Leistung setzen.

Abb. 2.17 Beispiel für Entlohnungsschema auf Basis eines Werkvertrags



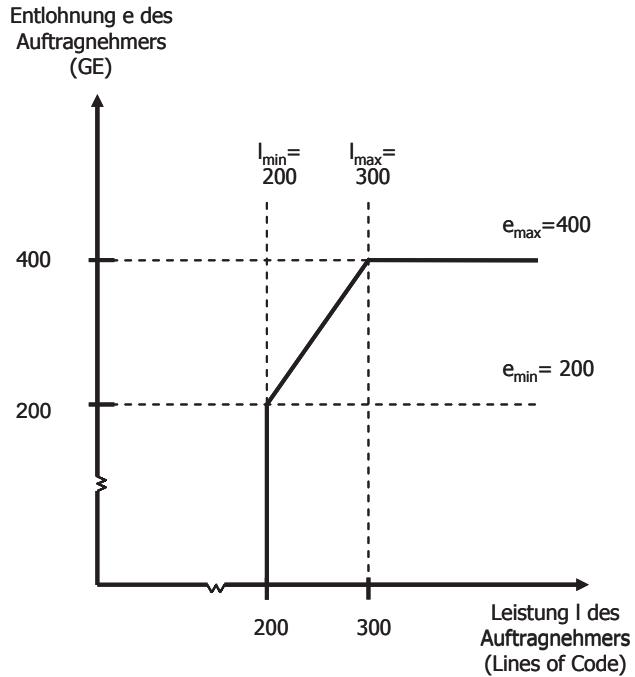
Dazu nehmen wir an, dass die erbrachte Leistung des Anbieters (z. B. die Anzahl der programmierten Zeilen) mit l_i bezeichnet wird. Liegt l_i unter einer vereinbarten Grenze l_{\min} , dann erhält der Auftragnehmer gar keine Entlohnung – quasi als Abschreckung. Im Intervall $[l_{\min}; l_{\max}]$ erhält der Dienstleister die Minimalvergütung e_{\min} sowie einen Anteil am Bonus b , der sich aus $e_{\max} - e_{\min}$ ergibt. Liegt die erbrachte Leistung dagegen über l_{\max} , dann erhält der Dienstleister nur die Maximalvergütung e_{\max} – dies wird ein Dienstleister daher kaum anstreben. Soll der Bonus im Intervall $[l_{\min}; l_{\max}]$ linear mit der erbrachten Leistung steigen, dann berechnet sich die Entlohnung e insgesamt wie folgt:

$$e = \begin{cases} 0 & \text{für } l_i < l_{\min} \\ e_{\min} + \frac{l_i - l_{\min}}{l_{\max} - l_{\min}} \cdot (e_{\max} - e_{\min}) & \text{für } l_{\min} \leq l_i \leq l_{\max} \\ e_{\max} & \text{für } l_i > l_{\max} \end{cases}$$

Für die praktische Umsetzung ist entscheidend, ob es gelingt, die Leistung l_i objektiv zu messen. Bei der Softwareentwicklung könnte man die Leistung etwa am erstellten Code, im Netzbereich anhand von Service-Level-Agreements festmachen.

Auch dieses Schema illustrieren wir an einem kleinen Beispiel. Dabei gehen wir davon aus, dass die Leistung eines Softwarehauses in Lines of Code (LoC) gemessen werden soll. Auch wenn diese Kennzahl natürlich aus verschiedensten Gründen nicht perfekt ist, so kann sie dennoch einen Anhaltspunkt für die Leistungen des Auftragnehmers liefern und wird etwa auch in Modellen wie COCOMO zur Schätzung des Aufwands von Softwareprojekten verwendet.

Abb. 2.18 Beispiel für Entlohnungsschema auf Basis eines Dienstleistungsvertrags



In unserem Beispiel gehen wir davon aus, dass ein beauftragtes Softwarehaus mindestens 200 LoC programmieren soll und für 200 LoC genau 200 GE erhält. Bis 300 LoC steigt die Entlohnung linear bis auf 400 GE an. Unter 200 LoC erhält das Softwarehaus gar keine Entlohnung. Über 300 LoC erhält das Softwarehaus immer 400 GE. Der Auftraggeber setzt damit zwei spezielle Anreize: Mindestens 200 LoC müssen sein (z. B. um den Fortgang des Folgeprojekts nicht zu gefährden) und bis zu 300 LoC wären sehr hilfreich (was sich an der Vergütung pro zusätzlicher LoC im Intervall [200; 300] GE zeigt). Abbildung 2.18 zeigt das Beispiel im Überblick.

Abschließend sei noch einmal auf die grundlegenden Unterschiede zwischen den beiden Entlohnungsschemata hingewiesen. Zwar findet sich in beiden Schemata die Idee der Risikoteilung in einem vorgegebenen Korridor. Entscheidender Unterschied ist jedoch die Bezugsbasis der Entlohnung des Auftragnehmers: Im Fall eines Werkvertrags sind dies die beim Auftragnehmer entstandenen Kosten, im Fall des Dienstleistungsvertrages die von ihm erbrachten Leistungen.

2.5.3 Kontrollsysteme

Neben Entlohnungsschemata werden Kontrollsysteme zur Reduktion der Agency-Kosten eingesetzt. Für die Gestaltung eines Kontrollsystems ist entscheidend, ob der Principal entweder nur das Ergebnis des Agenten oder sein Vorgehen bzw. die Input-Faktoren, wie

Abb. 2.19 Varianten der Ausgestaltung eines Kontrollsyste ms. (in Anlehnung an Hess und Schumann 1999, S. 360)

		Kontrolle des Vorgehens	Kontrolle des Vorgehens mit Fokus auf Input
		Kontrolle der Ergebnisse, ggf. mit Offenlegung	Kontrolle der Ergebnisse mit Fokus auf Input
Beobachtbarkeit durch Principal	gut möglich	hoch	gering
	schlecht möglich		
Beeinflussbarkeit durch Agent			

etwa die zu Grunde liegende Entwicklungsmethodik, kontrolliert. Forschungsergebnisse zur effizienten Gestaltung von Kontrollsyste ms zeigen, dass die Beeinflussbarkeit durch den Agent und die Beobachtbarkeit durch den Principal die zentralen Größen sind, die über die Auswahl eines adäquaten Kontrollsyste ms entscheiden (Ouchi 1977; Picot 1989). Abbildung 2.19 zeigt die grundlegende Logik.

Aus Abb. 2.19 lässt sich eine generelle Empfehlung für die Softwareentwicklung ableiten. Im Normalfall hat der Dienstleister eine Vielzahl von Möglichkeiten, Art und Umfang des Einsatzes von Arbeitskräften zu steuern. So kann ein Individualsoftwareanbieter beispielsweise unterschiedliche Projektleiter und Programmierer für ein Projekt einsetzen. Je nach Erfahrung und Ausbildung dieser Projektmitarbeiter sind Ergebnisse unterschiedlicher Qualität zu erwarten. Die Beeinflussbarkeit durch den Agent ist in diesem Fall hoch, wodurch bildlich gesprochen die linke Spalte in Abb. 2.19 relevant wird. Gleichwohl ist es für den Principal im Regelfall schwierig, die Anstrengungen des Agent zu beobachten und zu bewerten. Selbst wenn der Auftraggeber, weil er etwa aus der IT-Abteilung kommt, den Prozess der Softwareentwicklung grundsätzlich versteht, liegen ihm nur selten wirklich zuverlässige Informationen über den konkreten Status seines Auftrags vor. Im Endeffekt bleibt ihm daher in vielen Fällen nur übrig, sich auf die Kontrolle des Ergebnisses zu beschränken und gegebenenfalls dem Auftragnehmer zu drohen, seine Leistung in der Fachöffentlichkeit bekannt zu machen – gelegentlich wird die Reputation daher als „Pfand“ des Auftraggebers bezeichnet.

Die vorgestellten Überlegungen zur Begrenzung der Agency-Kosten beruhen auf einer Reihe vereinfachender Annahmen. So sind wir bei der Formulierung des Principal-Agent-Problems, sei es mit dem Softwareunternehmen als Auftragnehmer oder als Auftraggeber für Sub-Aufträge, davon ausgegangen, dass Software von einem Auftraggeber einmal spezifiziert und dann von einem Auftragnehmer auf dieser Basis realisiert wird. Dies ist eher ein Sonderfall als die Regel. Vielmehr ist davon auszugehen, dass bei der Entwicklung einer Software das beauftragende Unternehmen vielfach und zu verschiedenen Zeitpunkten in den Entwicklungsprozess einbezogen werden muss. Erschwerend kann noch

hinzukommen, dass die Auftragsvergabe und das erforderliche Know-how in dem beauftragenden Unternehmen an unterschiedlichen Stellen in der Organisation, typischerweise in der IT-Abteilung und in der Fachabteilung, angesiedelt sind, die keineswegs gleiche Interessen haben müssen. In Folge entstehen in zweifacher Hinsicht also weitaus komplexere Principal-Agent-Beziehungen als jene, die wir vorgestellt haben.

Literatur

- Armstrong M (2006) Competition in two-sided markets. *Rand J Econ* 37:668–691
- Arthur WB (1989) Competing technologies, increasing returns, and lock-in by historical events. *Econ J* 99:116–131
- Baligh H, Richartz L (1967) Vertical market structures. Allyn and Bacon, Boston
- Bansler JP, Havn EC (2002) Exploring the role of network effects in IT implementation: the case of knowledge management systems. In: Proceedings of the 10th European Conference on Information Systems, Information Systems and the Future of the Digital Economy. Gdansk, Poland, S 817–829
- Bauer S, Stickel E (1998) Auswirkungen der Informationstechnologie auf die Entstehung kooperativer Netzwerkorganisationen. *Wirtschaftsinformatik* 40:434–442
- Besen SM, Farrell J (1994) Choosing how to compete: strategies and tactics in standardization. *J Econ Perspect* 8:117–131
- Buxmann P (2002) Strategien von Standardsoftware-Anbietern: Eine Analyse auf Basis von Netzeffekten. *Z betriebswirtschaftliche Forsch* 54:442–457
- Buxmann P, Schade S (2007) Wieviel Standardisierung ist optimal? Eine analytische und simulative Untersuchung aus Anwenderperspektive. In: Blum U, Eckstein A (Hrsg) *Wirtschaftsinformatik im Fokus der modernen Wissensökonomik – Festschrift für Prof Dr Dr hc Wolfgang Uhr*. TUDpress, Dresden, S 67–88
- Buxmann P, Weitzel T, König W (1999) Auswirkung alternativer Koordinationsmechanismen auf die Auswahl von Kommunikationsstandards. *Z Betriebswirtschaft Ergänzungsheft* 2:133–151
- Buxmann P, Wüstner E, Kunze S (2005) Wird XML/EDI traditionelles EDI ablösen? Eine Analyse auf Basis von Netzeffekten und einer empirischen Untersuchung. *Wirtschaftsinformatik* 47:413–421
- Clemons EK, Reddi SP, Row MC (1993) The impact of information technology on the organization of economic activity – the „Move to the Middle“ hypothese. *J Manag Inf Syst* 10:9–35
- David PA (1985) Clio and the economics of qwerty. *Am Econ Rev* 75:332–337
- Domschke W, Wagner B (2005) Models and methods for standardization problems. *Eur J Oper Res* 162:713–726
- ESA (2013) Essential facts about the computer and game industry. http://www.theesa.com/facts/pdfs/esa_ef_2013.pdf
- Farrell J, Saloner G (1985) Standardization, compatibility, and innovation. *Rand J Econ* 16:70–78
- Farrell J, Saloner G (1987) Competition, compatibility, and standards: the economics of horses, penguins and lemmings. In: Gabel HL (Hrsg) *Product standardization and competitive strategy*. North Holland, Amsterdam, S 1–22
- Gartner (2013a) Gartner says worldwide video game market to total \$ 93 Billion in 2013. <http://www.gartner.com/newsroom/id/2614915>
- Heinrich B, Klier M, Bewernik M (2006) Unternehmensweite Anwendungsintegration – Zentrale Anreizsetzung zur Realisierung von Netzeffekten bei dezentralen Entscheidungsstrukturen. *Wirtschaftsinformatik* 48:158–168

- Hess T, Schumann M (1999) Medienunternehmen im digitalen Zeitalter – Neue Technologien – Neue Märkte – Neue Geschäftsansätze. Gabler, Wiesbaden
- Hess T, Ünlü V (2004) Systeme für das Management digitaler Rechte. Wirtschaftsinformatik 46:273–280
- Jensen MC, Meckling WH (1976) Theory of the firm: managerial behaviour, agency costs and ownership structure. *J Financ Econ* 3:305–360
- Katz M, Shapiro C (1985) Network externalities, competition, and compatibility. *Am Econ Rev* 75:424–440
- Liebowitz SJ, Margolis SE (1994) Network externality: an uncommon tragedy. *J Econ Perspect* 8:133–150
- Martin-Jung H (2012) Aufholjagd im Schatten der Revolution. <http://www.sueddeutsche.de/digital/computerspielmarkt-im-wandel-aufholjagd-im-schatten-der-revolution-1.1444289>
- Ouchi WG (1977) The relationship between organizational structure and organizational control. *Adm Sci Q* 3:95–113
- Picot A (1989) Zur Bedeutung allgemeiner Theorieansätze für die betriebswirtschaftliche Information und Kommunikation. In: Kirsch W, Picot A (Hrsg) Die Betriebswirtschaftslehre im Spannungsfeld zwischen Generalisierung und Spezialisierung. Gabler, Wiesbaden, S 361–379
- Picot A, Ertsey B (2004) IT-Management der Zukunft als Vertragsmanagement. *Inf Manag Consult* 19:11–19
- Picot A, Meier M (1992) Analyse- und Gestaltungskonzepte für das Outsourcing. *Inf Manag* 7(4):14–27
- Plattner H (2007) Trends and concepts, lecture. http://epic.hpi.uni-potsdam.de/Home/TrendsAndConcepts_I_2007
- Ross SA (1973) The economic theory of agency: the principal's problem. *Am Econ Rev* 63:134–139
- Schade S, Buxmann P (2005) A prototype to analyse and support standardization decisions. In: Egyedi TM, Sherif MH (Hrsg) Proceedings of the 4th International Conference on Standardization and Innovation in Information Technology, 21–23 Sept 2005. ITU Geneva, Switzerland, S 207–219
- Shapiro C, Varian HR (1998) Information rules: a strategic guide to the network economy. Harvard Business School Press, Boston
- Spence AM, Zeckhauser RJ (1971) Insurance, information and individual action. *Am Econ Rev* 61:380–391
- Statista (2014a) Ranking der zehn größten Game Publisher weltweit nach ihrem Umsatz im Jahr 2013 (in Milliarden Euro). <http://de.statista.com/statistik/daten/studie/194749/umfrage/die-10-groessten-games-publisher-weltweit/>
- Weitzel T (2004) Economics of standards in information networks. Physica, Heidelberg
- Weitzel T, Wendt O, von Westarp F (2000) Reconsidering network effect theory. In: Proceedings of the 8th European Conference on Information Systems (ECIS 2000), Wien, S 484–491
- Williamson OE (1985) The economic institutions of capitalism. Firms, markets, relational contracting. Free Press, New York
- Williamson OE (1991) Comparative economic organization: the analysis of discrete structural alternatives. *Adm Sci Q* 36:269–296
- Wüstner E (2005): Standardisierung und Konvertierung: Ökonomische Bewertung und Anwendung am Beispiel von XML/EDI. Shaker, Aachen

Vor dem Hintergrund der dargestellten ökonomischen Prinzipien untersuchen wir in diesem Kapitel ausgewählte Strategien für Softwareanbieter. Grundlegend für ein Unternehmen ist seine Positionierung in der Wertschöpfungskette. In Abschn. 3.1 gehen wir zunächst auf Möglichkeiten und Herausforderungen von Kooperationsstrategien ein. In diesem Kontext behandeln wir auch Unternehmensübernahmen, die auf Softwaremärkten eine besondere Rolle spielen. Darauf aufbauend untersuchen wir angebots-seitige Strategien. Darunter verstehen wir die Geschäftspolitik an der Schnittstelle zu Vertriebspartnern und Kunden, also den in der Wertschöpfungskette nachgelagerten Geschäftspartnern. In diesem Bereich werden Vertriebs- (Abschn. 3.2) und Preisstrategien (Abschn. 3.3) thematisiert. Abschließend betrachten wir in Abschn. 3.4 zentrale Managementfragen bei der Entwicklung von Software.

3.1 Kooperations- und Übernahmestrategien

In diesem Abschnitt sollen nun Kooperations- und Übernahmestrategien für die Softwareindustrie betrachtet werden. Diese Strategien sind – wie in Abschn. 2.2 bereits beschrieben – insbesondere vor dem Hintergrund der Existenz von Netzeffekten auf Softwaremärkten von zentraler Bedeutung. Darüber hinaus wird ein Anbieter in der Regel nicht in der Lage sein, alleine mit seinen Produkten und Dienstleistungen die Bedürfnisse der Kunden zu befriedigen. Zunächst wollen wir allgemeine Vorteile und Herausforderungen von Kooperationen beleuchten (Abschn. 3.1.1). Im Anschluss daran gehen wir auf Unternehmensübernahmen in der Softwareindustrie ein (Abschn. 3.1.2).

3.1.1 Kooperationen in der Softwareindustrie

3.1.1.1 Vorteile von Kooperationen

Unter Kooperationen verstehen wir im Folgenden eine auf stillschweigenden oder vertraglichen Vereinbarungen beruhende Zusammenarbeit zwischen rechtlich weiterhin selbstständigen Unternehmen (Blohm 1980, S. 1112). Dabei gehen wir davon aus, dass Kooperationen für eine mittel- oder langfristige Zusammenarbeit geschlossen werden und Investitionen seitens der teilnehmenden Unternehmen erfordern. Die Ziele bestehen grundsätzlich darin, Effizienzvorteile oder Mehrwerte zu generieren, die ohne die Kooperation nicht entstanden wären. Adam Brandenburger und Barry Nalebuff sprechen daher in diesem Zusammenhang auch von einem „Value Net“, das durch die Zusammenarbeit der Unternehmen entsteht (Brandenburger und Nalebuff 1996, S. 16–19). Hierbei können für die Beteiligten u. a. die folgenden Vorteile erzielt werden:

- *Kosteneinsparungen* lassen sich durch Skaleneffekte – hierzu gehören Economies of Scale sowie Economies of Scope – realisieren. Einfache Beispiele hierfür sind Kostenvorteile durch Einkaufsrabatte bei einer Abnahme großer Mengen oder auch die gemeinsame Nutzung von Ressourcen, z. B. Lagern oder Räumen. Darüber hinaus können durch kooperative Planungsprozesse, etwa bei der Beschaffungs- oder Tourenplanung, Kosten gesenkt werden (Martín Díaz 2006).
- *Zeiteinsparungen* lassen sich z. B. im Rahmen von Entwicklungsprojekten durch eine Zusammenlegung von Ressourcen realisieren. Auf diese Weise kann schließlich die so genannte „time to market“ verkürzt werden.
- Darüber hinaus können solche Entwicklungspartnerschaften zu einer *Reduktion von Risiken* führen. So können durch eine Aufteilung der Entwicklungsaufwendungen die Risiken eines Fehlschlags geteilt und für die jeweiligen Partner gemindert werden.
- Zudem können Kooperationen zu einem höheren *Wert des Produktes oder Services* führen. Dies erfolgt etwa dadurch, dass Allianzen von Fluggesellschaften, Autovermietungen und Hotels gemeinsam zusätzliche Leistungen anbieten, wie z. B. eine abgestimmte Aus- und Rückgabe der Leihwagen und die Verrechnung von Bonuspunkten. Auch Open Source Software wird im Rahmen von Kooperationen entwickelt. Je mehr Programmierer sich an der Entwicklung beteiligen, umso besser wird tendenziell auch die Software. Eric S. Raymond drückt es in seinem berühmten Artikel „The Cathedral and the Bazaar“ so aus: „Given enough eyeballs all bugs are shallow“ (Raymond 1999).
- Schließlich können Kooperationen sowie Unternehmensübernahmen den *Zugang zu neuen Märkten* eröffnen. Dies kann zum einen eine geographische Ausweitung, zum anderen auch eine Erweiterung der Produktpalette bedeuten.

Die potenziellen Kooperationsvorteile schlagen sich letztlich in Kosteneinsparungen oder (auf direktem bzw. auch indirektem Wege) in einer Steigerung der Erlöse nieder. Eine Herausforderung besteht nun darin, diesen Mehrwert der Kooperation zwischen den Partnern aufzuteilen. Viele Kooperationsvorhaben scheitern an dieser Problemstellung, die nur auf

den ersten Blick trivial erscheint, bereits im Vorfeld. Dies wollen wir anhand eines Beispiels aus der kooperativen Spieltheorie veranschaulichen: dem Bettler-Krösus-Problem.

Ein reicher Mann und ein Bettler laufen die Straße entlang und finden gleichzeitig einen Geldbetrag von – sagen wir – 100 €. Die Herausforderung für die beiden besteht nun darin, sich auf eine Aufteilung des Funds zu einigen, mit der beide zufrieden sind. Gelingt dies, darf jeder seinen Anteil an den 100 € behalten, andernfalls gehen beide leer aus.

Das Problem ist einfach zu verstehen, eine einvernehmliche Lösung jedoch schwierig zu erreichen. Denn sowohl der Bettler als auch der Krösus möchten so viel wie möglich des Geldes für sich behalten. Diese Konstellation lässt sich analog in vielen Fällen bei Geschäftspartnern beobachten. Die spieltheoretische Lösung des Verteilungsproblems ist mathematisch komplex und basiert auf unterschiedlichen Nutzenfunktionen für die Akteure (Sieg 2005, S. 181 ff.). Auf der Basis einer angenommenen linearen Nutzenfunktion für den reichen Mann und einer logarithmischen für den Bettler lässt sich für unser Beispiel eine Aufteilung von ca. 23 € für den Bettler und ca. 77 € für den reichen Mann ermitteln. Die Ermittlung und mathematische Formulierung von Nutzenfunktionen ist in der Praxis jedoch nahezu unmöglich.

Im Folgenden wollen wir daher die Verteilungsproblematik vereinfachen: Zunächst können wir davon ausgehen, dass die Aufteilung der Kooperationsgewinne das Kriterium der Pareto-Optimalität erfüllen muss. Damit ist gemeint, dass die Kooperationsgewinne so zu verteilen sind, dass mindestens einer der Partner besser, aber keiner schlechter gestellt ist als zuvor. Ist dieses Kriterium nicht erfüllt, wird mindestens einer der Partner in der Regel nicht an der Kooperation teilnehmen.

Es können insbesondere die folgenden einfachen Vorgehensweisen pragmatische Alternativen zur Aufteilung der Kooperationsgewinne darstellen (Buxmann et al. 2007):

- Der Kooperationsgewinn wird unter n Akteuren so verteilt, dass jeder Akteur den n -ten Anteil dieses zusätzlichen Gewinns erhält.
- Der Kooperationsgewinn wird entsprechend der Gewinnanteile vor der Kooperation verteilt.

Dabei profitieren die in der Ausgangsposition kleineren Partner überproportional vom erst genannten Verteilungsmodell. Bei der zweiten Variante wird demgegenüber tendenziell der in der Ausgangssituation stärkere Akteur bevorzugt. Auch wenn beide Alternativen zu einer pareto-optimalen Verteilung führen, ist damit also noch lange nicht gesagt, dass sich die Akteure tatsächlich auch einigen werden. Verschiedene Verteilungsschlüssel können eben entsprechend unterschiedlich vorteilhaft sein und jeder wird versuchen, ein möglichst großes Stück des zu verteilenden Kuchens zu erhalten.

Neben der Fragestellung der Aufteilung der Kooperationsgewinne ist zu berücksichtigen, dass das Eingehen von Kooperationen für die beteiligten Unternehmen häufig eine nicht zu unterschätzende Investition darstellt. So entstehen zum einen Anbahnungskosten für die Suche nach den richtigen Geschäftspartnern. Daneben fallen in der Regel erhebliche Verhandlungskosten für die Vertragsgestaltung an. Diese haben etwa die Beteiligun-

gen an den Investitionen in die Partnerschaft und auch die Ausgestaltung des Schlüssels zur Verteilung der Kooperationsgewinne zum Gegenstand. Nicht zu unterschätzen sind auch die Investitionen zum Aufbau einer gemeinsamen Infrastruktur oder zur Erhöhung des Know-hows in den beteiligten Unternehmen (Hirnle und Hess 2007).

Im nächsten Abschnitt wollen wir uns nun auf die Kooperationsformen und -partner in der Softwareindustrie konzentrieren.

3.1.1.2 Kooperationsformen und -partner in der Softwareindustrie

Wenden wir uns zunächst der Fragestellung zu, welche potenziellen Kooperationspartner für Softwarehäuser in Frage kommen. Dazu wollen wir uns im Weiteren an dem Modell von Brandenburger und Nalebuff orientieren, das in Abb. 3.1 dargestellt ist (Brandenburger und Nalebuff 1996, S. 17).

Wir gehen also davon aus, dass ein Unternehmen – sei es aus der Automobilbranche, der Softwareindustrie oder einem anderen Sektor – grundsätzlich die vier oben dargestellten potenziellen Kooperationspartner hat: Kunden, Komplementärantenbieter, Zulieferer sowie Wettbewerber.

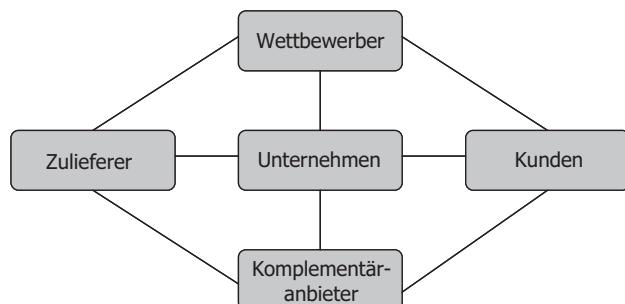
Zur näheren Untersuchung von Partnerschaften in der Softwareindustrie bauen wir im Weiteren auf der Klassifikation von Ralf Meyer auf und unterscheiden zwischen (Meyer 2008):

- Entwicklungspartnerschaften,
- Reseller-Partnerschaften,
- Revenue-Share-Partnerschaften,
- OEM-Partnerschaften,
- Referral-Partnerschaften sowie
- Standardisierungs-Partnerschaften.

Dabei sind bei der Ausgestaltung der konkreten Kooperationsbeziehungen grundsätzlich die folgenden Fragen zu beantworten (Meyer 2008):

- Welcher Partner liefert das Produkt an den Endkunden (Who ships)?
- Welcher Partner entscheidet über die Preissetzung und Rabatte (Pricing)?

Abb. 3.1 Systematisierung potenzieller Kooperationspartner



- Unter welcher Marke erscheint das Produkt beim Endkunden (Branding)?
- Welcher Partner hat welche Eigentumsrechte am Produkt (IP)?
- Auf wessen Rechnung wird das Produkt verkauft (Revenue Booking)?
- Welcher Partner ist für den Kundenkontakt verantwortlich (Customer Control)?
- Welcher Partner ist für den Marktauftritt verantwortlich (Go-To-Market)?
- Nimmt der Abnehmer Qualitätskontrollen vor (Quality Assurance)?
- Welcher Partner ist für den Support verantwortlich (Support by)?

Entwicklungspartnerschaften Unter einer Entwicklungspartnerschaft wird die gemeinsame Entwicklung von neuen Software- und Servicelösungen verstanden. Dabei können Softwareanbieter solche Kooperationen grundsätzlich mit allen der in Abb. 3.1 dargestellten Partnern eingehen. Die beteiligten Partner teilen die Entwicklungsaufgaben untereinander auf und nehmen damit die Rolle von Inventors ein.

Ein häufig anzutreffendes Beispiel ist die gemeinsame Produkt- und Systementwicklung zwischen Standardsoftwareanbietern und Kunden. Hierbei geht es meist darum, eine Lösung zu entwickeln, die bestimmte Branchenanforderungen abdeckt, welche in einer Standardlösung nicht abgebildet sind. Doch worin bestehen nun die Vorteile der Kooperation? Der Kunde erhält eine Softwarelösung, die speziell für seine Anforderungen entwickelt wurde. Dafür erwirbt der Softwareanbieter im Rahmen der Kooperation häufig erforderliches Branchen-Know-How. Ein Beispiel für eine solche Zusammenarbeit ist die Entwicklung einer Lösung für eine kooperative Lieferplanabwicklung, die in einer Kooperation von SAP und Bosch entstanden ist (Buxmann et al. 2004).

Nicht selten führt, wie in dem folgenden Beispiel dargestellt, eine gemeinsame Softwareentwicklung auch zu einer Gründung von Joint Ventures.

iBS Banking Solution

Im Rahmen eines Individualprojekts hat CSC Ploenzke für die DePfa Bank (heutige Areal Bank) eine Inhouse-Lösung für den Hypothekenbereich entwickelt. Die Lösung erweitert den SAP-Standard für Banken um wichtige Komponenten. Funktional umfasst die Mortgage Banking Solution das gesamte Aktiv- und Passivgeschäft, integrierte Derivate sowie den Geld- und Devisenhandel und bildet die Grundlage der Gesamtbanksteuerung sowie des Risikomanagements auf Basis des SAP-Systems. Nach Projektabschluss waren die Beteiligten der Ansicht, dass die Softwarelösung allgemein vermarktungsfähig war. Insbesondere die folgenden beiden Gründe sprachen für die Gründung eines Joint Ventures, an dem CSC Ploenzke mit 51 % und die DePfa mit 49 % beteiligt war: Zum einen schien es problematisch zu sein, dass die potenziellen Kunden genau die Wettbewerber der DePfa waren. Zum anderen verfügte die DePfa nicht über ausreichende Beratungskapazitäten, um die Software bei Wettbewerbern zu implementieren. Inzwischen ist die DePfa bzw.

Areal Bank nicht mehr an iBS Banking Solutions beteiligt, seit 2011 hält die SKS Group 49 % am Unternehmen.

Quellen: www.ibs-banking.com; *Sapinfo.net/SAP-Branchenmagazin Banken & Versicherungen, Nr. 3 März 2001, S. 20–21;* *SKS Group (2011)* [http://www.sks-group.eu/aktuelles-detail.html?&tx_ttnews\[tt_news\]=83&cHash=802f207145866de1acd1545b27af3120](http://www.sks-group.eu/aktuelles-detail.html?&tx_ttnews[tt_news]=83&cHash=802f207145866de1acd1545b27af3120)

Eine weitere Gruppe potenzieller Kooperationspartner im Rahmen gemeinsamer Entwicklungsaufgaben sind Zulieferer. Dabei handelt es sich für die Softwareindustrie insbesondere um andere Softwarehäuser, die Teile der Entwicklung übernehmen, oder auch um freie Mitarbeiter, die in Projekten mitarbeiten. So arbeiten gerade große Standardsoftwarehersteller mit einer Vielzahl von Softwarezulieferern zusammen. Speziell der Trend in Richtung der Etablierung von serviceorientierten Architekturen kann einen Beitrag dazu leisten, dass an dieser Schnittstelle zwischen Softwareanbietern und Zulieferern weitere Kooperationen entstehen. So ergeben sich etwa für Nischenanbieter neue Chancen, Software als Service anzubieten (Kude et al. 2012).

Wenn für einen Standardsoftwareanbieter die Kosten für die Entwicklung eines bestimmten Services den für ihn erwarteten Zuwachsnutzen übersteigen, so bietet es sich an, diese Entwicklung an einen Softwarezulieferer auszulagern. Damit hat der Softwareanbieter sein Entwicklungsrisiko begrenzt und der Zulieferer erhält die Chance, seine Services in die Lösung des großen Anbieters zu integrieren. Bislang funktioniert die Zusammenarbeit in der Praxis in den meisten Fällen so, dass Anbieter und Zulieferer vor dem Abnehmer der Lösung getrennt auftreten und auch fakturieren. Zukünftig sind hier engere Kooperationen denkbar und sinnvoll. Die Zusammenarbeit könnte etwa so geregelt werden, dass die Zulieferer auch an den Umsätzen partizipieren. Eine wesentliche Herausforderung bei dem Eingehen solcher Kooperationen wird jedoch, wie bereits dargestellt, darin bestehen, entsprechende Schlüssel für die Aufteilung der Erlöse zu finden. Zudem müssen beide Parteien in die Zusammenarbeit investieren, wobei das größere Investment typischerweise bei den Zulieferern liegt und insbesondere in Form von Schulungskosten anfällt. So müssen Zulieferer bzw. Entwicklungspartner häufig an – nicht selten teuren – Trainings der Softwareanbieter teilnehmen, um in bestimmte Partnerprogramme aufgenommen zu werden. Ein aus Sicht der Anbieter verständliches Verfahren: Zum einen kann auf diese Weise sichergestellt werden, dass die Zulieferer die entsprechenden Basistechnologien kennen, zum anderen werden zum Teil erhebliche Schulungsumsätze generiert.

Darüber hinaus sind solche Partnerschaften auch mit Wettbewerbern denkbar. Diese Kooperationsform wird auch mit dem Begriff „Co-opetition“ beschrieben, denn: „You have to cooperate and compete at the same time“ (Brandenburger und Nalebuff 1996, S. 4). Die Herausforderung besteht darin, eine Win-Win-Situation zu erzeugen, in der die Kooperationspartner von einer Zusammenarbeit profitieren, obwohl sie Wettbewerber sind und auch in Zukunft bleiben.

Als potenzielle Vorteile einer Co-opetition können im Wesentlichen genau jene identifiziert werden, die wir bereits in Abschn. 3.1.1.1 angeführt haben. Eine offensichtliche

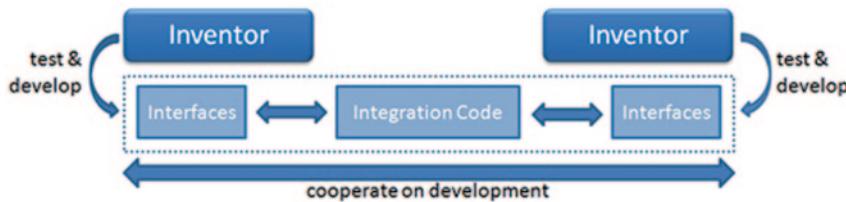


Abb. 3.2 Überblick über die gemeinschaftliche Entwicklung von Integrationskomponenten. (in Anlehnung an Meyer 2008, S. 110)

Kooperationsmöglichkeit besteht immer dann, wenn es darum geht, Ressourcen gemeinsam zu nutzen. Sind etwa Entwicklungs- oder Produktionskapazitäten von mindestens einem der Beteiligten nicht voll ausgelastet, während der Wettbewerber an der Kapazitätsgrenze arbeitet, bietet sich ein „Ressourcen-Sharing“ an. Insbesondere in der Automobilindustrie sind derartige Kooperationsformen bekannt geworden. So kooperieren beispielsweise Daimler und Volkswagen seit Langem bei der Entwicklung von Motoren und Nutzfahrzeugen, während Porsche und Toyota bei der Entwicklung von Hybridantrieben eng zusammenarbeiten.

Ein Beispiel für eine Kooperation zwischen Softwareanbietern, die zumindest teilweise im Wettbewerb stehen, ist die von Microsoft und SAP vorangetriebene Entwicklung von DUET. Dabei stellt DUET dem Anwender eine Schnittstelle zwischen den Office-Applikationen von Microsoft und den ERP-Systemen von SAP bereit. Microsoft und SAP sind Wettbewerber auf dem Markt für betriebswirtschaftliche Software für kleine und mittelständische Unternehmen und erhoffen sich eine Win-Win-Situation durch die Entwicklung dieser Softwarelösung.

Die Vergütung und Verteilung der Intellectual Property (IP) unter den Beteiligten kann von Fall zu Fall sehr unterschiedlich sein. In Abb. 3.2 ist beispielhaft eine Ausprägung einer Entwicklungspartnerschaft aufgeführt, die vor allem im SAP-Ökosystem häufig anzutreffen ist. Es handelt sich dabei um die gemeinschaftliche Entwicklung von Integrationskomponenten.

Abbildung 3.3 zeigt eine Entwicklungspartnerschaft am Beispiel der Integration von Partnerprodukten in eine SAP-Lösung.

Who ships	Pricing	Branding	IP ¹	Revenue Booking	Customer Control	GTM ²	QA ³	Support by
Integr.-partner ⁴	Integr.-partner	no co-branding	no joint IP	Integr.-partner	Integr.-partner	Integr.-partner	Integr.-partner	Integr.-partner
SAP	SAP	no co-branding	no joint IP	SAP	SAP	SAP	SAP	SAP

¹ IP = Intellectual Property ² GTM = Go-to-market ³ QA = Quality Assurance

⁴ Integr.partner = Integration partner bzw. Integrationspartner

Abb. 3.3 Entwicklungspartnerschaft – Parameter der Ausgestaltung (am Beispiel einer SAP Integration). (in Anlehnung an Meyer 2008, S. 76)

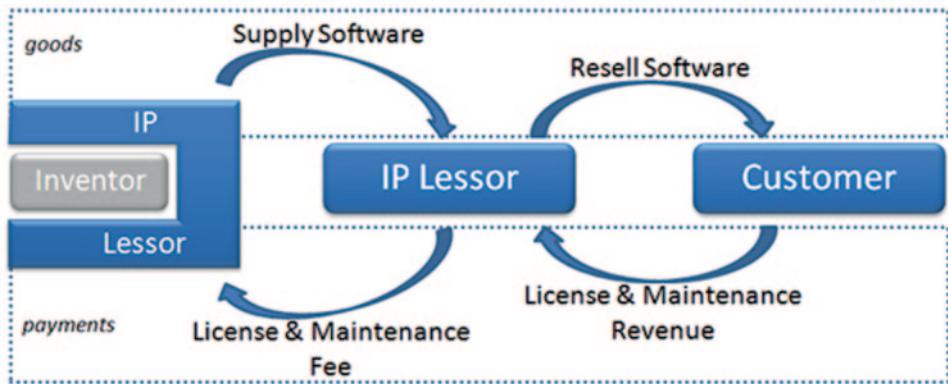


Abb. 3.4 Überblick über die Reseller-Beziehung. (in Anlehnung an Meyer 2008, S. 78)

Reseller-Beziehungen Dieses Modell ist durch eine Anbieter-Abnehmer-Beziehung gekennzeichnet. Das bedeutet in diesem Fall, dass ein Abnehmer die Lösungen eines Anbieters in sein Produktprogramm aufnimmt und an seine Kunden weiterverkauft. Der Anbieter erstellt also ein Softwareprodukt und verkauft Lizenzen an den Abnehmer, d. h., er übernimmt die doppelte Rolle eines Inventors und – aus Sicht des Abnehmers – eines IP Lessors. Der Abnehmer selbst übernimmt ebenfalls die Rolle des IP Lessors, da er die erworbenen Lizenzen weiterverkauft. Die Reseller-Beziehung ist überblicksartig in Abb. 3.4 dargestellt.

Die Reseller-Beziehung ist häufig zwischen Softwareanbietern und Vertriebspartnern anzutreffen. Aber sie kann ebenso zwischen Lieferanten und Softwareanbietern auftreten.

Potenzielle Vorteile für Zulieferer bestehen darin, dass sie Zugang zu den Kunden des Abnehmers erhalten, ohne hierfür Vertriebskapazitäten aufzubauen zu müssen. Es entstehen zusätzliche Kosten für die Koordination der Partnerschaft, die möglicherweise durch die zusätzlichen Umsätze über den Vertriebskanal überkompensiert werden.

Typischerweise wird die Reseller-Beziehung wie in Abb. 3.5 dargestellt ausgestaltet:

Revenue Share-Beziehungen Bei diesem Partnermodell vertreibt ein Softwareanbieter seine Produkte über einen Broker, bei dem es sich beispielsweise um eine Plattform handeln kann. Der Broker stellt hier vor allem die eigene Marktposition im Sinne eines gemeinsamen Marktauftritts zur Verfügung und übernimmt im Gegensatz zum Resell

Who ships	Pricing	Branding	IP ¹	Revenue Booking	Customer Control	GTM ²	QA ³	Support by
Resell partner	Resell partner	both or IP Distr.	no joint IP	Resell partner	Resell partner	Resell partner	Resell partner	Level1,2: Resell partner

¹ IP = Intellectual Property

² GTM = Go-to-market

³ QA = Quality Assurance

Abb. 3.5 Reselling – Parameter der Ausgestaltung. (in Anlehnung an Meyer 2008, S. 76)

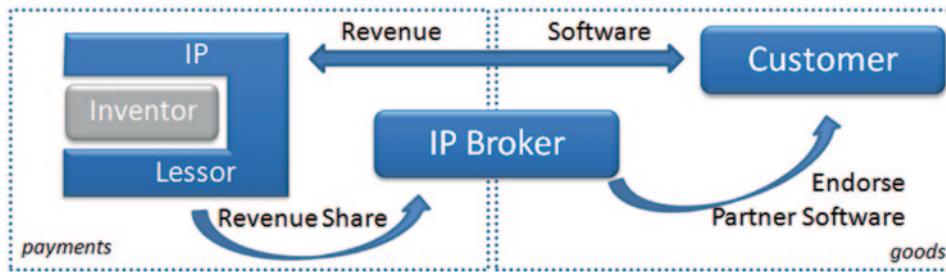


Abb. 3.6 Überblick über die Revenue-Share-Beziehung. (in Anlehnung an Meyer 2008, S. 86)

keine Eigentumsrechte am Softwareprodukt selbst. Er empfiehlt somit lediglich die externe Lösung an eigene Kunden weiter und tritt somit als IP Broker auf. Die Broker-Beziehung ist in Abb. 3.6 dargestellt.

Ein aktuelles Beispiel für ein solches Kooperationsmodell ist die Softwaredistribution über mobile Portale. Hierbei stellen Softwareanbieter, bei denen es sich um Softwarefirmen, aber auch um einzelne Entwickler handeln kann, über eine solche Plattform Softwarelösungen bzw. Apps für die Kunden zur Verfügung. Die bekanntesten Beispiele sind der Android Markt von Google sowie der AppStore von Apple.

Gemäß der oben vorgestellten Systematik wird eine Revenue-Share-Beziehung häufig wie folgt ausgestaltet (siehe Abb. 3.7).

OEM-Beziehung Bei dem OEM-Modell handelt es sich um die klassische Zulieferer-Abnehmer-Beziehung, wie sie beispielsweise in der Automobilindustrie in der Regel anzutreffen ist. Das bedeutet, dass ein Unternehmen von Zulieferern entwickelte Komponenten oder ganze Teilsysteme in die eigene Lösung integriert. In der Softwareindustrie treten bei einer OEM-Beziehung sowohl Zulieferer als auch Abnehmer als Inventor und IP Lessor auf (siehe Abb. 3.8). Im Extremfall kann der Abnehmer eine Lösung vollständig aus zugelieferten Teilen zusammenstellen (siehe hierzu auch Abschn. 3.4.3). In diesem Fall beschränkt sich die Tätigkeit des Abnehmers als Inventor lediglich auf die notwendigen Integrationsentwicklungen.

Ein Zulieferer kann bei diesem Modell zum einen von höheren Umsätzen und zum anderen von einer höheren Marktdurchdringung profitieren. Häufig sieht die Partnerbeziehung so aus, dass kleinere Unternehmen größeren Abnehmern ihre Komponenten zur Ver-

Who ships	Pricing	Branding	IP ¹	Revenue Booking	Customer Control	GTM ²	QA ³	Support by
Soft. vendor ⁴	Soft. vendor	Soft. vendor	no joint IP	Soft. vendor	jointly	jointly	Rev. partner ⁵	Soft. vendor

¹ IP = Intellectual Property ² GTM = Go-to-market ³ QA = Quality Assurance

⁴ Soft. vendor = Software vendor bzw. Softwarehersteller ⁵ Rev. partner = Revenue partner

Abb. 3.7 Revenue-Share-Beziehung – Parameter der Ausgestaltung. (in Anlehnung an Meyer 2008, S. 76)

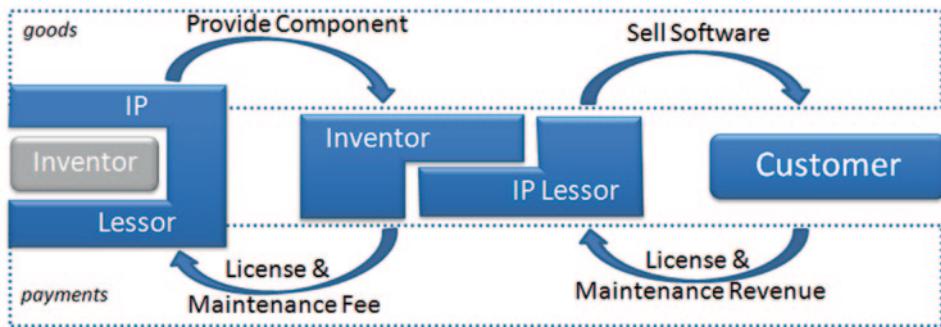


Abb. 3.8 Überblick über die OEM-Beziehung. (in Anlehnung an Meyer 2008, S. 97)

fügung stellen. Aber diese Geschäftsbeziehung kann auch anders aufgebaut sein. So tritt beispielsweise auch die SAP AG häufig als OEM-Partner auf, indem sie ihre NetWeaver Plattform anderen Unternehmen bereitstellt. Diese können auf dieser Basis komplementäre Lösungen entwickeln und als Gesamtpaket an eigene Kunden vertreiben.

Aus Sicht der Abnehmer besteht der zentrale Vorteil des OEM-Modells darin, dass sie durch die bezogenen Komponenten und Lösungen ihre Entwicklungskosten, -risiken und -dauer reduzieren können. Die Kostensenkungen ergeben sich vor allem daraus, dass der Zulieferer seine Produkte in vielen Fällen an mehrere Abnehmer vertreibt und daher die Entwicklungskosten auf viele Parteien umlegen kann.

Die häufige Ausgestaltung einer OEM-Beziehung zeigt Abb. 3.9.

Referral-Beziehung Auch bei der Referral-Beziehung existiert, wie im Revenue-Share-Modell, ein IP Broker, den wir im Weiteren auch als Referralgeber bezeichnen (siehe Abb. 3.10). Dieser verkauft Informationen über potenzielle Kunden (Leads) an den Referralnehmer (IP Lessor). Für diese Information kann der IP Broker eine Kombination aus fixer Entlohnung und einer Beteiligung an resultierenden Umsätzen erhalten. Da der Referralgeber hier meist keinen direkten Einfluss auf seine Kunden nimmt, sondern lediglich die Kontakte vermittelt, ist diese Beteiligung normalerweise deutlich geringer als beim Revenue-Share Modell.

Die Ausgestaltung einer solchen Zusammenarbeit ist in Abb. 3.11 dargestellt.

Who ships	Pricing	Branding	IP ¹	Revenue Booking	Customer Control	GTM ²	QA ³	Support by
Soft. vendor ⁴	Soft. vendor	Soft. vendor	no joint IP	Soft. vendor	jointly	jointly	Rev. partner ⁵	Soft. vendor

¹ IP = Intellectual Property ² GTM = Go-to-market ³ QA = Quality Assurance

⁴ Accept. = Acceptor ⁵ Supp. = Supplier bzw. Zulieferer

Abb. 3.9 OEM-Beziehung – Parameter der Ausgestaltung. (in Anlehnung an Meyer 2008, S. 76)

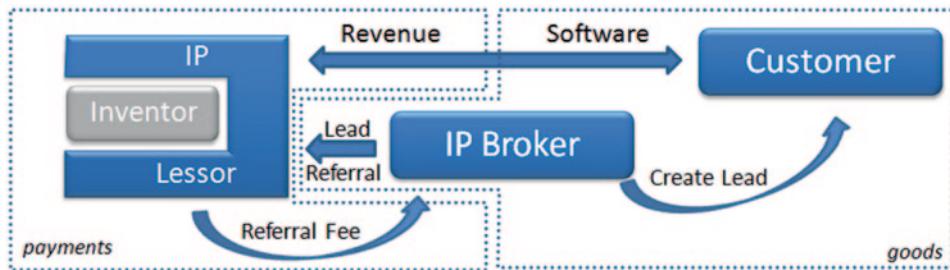


Abb. 3.10 Überblick über die Referral-Beziehung. (in Anlehnung an Meyer 2008, S. 93)

Who ships	Pricing	Branding	IP ¹	Revenue Booking	Customer Control	GTM ²	QA ³	Support by
Referrals-taker ⁴	Referrals-taker	Referrals-taker	Referrals-taker	Referrals-taker	Referrals-taker	Referrals-taker	Referrals-taker	Referrals-taker

¹ IP = Intellectual Property ² GTM = Go-to-market ³ QA = Quality Assurance

⁴ Referrals-taker: Inventor bzw. IP Lessor

Abb. 3.11 Überblick über die Referral-Beziehung. (in Anlehnung an Meyer 2008, S. 76)

Standardisierungspartnerschaften Auf etwas anderer Ebene sind Standardisierungspartnerschaften anzusiedeln, da hier in der Regel keine Finanzflüsse zwischen den Beteiligten stattfinden. Dennoch sind sie ein wesentlicher Baustein in den Strategien vieler Software- und IT-Unternehmen. In diesem Zusammenhang werden häufig Strategische Allianzen eingegangen. Das Ziel besteht in der Regel darin, Standards durchzusetzen oder aber auch genau einen solchen Erfolg zu verhindern, der potenziellen Konkurrenten nützen könnte.

Dabei finden Kooperationen häufig in Arbeitsgruppen von Standardisierungsorganisationen, wie dem World Wide Web Consortium, statt. Das Ziel der Unternehmen besteht einerseits darin, bei der technischen Spezifizierung des zu entwickelnden Standards mitzuwirken. Andererseits sollen damit natürlich möglichst viele Akteure in das sprichwörtliche Boot geholt werden, um den eigenen oder favorisierten Standard, der möglicherweise im Wettbewerb zu anderen steht, durchzusetzen. Auf die Bedeutung der Durchsetzung von Standards sind wir in Abschn. 2.2 dieses Buches bereits ausführlich eingegangen.

Ein Beispiel hierfür ist die Strategische Allianz von Softwareanbietern zur Unterstützung und Durchsetzung des OpenDocument-Formats (ODF), das ein Austauschformat für Office-Dateien darstellt. Dabei haben sich Softwarehäuser, wie IBM, Oracle, Sun Microsystems und Google, zusammengetan, um ein Gegengewicht zu dem insbesondere von Microsoft unterstützten OpenXML-Format zu schaffen. Es folgte ein monatelanger Standardisierungsstreit: Die OpenXML-Gegner argumentierten, dass mit dem konkurrierenden OpenDocument-Format schon seit 2006 ein entsprechender ISO-Standard gegeben sei. Allerdings schaffte Microsoft es im Frühjahr 2008 dennoch, das OpenXML-Format als weiteren ISO-Standard durchzusetzen.

3.1.2 Mergers & Acquisitions in der Softwareindustrie

In diesem Abschnitt des Buches wollen wir uns mit Mergers & Acquisitions bzw. Unternehmensübernahmen befassen. Diese spielen in der Softwareindustrie auch deshalb eine besondere Rolle, weil aufgrund von Netzeffekten die Größe eines Anbieters und seines Netzwerkes einen entscheidenden Wettbewerbsvorteil darstellen kann. Zunächst wollen wir uns mit unterschiedlichen Formen von Unternehmenszusammenschlüssen beschäftigen (Abschn. 3.1.2.1). Darauf aufbauend werden verschiedene Motive für Unternehmensübernahmen untersucht (Abschn. 3.1.2.2), bevor wir Konzentrationstendenzen in der Softwareindustrie (Abschn. 3.1.2.3) betrachten. Schließlich analysieren wir den Erfolg solcher Unternehmensübernahmen in der Softwareindustrie (Abschn. 3.1.2.4).

3.1.2.1 Formen von Unternehmenszusammenschlüssen

Auch für den Bereich Mergers & Acquisitions gibt es eine Vielzahl von Definitionen, auf die wir hier nicht im Einzelnen eingehen wollen. Wir folgen vielmehr den Überlegungen von Wirtz, nach denen Unternehmensübernahmen dadurch gekennzeichnet sind, dass ein beteiligtes Unternehmen mindestens die wirtschaftliche Selbstständigkeit – also gegebenenfalls zusätzlich auch die rechtliche – aufgibt (Wirtz 2003, S. 15).

Unternehmensübernahmen können, wie Abb. 3.12 zeigt, in Akquisitionen einerseits und Mergers andererseits unterschieden werden. Unter einem Merger wird eine Fusion verstanden, die die Verschmelzung rechtlich unabhängiger Unternehmen zu einer rechtlichen Einheit zur Folge hat. Die beteiligten Parteien geben dabei also ihre rechtliche Selbstständigkeit auf, wobei es sich um eine Fusion durch Aufnahme oder Neugründung handeln kann. Unter Akquisition wird hingegen die Eingliederung eines Unternehmens in einen Unternehmensverbund ohne die zwingende rechtliche Verschmelzung verstanden.

Darüber hinaus werden horizontale, vertikale und diagonale bzw. konglomerate Unternehmenszusammenschlüsse unterschieden (Wirtz 2003, S. 18 f.).

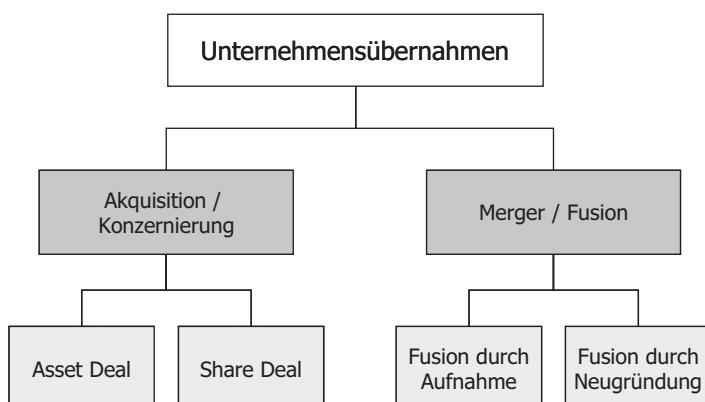


Abb. 3.12 Unternehmensübernahmen. (Wirtz 2003, S. 13)

Von einem horizontalen Zusammenschluss spricht man, wenn Unternehmen derselben Branche auf der gleichen Stufe der Wertschöpfungskette miteinander verschmelzen. Das Bundeskartellamt unterscheidet zudem zwischen horizontalen Zusammenschlüssen mit und ohne Produktausweitung. Ein horizontaler Zusammenschluss zielt in der Regel auf eine Verbesserung der eigenen Wettbewerbsposition sowie auf die Realisierung von Synergieeffekten in Form von Economies of Scale bzw. Economies of Scope ab. Beispiele für horizontale Zusammenschlüsse in der Softwareindustrie sind die Übernahme von Peoplesoft durch Oracle (siehe hierzu Abschn. 3.1.2.3) oder der Kauf des Unternehmens Veritas durch Symantec im Bereich Speicher- und Sicherheitslösungen für rund 13,5 Mrd. \$ (Parbel 2005).

Bei vertikalen Zusammenschlüssen verschmelzen Unternehmen auf unterschiedlichen Stufen einer Wertschöpfungskette. Dabei schließt sich ein Unternehmen mit einem anderen Unternehmen zusammen, das sich auf der vor- bzw. nachgelagerten Wertschöpfungsstufe befindet, was auch als Rückwärts- bzw. Vorwärtsintegration bezeichnet wird. Die Ziele vertikaler Zusammenschlüsse sind die Senkung der Transaktionskosten, die Verbesserung der Planung entlang der Wertschöpfungskette sowie ein besserer Zugang zu Beschaffungs- (im Falle einer Rückwärtsintegration) bzw. Absatzmärkten (im Falle einer Vorwärtsintegration). Ein Beispiel für einen solchen Zusammenschluss ist die Expansion der Software AG in Lateinamerika durch die Übernahme der APS Venezuela sowie fünf Schwesterunternehmen in Panama, Costa Rica und Puerto Rico im Jahr 2005. APS Venezuela war bis zu diesem Zeitpunkt Vertriebspartner für Produkte der Software AG und etablierter Vertreiber von Transaktionssystemen für Großkunden aus dem Finanzsektor, der Fertigungs-, Öl- und Bergbauindustrie und der öffentlichen Verwaltung. Mit der Übernahme wollte die Software AG eine stärkere Präsenz auf dem lateinamerikanischen Markt erreichen.

Von einem diagonalen oder auch konglomeraten Zusammenschluss wird indes gesprochen, wenn es sich um eine Verschmelzung von Unternehmen unterschiedlicher wirtschaftlicher Bereiche handelt. Dieser Zusammenschluss ist also mit einem Vordringen in neue Produkt-Markt-Felder verbunden. Diagonale Zusammenschlüsse basieren in der Regel auf einer Diversifikations- oder Expansionsstrategie. Ein Beispiel hierfür ist die Strategie des Softwarekonzerns Infor, der mit einem Umsatz von rund 2,8 Mrd. US-Dollar und mehr als 13.000 Mitarbeitern zu den großen weltweiten Softwarekonzernen gehört. Die Strategie besteht in einem schnellen Wachstum durch den Zukauf einer Vielzahl von Softwarefirmen aus den unterschiedlichsten Bereichen. Dabei steht im Gegensatz zu vielen anderen Unternehmensübernahmen in der Softwareindustrie keine Migrationsstrategie im Mittelpunkt, die das Ziel verfolgt, die übernommenen Anwendungen zu einer integrierten Lösung zusammenzuführen. Erlöse werden, neben dem Lizenzgeschäft, auch kurzfristig aus den laufenden Serviceverträgen generiert. Darüber hinaus ergeben sich Möglichkeiten zur Kosteneinsparung etwa durch ein straffes Kostenmanagement sowie die Zusammenlegung von Overhead-Bereichen.

In der folgenden Abb. 3.13 sind die verschiedenen Formen von Unternehmenszusammenschlüssen dargestellt.

Im folgenden Abschnitt werden die Motive untersucht, die das Management dazu bewegen, ein Unternehmen zu akquirieren oder das eigene Unternehmen zu verkaufen.

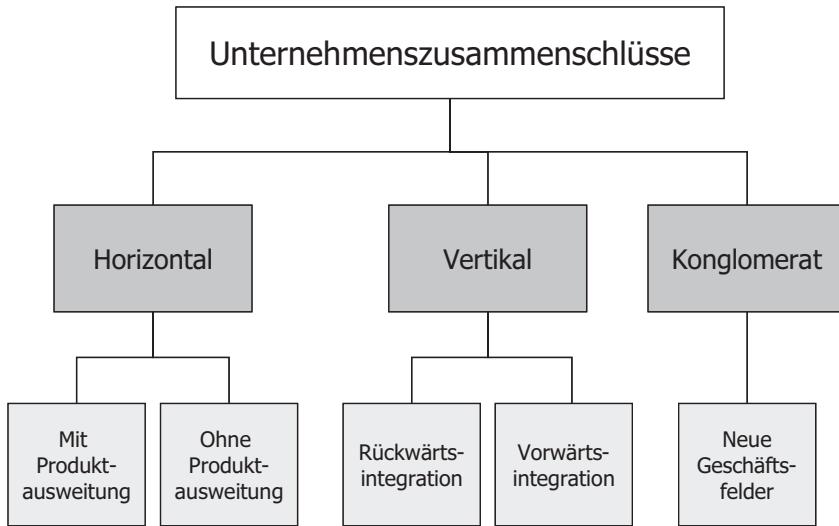


Abb. 3.13 Formen von Unternehmenszusammenschüssen. (Wirtz 2003, S. 19)

3.1.2.2 Motive für Unternehmensübernahmen

Da Unternehmensübernahmen sämtliche Bereiche der beteiligten Unternehmen betreffen können, ist grundsätzlich die Perspektive aller relevanten Stakeholder zu betrachten. Dazu zählen Kapitaleigentümer, das Management, Mitarbeiter, Zulieferer, Kunden, Konkurrenten sowie die Gesellschaft, da Mergers & Acquisitions häufig auch massive Veränderungen der Arbeitsplatzsituation sowie Entlassungen zur Folge haben.

Im Folgenden wollen wir uns auf eine kurze Darstellung der Motive aus Unternehmenssicht konzentrieren. Diese können in strategische, finanzielle und persönliche Motive differenziert werden (Wirtz 2003, S. 57–76).

Strategische Motive für Unternehmenszusammenschlüsse beziehen sich in der Regel auf die Realisierung von Synergieeffekten und lassen sich in

- Marktmotive,
- Leistungsmotive sowie
- Risikomotive

unterscheiden.

Marktmotive beziehen sich dabei sowohl auf die Beschaffungs- als auch die Absatzseite. So lässt sich etwa die Verhandlungsmacht gegenüber einem gemeinsamen Lieferanten stärken, da größere Mengen beschafft werden. Neben dem Beschaffungsmarkt spielt die Förderung des Absatzes als Marktmotiv für Unternehmenszusammenschlüsse eine wichtige Rolle. Durch eine Zusammenlegung der Absatzaktivitäten der beteiligten Unternehmen können neben Synergien auch Wettbewerbsvorteile erzielt werden. Über eine stärkere Position auf dem Absatzmarkt kann zum einen ein stärkerer Einfluss auf den Absatzpreis

genommen, zum anderen können mitunter sogar Wettbewerber aus dem Markt gedrängt werden. Schließlich kann eine Übernahme auch Wachstumspotenziale, z. B. durch neue Absatzregionen, erschließen.

Synergien können nach einem Unternehmenszusammenschluss auch dadurch entstehen, dass Ressourcen und Fähigkeiten von den beteiligten Unternehmen gemeinsam genutzt werden. Ein Leistungsmotiv für eine Übernahme liegt demnach vor, wenn Synergien in einzelnen Unternehmensfunktionen, wie Forschung und Entwicklung, Beschaffung, Produktion, Marketing etc., zu erwarten sind und genutzt werden sollen. Dabei ist es zum einen möglich, die Technologien sowie das Know-how der einzelnen Unternehmen zu kombinieren, um neue oder qualitativ bessere Produkte und Dienstleistungen herzustellen. Im Kontext des Softwaremarktes ist hier beispielsweise an die Entwicklung integrierter Systeme zu denken. Zum anderen lassen sich durch einen Zusammenschluss die vorhandenen Kapazitäten im Rahmen von Entwicklungsprozessen besser ausnutzen.

Ein Risikomotiv liegt bei Unternehmenszusammenschlüssen insbesondere dann vor, wenn diese aufgrund einer Diversifikationsstrategie erfolgt sind. Risiken entstehen etwa durch die Abhängigkeit eines Unternehmens von einem Produkt oder von der Entwicklung einer Branche. Die Erweiterung der Produktpalette oder das Vorstoßen in neue Branchen durch eine Unternehmensübernahme soll dazu beitragen, diese Risiken zu reduzieren.

Dabei gelten die hier unter der strategischen Perspektive genannten Motive für Unternehmensübernahmen häufig auch für Kooperationen im Allgemeinen und sind somit als eine Ergänzung der in Abschn. 3.1.1.1 dargestellten Kooperationsvorteile anzusehen.

Neben den angeführten Markt-, Leistungs- und Risikomotiven, die aus der Unternehmensstrategie entstehen, können auch finanzielle Motive Anlass für eine Unternehmensübernahme bzw. einen Unternehmenszusammenschluss sein. Das übergeordnete Motiv ist in der Regel die Steigerung der Rentabilität durch die Erzielung von Gewinnen oder durch die Ausnutzung von steuerlichen Verlustvorträgen. Die Basis hierfür bilden kapitalmarktbedingte sowie bilanzpolitische und steuerliche Überlegungen.

Darüber hinaus können auch persönliche Motive des Managements zur Maximierung des eigenen Nutzens den Anlass für eine Unternehmensakquisition oder -fusion geben. Hierzu sind mehrere Erklärungsansätze entwickelt worden, die u. a. Selbstüberschätzung der Manager und Machterweiterung als entscheidende Gründe für Unternehmensübernahmen anführen (Wirtz 2003, S. 57–76).

3.1.2.3 Konzentrationstendenzen in der Softwareindustrie

Wir haben bereits an verschiedenen Stellen darauf hingewiesen, dass auf Softwaremärkten aufgrund von Netzeffekten das Winner-takes-it-all-Prinzip gilt. Diese Tendenz hin zum Monopol wird letztlich auch durch Unternehmensübernahmen vorangetrieben. Die Bedeutung von Mergers & Acquisitions (M&A) in der Softwareindustrie wird zudem durch den in der folgenden Abb. 3.14 dargestellten Branchenvergleich deutlich.

Die Abbildung zeigt, dass die Technologie-, Medien- und Telekommunikationsbranche mit großem Abstand an der Spitze liegt, wenn man Mergers & Acquisitions nach dem Transaktionsvolumen listet. Vor diesem Hintergrund ist auch die Relevanz der Software-

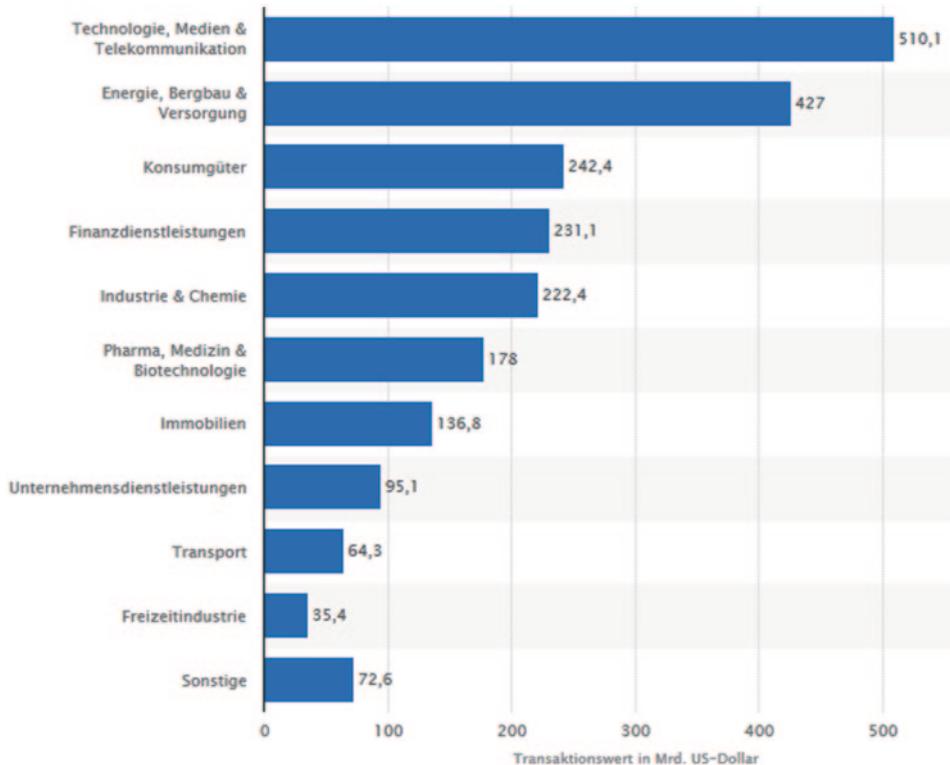


Abb. 3.14 Verteilung der weltweiten M&A Deals in 2013 (Q1–Q3) nach Branchen. (Statista 2014b)

industrie für die Weltwirtschaft zu betrachten. Sie stellt einen entscheidenden Anteil des Informations- und Technologiesektors dar, der zurzeit 5,4 % des weltweiten Bruttosozialprodukts einnimmt. Vor allem die jüngsten Zukäufe der Branchenriesen haben ein beeindruckendes Ausmaß angenommen (Schieff et al. 2013, S. 421). So übernahm Hewlett-Packard in 2011 Autonomy für 10,3 Mrd. US-Dollar, Skype wurde von Microsoft für 8,5 Mrd. \$ und Cognos von IBM für 4,9 Mrd. übernommen. In 2014 übernahm Facebook WhatsApp für 16 Mrd. US-Dollar. Diese Beispiele verdeutlichen zudem die praktische Bedeutung der Übernahmen in der Softwareindustrie.

Ein weiteres prominentes Beispiel für diese Konzentrationstendenzen sind die Anbieter von ERP-Software und hier insbesondere die Übernahmestrategien von Oracle. Oracle hat in der Vergangenheit durch eine Vielzahl von M&A-Aktivitäten auf sich aufmerksam gemacht.

Nur vier Tage, nachdem PeopleSoft bekannt machte, das Konkurrenzunternehmen J.D. Edwards kaufen zu wollen, kündigte Oracle am 6.6.2003 seinerseits den beabsichtigten Kauf von PeopleSoft für 5,1 Mrd. \$ an. Durch die Übernahme von J.D. Edwards wäre PeopleSoft zum zweitgrößten Anbieter für Unternehmenssoftware nach SAP aufgestiegen und hätte Oracle damit überholt. Mit der Ankündigung von Oracle begann eine rund 19 Monate dauernde Übernahmeschlacht. Schließlich konnte Oracle das Unternehmen für

insgesamt rund 10,3 Mrd. \$ übernehmen. Dies war der bis dahin größte Merger in der Geschichte der Softwareindustrie; Oracle gewann den zweiten Rang im Markt für Unternehmenssoftware zurück und konnte den Abstand zur SAP verringern.

Oracle unterstützt und entwickelt bis dato die Software von PeopleSoft und J.D. Edwards weiter. Gleichzeitig bietet das Unternehmen unter dem Namen „Fusion“ eine neue, alle Produktlinien integrierende Softwarelösung an. Hierbei war ein sehr großer Integrationsaufwand notwendig, das Forschungs- und Entwicklungsteam für das Projekt Fusion war eines der größten weltweit.

Mit dem Kauf von Siebel für rund 5,85 Mrd. \$ akquirierte Oracle zudem einen der führenden Anbieter für CRM-Software. Damit wurde die Konsolidierungswelle auf dem ERP-Softwaremarkt weiter vorangetrieben (o. V. 2005).

In 2013 übernahm Oracle für 1,7 Mrd. \$ Acme Packet, einen Netzwerkkomponentenanbieter, um seine mit dem Kauf durch Sun Microsystems begründete Stellung auf dem Hardwaremarkt zu stärken (o. V. 2013).

Eine viel diskutierte These besagt, dass durch Mergers & Acquisitions das Gleichgewicht auf dem Markt gestört wird. Denn die anderen Akteure werden quasi dazu gezwungen, ihrerseits Übernahmen zu tätigen, um langfristig auf dem Markt unabhängig bleiben zu können und nicht selbst zum Übernahmekandidaten zu werden (Hutzschenreuter und Stratigakis 2003).

Gerade für die Softwareindustrie gilt aufgrund der Existenz von Netzeffekten, dass Größe für einen Anbieter per se in der Regel sinnvoll ist und einen Wettbewerbsfaktor darstellen kann. So kommentierte Oracle-Chef Lawrence Ellison die Übernahme von Siebel mit den Worten: „Die Akquisition von Siebel bringt uns dem Ziel einen Schritt näher, global die Nummer eins im Geschäft mit Business-Software zu werden“ (o. V. 2005). Darüber hinaus erhält Oracle in diesem Fall auch die Chance, seinen Kunden eine umfassende und integrierte Lösung anbieten zu können.

Ähnliche Konzentrationstendenzen sind auch auf dem Markt für Office-Software zu beobachten. Während vor einigen Jahren noch eine Vielzahl von erfolgreichen Alternativen am Markt existierte, z. B. Lotus 1-2-3 für Spreadsheets oder Word Perfect für Textverarbeitung, ist Microsoft in diesem Markt fast zum Monopolisten geworden. Die Open Source Community ist der einzige noch ernst zu nehmende Wettbewerber (siehe hierzu Kap. 8).

Auch wenn monopolistische Strukturen für den Kunden bzw. Nachfrager in vielen Fällen als hochproblematisch eingeschätzt werden, sind ihre Konsequenzen auf Netzeffektmärkten durchaus umstritten. Einerseits gilt natürlich, dass die Abhängigkeit der Kunden vom Anbieter drastisch zunimmt. Theoretisch wäre der Anbieter auch in der Lage, die Preise zu erhöhen, was – wie oben bereits dargestellt – etwa im Fall Microsoft bislang aber kaum passiert ist. Zudem besteht ein weiteres Problem für die Anwender darin, dass in Monopolen tendenziell eine eher geringe Innovationstätigkeit zu erwarten ist. Andererseits führen diese monopolartigen Strukturen für den Anwender zu dem Vorteil, dass Inkompatibilitäten weitgehend vermieden werden. In diese Richtung argumentieren etwa Stanley Liebowitz und Stephen Margolis in verschiedenen Beiträgen (z. B. Liebowitz und Margolis 1994, 2001).

In der Literatur wird häufig angemerkt, dass es in der Softwareindustrie eine zunehmende Konsolidierungswelle gibt (Schief et al. 2013, S. 421), die hauptsächlich von den großen Anbietern vorangetrieben wird (Friedewald et al. 2001). Eine mögliche Reaktion kleinerer Softwareanbieter auf diese Konsolidierungswelle sind Kooperationen mit anderen Softwareherstellern sowie die Konzentration auf Spezial- oder Nischenlösungen.

3.1.2.4 Determinanten des M&A-Erfolges in der Softwareindustrie

Der M&A-Erfolg kann aus Perspektive des Käufer- und des Zielunternehmens sowie aus der Kombination beider Unternehmen analysiert werden. Kurzfristige Analysen erfassen besser den spezifischen Effekt einer M&A-Transaktion, da dieser weniger durch andere Effekte verwässert werden kann. Langfristige Betrachtungen bieten hingegen Erkenntnisse über den (Integrations-) Erfolg einer M&A-Transaktion im Zeitverlauf. Die Verwendung von Kontrollgruppen ermöglicht es dabei, nicht direkt transaktionsbedingte Effekte zu bereinigen (Beitel und Schiereck 2003, S. 505).

Meglio und Risberg (2011, S. 422–427) unterscheiden auf Basis einer Literaturanalyse vier Klassen von Kennzahlen zur Bewertung des M&A-Erfolges: Kapitalmarkt-, Jahresabschluss-, operative und allgemeine Erfolgsgrößen.

Kapitalmarktgrößen umfassen die Marktkapitalisierung oder das Risiko von Unternehmen. Die in der M&A-Literatur am häufigsten verwendete Erfolgsmetrik in diesem Zusammenhang ist die kumulative abnormale Rendite (Cumulative Abnormal Return – CAR). Die Methode untersucht den Einfluss einer M&A-Transaktion auf den Aktienkurs in einem definierten Zeitfenster, das häufig mehrere Tage vor und nach dem Ankündigungszeitpunkt umfasst. Hierfür wird an jedem Tag innerhalb des Zeitfensters eine abnormale Rendite aus der Differenz des realisierten und des erwarteten Aktienkurses ermittelt. Dabei wird der erwartete Aktienkurs in einer Schätzperiode (z. B. 200 Tage) über die Entwicklung des Aktienkurses im Vergleich zu einem Referenzwert approximiert. Zur Berechnung des Referenzwertes (z. B. S&P 500) wird häufig das sogenannte Marktmodell verwendet (MacKinlay 1997, S. 15). Die abnormalen Renditen werden schließlich über alle Tage des definierten Ereigniszeitfensters summiert und als kumulative abnormale Rendite ausgewiesen. Im Falle positiver CARs sind M&A-Transaktionen als erfolgreich anzusehen, da der realisierte Aktienkurs über den Erwartungswert gesteigert werden konnte.

Die Vorteile der Kapitalmarktstudien liegen in der Datenverfügbarkeit und der einheitlichen Bemessungsgrundlage. Allerdings sind diese Analysen nur für öffentlich gelistete Firmen möglich und setzen in Hinblick auf die effiziente Kapitalmarkttheorie eine adäquate Bewertung der Transaktion durch die Kapitalmarktteilnehmer voraus (Datta et al. 1992, S. 73; King et al. 2004, S. 196).

Als Jahresabschlussgrößen werden Kennzahlen in den Bereichen Profitabilität, Wachstum, Fremdfinanzierungsgrad, Liquidität und Zahlungsströme betrachtet. Zur Erfolgsmesung wird die zeitliche Entwicklung der Kennzahlen analysiert und geprüft, ob diese sich nach einer M&A-Transaktion signifikant verändern. Der Vorteil ist, dass die Kennzahlen reale Finanzgrößen der Firmen erfassen. Allerdings ist die Verfügbarkeit des Jahresabschlusses von Publikationsvorschriften abhängig und die Kennzahlen können durch die angewandte Rechnungslegung beeinflusst werden (Thanos und Papadakis 2011, S. 113–114).

Operative Größen erfassen Unternehmenskennzahlen in den Bereichen Marketing, Innovation und Produktivität. Durch eine Analyse der zeitlichen Entwicklung dieser Kennzahlen kann das Erreichen spezifischer M&A-Ziele, wie z. B. Marktanteil oder Patentanzahl, untersucht werden. Nachteile sind hingegen, dass die Operationalisierung der Metriken kontrovers und die Verfügbarkeit der Daten begrenzt sein können (Meglio und Risberg 2011, S. 422–427).

Als allgemeine Erfolgsgrößen werden schließlich die subjektive Bewertung der M&A-Transaktion und die Analyse des zeitlichen Fortbestands der akquirierten Einheit angesehen. Die subjektive Bewertung der M&A-Transaktion wird meist durch eine Befragung des Managements erhoben. Dies ermöglicht spezifische Einblicke, etwa in die Realisierung von M&A-Zielen, kann jedoch andererseits Verzerrungen unterliegen (Bruner 2001, S. 12). Die Messung des zeitlichen Fortbestands kann hingegen methodisch objektiv festgestellt werden, jedoch auch unabhängig vom Transaktionserfolg sein (Halebian et al. 2009, S. 491).

3.1.2.5 Bewertung des M&A-Erfolges in der Softwareindustrie

In der Literatur findet sich eine Reihe von branchenunabhängigen Analysen, die den Erfolg von M&A untersuchen (Datta et al. 1992; Bruner 2001; Schief et al. 2013). Wir wollen uns hier jedoch auf die Studien beschränken, die sich mit dem M&A-Erfolg in der Softwareindustrie beschäftigen, und lediglich am Ende des Abschnitts kurz auf die Frage eingehen, inwieweit die Ergebnisse verallgemeinerbar sind.

Unternehmenszusammenschlüsse bieten auf Softwaremärkten die Chance, strategische Wettbewerbsvorteile zu erzielen. Dies gilt – wie bereits ausführlich dargestellt – insbesondere, weil Softwareanbieter auf Netzeffektmärkten agieren. Doch werden Käufer durch eine Akquisition wirklich mit einer Steigerung ihres Unternehmenswertes belohnt? Diese Frage wurde z. B. von Izci und Schiereck (2010) im Rahmen einer empirischen Studie untersucht.

Die Datenbasis bestand aus 81 internationalen Unternehmensübernahmen im Bereich Unternehmenssoftware im Zeitraum von 2000 bis 2007. Dabei wurden folgende Typen von Transaktionen betrachtet: Die Transaktion ist abgeschlossen, das Transaktionsvolumen beträgt mindestens 50 Mio. US-Dollar, und der Käufer erwirbt über 20 % der Anteile des Zielunternehmens. Käufer und Ziel waren börsennotierte Unternehmen, deren Aktienkurse zwischen 230 Tagen vor und 30 Tage nach dem Ereignis der M&A-Ankündigung abrufbar waren.

Die Untersuchung basiert auf der Analyse der oben skizzierten „abnormalen Renditen“, also der Differenz zwischen der tatsächlich beobachteten Rendite und der „normalen“ Rendite, die ohne M&A-Ankündigung zu erwarten gewesen wäre. In einem Zeitfenster von maximal 15 Tagen vor und nach der M&A-Ankündigung wurden die abnormalen Renditen für die Käufer- und Zielunternehmen berechnet und kumuliert.

Die Ergebnisse zeigen, dass die Zielunternehmen eine positive kumulierte abnormale Rendite (CAR) erzielen, während die Käuferunternehmen mit leicht negativen Werten rechnen müssen. Insbesondere zum Zeitpunkt der Ankündigung sowie wenige Tage da-

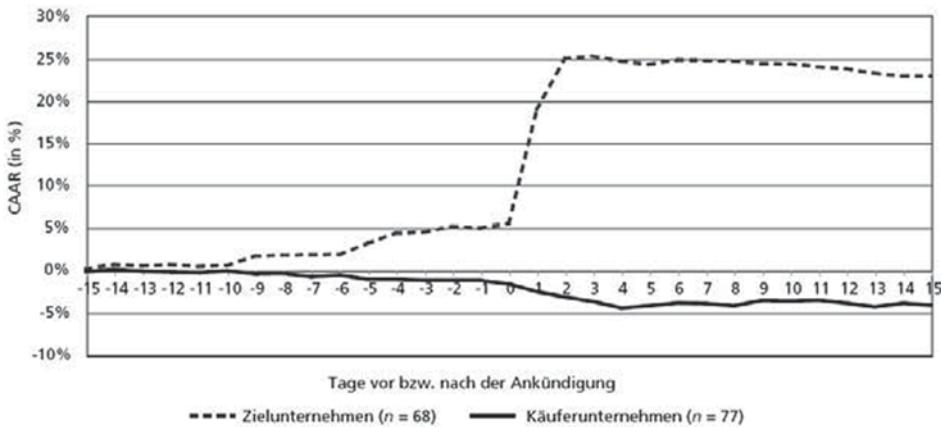


Abb. 3.15 Kumulierte durchschnittliche abnormale Rendite (CAR) für Käufer- und Zielunternehmen. (Izci und Schiereck 2010)

nach sind der Anstieg bzw. Abfall der CAR statistisch signifikant und die Tendenz bleibt über den kompletten weiteren Beobachtungszeitraum erhalten (siehe Abb. 3.15).

In einem zweiten Schritt befasste sich die Studie mit den Faktoren, die letztendlich die Kursentwicklung und somit auch den Erfolg der Käuferunternehmen beeinflussen. Auf Basis einer multivariaten Regression konnte der erwartete positive Einfluss bei einer Übernahme eines Unternehmens desselben Wirtschaftszweiges oder im internationalen Umfeld nicht bestätigt werden (Izci und Schiereck 2010). Der ermittelte Zusammenhang war zwar positiv, aber nicht statistisch signifikant. Ebenso beeinflussten die Determinanten „akquirierter Anteil“ und „Transaktionsvolumen“ die Bewertung des Käuferunternehmens nicht signifikant. Ein leicht positiver signifikanter Einfluss konnte lediglich für die Variable „Bar- oder Aktienkauf“ beobachtet werden: Der reine Barkauf wirkte sich positiv auf die Kapitalmarktbewertung aus.

Hochsignifikant negativ wirkten dagegen die Variablen „relative Größe“ sowie „Kapitalmarktpreformance des Zielunternehmens“. Die negative Reaktion im Falle einer Großakquisition lässt sich mit der Erwartung des Kapitalmarkts begründen, dass die anfallenden Integrationskosten die potenziellen Synergieeffekte übertreffen werden (Loefert 2007, S. 163; Izci und Schiereck 2010). Der negative Einfluss der historischen Kapitalmarktpfomance des Zielunternehmens kann aufgrund der implizit eingepreisten Wachstumservartungen erklärt werden: Die in den Preisaufschlägen der Übernahmeangebote erfassten hohen Übernahmeprämien deuten entweder auf eine systematische Überbewertung des Zielunternehmens durch den Käufer oder auf eine vermutete nachhaltige Abschwächung der Wachstumsdynamik des Zielunternehmens nach der Akquisition hin. Beide Varianten ziehen negative Reaktionen bei der Börsenbewertung des Käuferunternehmens nach sich.

Die Ergebnisse der Kapitalmarktstudie zeigen, dass M&A-Transaktionen in der Unternehmenssoftwareindustrie zunächst nur für Zielunternehmen wertgenerierend wirken, wohingegen Käuferunternehmen eher mit einer Wertvernichtung im Sinne des Shareholder Value-Konzeptes rechnen müssen.

Überraschend ist, dass nicht das insgesamt hohe Risiko in dem innovativen und sich ständig verändernden Industriezweig ausschlaggebend für die negative Bewertung des Zielunternehmens ist. Grenzüberschreitende oder diversifizierende Transaktionen werden nicht besonders negativ aufgenommen. Vielmehr sind es kulturelle Herausforderungen, die den Kapitalmarkt zu negativen Bewertungen leiten. Um positive Marktreaktionen zu erreichen, müssen die Anbieter von Unternehmenssoftware zukünftig überzeugend darlegen, dass die Integration von großen Wettbewerbern mit besonders dynamischem Wachstum ohne Gefährdung der Synergieeffekte erfolgen kann. Denn der Verzicht auf Übernahmen stellt für eine rasch konsolidierende, von Netzeffekten geprägte Branche wie die Softwareindustrie keine sinnvolle Alternative dar.

Neben der Analyse von Izci und Schiereck findet sich eine Reihe weiterer Studien, in denen ebenfalls die kumulierte abnormale Rendite (CAR) als Erfolgsmetrik genutzt wird und die als kurzfristige Ereignisstudien durchgeführt wurden. Während der Erfolg der Zielunternehmen gemäß den CAR-Studien regelmäßig positiv ausfällt, ergeben die Analysen der Käuferunternehmen ein uneinheitliches Bild. Es ist somit für Softwareunternehmen fraglich, ob Käufer das mit einer Transaktion erhoffte Wertschöpfungspotential, das z. B. durch Netzeffekte entsteht, ausschöpfen können. In der Telekommunikationsindustrie, die ebenfalls durch Netzeffekte geprägt ist, wurden auch negative Ergebnisse für Käufer belegt (Izci und Schiereck 2010, S. 69).

Vor dem Hintergrund dieser empirischen Ergebnisse stellt sich die Frage, von welchen Faktoren der M&A-Erfolg in der Softwareindustrie abhängt. Um diese Frage zu beantworten, verwenden wir im Folgenden das an Halebian et al. (2009, S. 473) angelehnte und in Abb. 3.16 dargestellte Analysemodell. Die Erfolgstreiber werden hierbei in fünf Kategorien unterteilt: Umweltfaktoren umfassen exogene Größen (z. B. rechtliche Regularien), die für die Unternehmen gegeben sind. Transaktionseigenschaften beschreiben spezifische Charakteristika einer bestimmten Transaktion (z. B. Art der Zahlung). Bei den Eigenschaften der Käufer- (z. B. Akquisitionserfahrung) und Zielunternehmen (z. B. Rechtsform) handelt es sich um Merkmale, die unabhängig von der Transaktion gelten. Eigenschaften der Unternehmenskombination setzen schließlich Charakteristika beider Akteure ins Verhältnis (z. B. relative Größe).

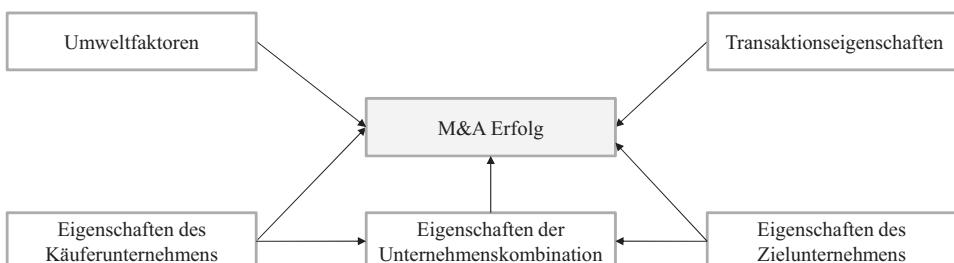


Abb. 3.16 Kategorien der M&A-Erfolgsfaktoren

Bezüglich der Relevanz dieser M&A-Erfolgsfaktoren in der Softwareindustrie liegen bislang erst wenige und teilweise auch widersprüchliche Ergebnisse vor (siehe Tab. 3.1). Zentrale Erkenntnisse wollen wir im Folgenden darstellen und kurz diskutieren.

Umweltfaktoren werden in den vorliegenden Studien nicht betrachtet. Dies ist etwas überraschend, da diese Faktoren durchaus auch für die Softwareindustrie relevant sind. Dies gilt z. B. für den Einfluss von Transaktionswellen, die bereits mehrfach auch in der Softwareindustrie beobachtet werden konnten (Schiff 2007, S. 1). Ein Beispiel hierfür ist die Konsolidierungswelle im Bereich Business Intelligence Software: Im Jahr 2007 waren mehrere zeitnahe Großtransaktionen zu verzeichnen, so wurde Hyperion durch Oracle, Business Objects durch SAP und Cognos durch IBM akquiriert. Die hohe Dynamik in der Softwareindustrie bringt neue Innovationsfelder hervor, in denen mit zunehmender Reife eine Konsolidierung erfolgt. Hinzu kommt, dass der Netzeffektcharakter von Software häufig zu Lock-in-Effekten führt. Es ist daher davon auszugehen, dass Vorreiter in einer Übernahmewelle überdurchschnittliche Renditen erzielen (Mcnamara et al. 2008, S. 113), da sie ihre Nutzerbasis steigern und durch Netzeffekte Marktmacht etablieren können.

Zu berücksichtigen ist allerdings, dass eine Ausweitung der Marktmacht auch in den Blick gesetzlicher Regularien geraten kann. Microsofts Kauf von Skype wurde z. B. durch die Kartellbehörden eingehend geprüft. Käufer müssen somit versuchen, eine starke Marktposition zu etablieren, um Netzeffekte zu erzielen, ohne dabei eine offensichtliche Monopolstellung einzunehmen (Budzinski und Christiansen 2007, S. 9–10). Im Falle einer Konfrontation mit gesetzlichen Regularien kann es zu Auflagen kommen, die die angestrebten Wertpotentiale verringern. Es ist folglich anzunehmen, dass Transaktionen, die besonderen juristischen Überprüfungen unterliegen, negativ bewertet werden.

Im Hinblick auf die Transaktionseigenschaften kommen Léger und Quach (2009, S. 709) zu dem Ergebnis, dass ein hohes Preis-Buch-Verhältnis einen negativen Einfluss hat. Das Preis-Buch-Verhältnis setzt den M&A-Übernahmepreis ins Verhältnis zu dem Buchwert des Zielunternehmens. Je höher das Preis-Buch-Verhältnis, desto größer ist das bezahlte Akquisitionspremium und somit auch das Risiko, die erhofften Wertpotentiale nicht zu realisieren.

Ein weiterer interessanter Aspekt im Zusammenhang mit dem Transaktionscharakter sind Bieterwettkämpfe. Durch den intensiven Kampf um Innovationen und eine große Nutzerbasis konkurrieren in der Softwareindustrie oft mehrere Anbieter um attraktive Übernahmeziele. Im Jahre 2005 überbot z. B. Oracle den Konkurrenten SAP bei der Übernahme von Retec. Es liegt nahe, dass sich Angebote mehrerer Interessenten negativ auf den M&A-Erfolg des Käufers auswirken, da dies den Preis in die Höhe treiben kann.

Eine zentrale Eigenschaft des Käuferunternehmens ist seine Größe. Große Unternehmen verfügen meist über eine beträchtliche Nutzerbasis, die sie für zugekaufte Produkte nutzen können. Es ist daher davon auszugehen, dass große Käuferunternehmen in der Softwareindustrie überproportionale Umsatzsynergien durch Netzeffekte erzielen können. Überraschenderweise bewerten z. B. Gao und Iyer (2006, S. 134) diesen Faktor negativ und führen dies unter anderem darauf zurück, dass Manager von großen Unternehmen eher zur Überheblichkeit neigen. Laamanen et al. (2013, S. 21) sehen positive Effekte für

Tab. 3.1 Publikationen der M&A-Erfolgsforschung in der Softwareindustrie. (Schief et al. 2013, S. 427)

Die Determinanten werden in Hinblick auf ihre Wirkung auf den Erfolg des Käuferunternehmens betrachtet

Kursive Determinanten: werden nur in Studien der Softwareindustrie verwendet, nicht in allgemeinen M&A Literaturstudien

*Studie basierend auf dreigliedrigem Produktklassifikationssystem

**Studie basierend auf fünfgliedrigem Prod.

"n.s." steht für nicht signifikante Resultate

1) "n.a." bedeutet, dass kein R² angegeben wurde

2) "heterogene Ergebnisse" bedeutet, dass innerhalb

3) "uneinheitlich" bedeutet, dass über alle Studien

3) unethisch bedeutet, dass über alle Studien

stärker diversifizierte Käuferunternehmen. Dieser Effekt könnte auf die Realisierung von indirekten Netzeffekten durch Komplementärangebote zurückzuführen sein.

Hinsichtlich der Eigenschaften der Zielunternehmen spielt möglicherweise ihr ex-ante Erfolgsausweis eine wichtige Rolle. Innovative Unternehmen sind kurz nach ihrer Gründung oft noch nicht profitabel, wachsen dann jedoch häufig schnell, z. B. durch Netzeffekte. In der Softwareindustrie sollte es somit vorteilhafter sein, junge Unternehmen aufzukaufen, bevor deren Wachstum und Erfolg den Übernahmepreis zu sehr erhöhen. Hinzu kommt, dass viele der Unternehmen zunächst in privater Hand sind und erst mit zunehmender Größe an den Kapitalmarkt gehen. Private Unternehmen sind durch die niedrigere Fungibilität privater Anteilsbesitze i. d. R. günstiger zu erwerben. Tatsächlich kommen Laamanen et al. (2013, S. 21 f.) zu dem Ergebnis, dass Übernahmen von privaten Unternehmen aus diesem Grund zu einer erfolgreicher kurzfristigen Kapitalmarktreaktion führen. Darüber hinaus zeigen sie, dass die Akquisition veräußerter Anteile von anderen Unternehmen sowohl kurz- als auch langfristig positiv zu bewerten ist. Einer der Gründe hierfür wird in den Verhandlungsvorteilen der Käufer gesehen, die auf die oftmals ernste Lage des Verkäufers zurückzuführen sind.

Als zentrale Eigenschaften der Unternehmenskombination wird der Grad der Verwandtschaft in empirischen Studien analysiert. Gao und Iyer (2006, S. 123–128) führen zu diesem Zweck ein fünfgliedriges Produktklassifikationssystem ein, das die Bereiche Hardware, Systemsoftware, Middleware, Applikationssoftware und Services umfasst. Sie kommen zu dem Ergebnis, dass Transaktionen in gleichen oder entfernten Schichten eher zu negativen Resultaten führen. Die Ursache hierfür wird in mangelnden Komplementäreffekten und damit einhergehenden indirekten Netzeffekten gesehen. Léger und Quach (2009, S. 709) untersuchen ebenfalls Komplementäreffekte, jedoch basierend auf qualitativen Einschätzungen von Pressemeldungen. Sie erkennen negative Effekte für Transaktionen mit hohem Komplementäreffektpotential. Vor dem Hintergrund der ökonomischen Brancheneigenschaften ist dies sehr überraschend: Die niedrigen Produktionskosten und umsatzfördernden Netzeffekte in der Softwareindustrie sowie die Tatsache, dass Marktmacht für Softwareunternehmen eine entscheidende Rolle spielt, hätten eigentlich andere Ergebnisse erwarten lassen. Insofern bleibt fraglich, ob die Studienergebnisse auf die Art der Erhebungsmethode zurückzuführen sind, oder ob die indirekten Netzeffekte bei Komplementärtransaktionen wirklich nicht zum Tragen kommen.

In welchem Maße sind die bisherigen Ergebnisse der Studien als spezifisch für die Softwareindustrie einzuschätzen? Schief (2014) vergleicht einerseits branchenübergreifende Studien sowie andererseits solche innerhalb der Softwareindustrie und zeigt, dass Einigkeit insofern besteht, als der Erfolg der Zielunternehmen in allen Studien positiv bewertet wird, während die Analysen für den Erfolg der Käuferunternehmen ein uneinheitliches Bild zeigen. Deutlich wird aber auch, dass bislang nur wenige der generischen M&A-Erfolgsfaktoren auf ihre Bedeutung im Kontext der Softwareindustrie untersucht wurden. Zudem wird die Wirkung der untersuchten Faktoren teilweise widersprüchlich (z. B. Ex-ante Erfolg des Ziels) oder nur in einer Studie (z. B. Preis-Buch-Verhältnis) bewertet. Lediglich vereinzelt werden die Effekte (z. B. Größe des Käufers, Rechtsform) in

mehreren Studien übereinstimmend eingeschätzt. Insgesamt kann festgehalten werden, dass die Ergebnisse der empirischen Studien noch recht widersprüchlich sind und viele Fragen offen lassen. Dies hängt vermutlich auch damit zusammen, dass die software-industriespezifische M&A-Forschung erst in der jüngeren Vergangenheit an Bedeutung gewonnen hat.

Ein Großteil der bisher veröffentlichten empirischen Untersuchungen wurde als Ereignisstudien durchgeführt. Bei Anwendung dieser Methode werden die kurzfristigen Kapitalmarktreaktionen fokussiert – langfristig spielen jedoch Wachstum, Profitabilität und Innovation eine entscheidende Rolle, da Akquisitionen die Netzeffekte und Dynamik der Softwareunternehmen stärken sollen. So kommen auch Léger und Quach (2009, S. 710 f.) in ihrer Analyse, in der sie den Einfluss auf Jahresabschlussgrößen untersuchen, zu dem Schluss, dass Transaktionen in der Softwareindustrie kurzfristig an den Kapitalmärkten unterbewertet werden und Synergien, z. B. durch Netzeffekte, langfristig erzielt und besser gemessen werden können.

3.2 Vertriebsstrategien

Vertriebsstrategien beschäftigen sich mit den Entscheidungen in Bezug auf die Versorgung nachgelagerter Unternehmen in der Wertschöpfungskette mit Gütern und/oder Dienstleistungen. Sie umfassen marktgerichtete akquisitorische sowie vertriebslogistische Aktivitäten (Homburg und Krohmer 2006, S. 864–866).

In diesem Buch beschäftigen wir uns ausschließlich mit den marktgerichteten akquisitorischen Vertriebsaktivitäten, da es im Rahmen der Vertriebslogistik um die Sicherstellung der physischen Verfügbarkeit der Produkte bei den Kunden geht, die für Softwareprodukte keine besondere Rolle spielt.

Im akquisitorischen Bereich sind insbesondere die folgenden Aspekte von Bedeutung:

- die Gestaltung des Vertriebssystems,
- die Gestaltung der Beziehungen zu Vertriebspartnern und Key Accounts,
- Kennzahlensysteme als Instrument für das Vertriebscontrolling sowie
- die Gestaltung der Verkaufsaktivitäten.

3.2.1 Gestaltung des Vertriebssystems: Organisation und Vertriebswege in der Softwareindustrie

Im Rahmen der Gestaltung des Vertriebssystems sind insbesondere Entscheidungen über die Vertriebsorganisation sowie die Vertriebswege zu treffen.

Betrachten wir zunächst die Frage nach der Vertriebsorganisation. Dabei kann die Strukturierung grundsätzlich nach folgenden Kriterien erfolgen:

- Regionen,
- Branchen,
- Produkte sowie
- Bestands- und Neukunden.

Bei der Gliederung des Vertriebes nach Regionen ist eine Einteilung in Kontinente, Länder oder auch Bundesländer üblich. Damit ist der Vorteil einer großen Kundennähe verbunden.

Zudem erfolgt häufig eine Strukturierung des Vertriebs nach Branchen. Der Vorteil einer solchen Einteilung gegenüber einer regionalen Struktur liegt darin, dass die Vertriebsmitarbeiter in der Regel branchenspezifische Fachkenntnisse haben und die Sprache des Kunden sprechen. Ein potenzieller Nachteil ist demgegenüber der oftmals entstehende höhere Reiseaufwand.

Eine produktorientierte Organisationsform ist in erster Linie für Softwareanbieter sinnvoll, die ein breites Produktangebot besitzen und bei denen ein produktspezifisches Know-how für die Vertriebsaktivitäten erforderlich ist. Dies gilt etwa für den Vertrieb von komplexen SCM- sowie CRM-Systemen. So sind bei vielen Softwareunternehmen sowohl der Vertrieb als auch das Consulting nicht nur regional, sondern auch produktorientiert aufgestellt. Beispielsweise gibt es im SCM- und CRM-Umfeld bei vielen Softwareanbietern Experten, die in weltweiten Projekten für spezielle Fragestellungen, etwa Optimierungsalgorithmen im Rahmen der Einführung und Nutzung von SCM-Lösungen, eingesetzt werden.

Die Unterteilung in Bestands- und Neukunden wird oft vor dem Hintergrund der verschiedenen Charaktere von Vertriebsmitarbeitern diskutiert. Diese lassen sich hinsichtlich ihres Typus als „Jäger“ oder „Farmer“ einordnen. Der „Jäger“ ist – wie der Begriff es bereits ausdrückt – eher geeignet, um Neukunden zu gewinnen und diese für ein neues Produkt oder eine neue Lösung zu begeistern. Der „Farmer“ hingegen fühlt sich eher in einem Kundenumfeld wohl, in dem er sich über langfristig aufgebaute Kontakte in einer vertrauten Umgebung bewegen kann. Er sollte daher in der Regel für die Betreuung von Bestandskunden eingesetzt werden.

Dabei schließen sich diese Kriterien nicht gegenseitig aus. So ist es durchaus üblich, Verantwortungsbereiche nach Produkten und Regionen zu bilden. Das führt etwa dazu, dass ein Bereich für bestimmte Kontinente und Produkte zuständig ist. Dem Vorteil, dass bei einer solchen Organisationsform sowohl regional- als auch produktspezifische Besonderheiten abgebildet werden können, steht der Nachteil gegenüber, dass die Kompetenzverantwortung nicht immer eindeutig geregelt ist.

Im Weiteren wollen wir untersuchen, wie die Vertriebswege zu gestalten sind. Eine elementare Gestaltungsfrage betrifft hierbei die Entscheidung zwischen direktem und indirektem Vertrieb. Wir sprechen von einem indirekten Vertrieb, wenn die Vertriebsaufgaben von externen Unternehmen, beispielsweise Systemhäusern, durchgeführt werden. Beim direkten Vertrieb werden diese Aufgaben intern wahrgenommen (Homburg und Krohmer 2006, S. 873–877).

Die Entscheidung zwischen direktem und indirektem Vertrieb ist sowohl unter Berücksichtigung von Effizienz- als auch Effektivitätsüberlegungen zu treffen. Im Hinblick auf die Effizienz sind die Vertriebsformen auf Basis der entstehenden Transaktionskosten (siehe Abschn. 2.4 dieses Buches) zu beurteilen. Dabei lautet die Grundüberlegung, dass durch den Einsatz von Vertriebspartnern Transaktionskosten eingespart werden können. Diese Einsparungen sind mit der Handelsmarge des Lizenzvertriebs zu vergleichen. Zudem können Effektivitätsüberlegungen die Qualität der Kundenbetreuung, etwa durch räumliche Nähe oder Spezialisierung, sowie die Kundenloyalität betreffen. So ist es denkbar, dass sich Systemhäuser auf bestimmte Branchen konzentrieren und dort ein besonderes Know-how aufbauen. Darüber hinaus ist gerade in stark wachsenden Bereichen der Softwareindustrie zu beobachten, dass Vertriebspartner eingesetzt werden, weil ohne diese Partner ein schnelles Wachstum nicht möglich wäre.

Betrachten wir im Weiteren einige ausgewählte Einflussfaktoren, die die Vorteilhaftigkeit von direktem bzw. indirektem Vertrieb beeinflussen können. Dabei wird in der Marketingliteratur grundsätzlich davon ausgegangen, dass eine hohe Spezifität und Komplexität von Produkten tendenziell für einen Direktvertrieb sprechen (Homburg und Krohmer 2006, S. 874).

Doch gilt dies auch für die Softwareindustrie? Hinsichtlich der Spezifität ist der Aussage zuzustimmen. Ein Anbieter von Individualsoftware, also von spezifischen Lösungen, würde kaum auf die Idee kommen, seine Lösungen auf indirektem Wege zu vertreiben. Der indirekte Vertrieb kommt also grundsätzlich nur für Anbieter von Standardsoftware in Frage. Diese weist jedoch häufig einen sehr hohen Komplexitätsgrad auf. Dies gilt, wie wir in Kap. 1 gesehen haben, insbesondere für Anbieter von ERP-Systemen, wenn wir an die Vielzahl möglicher Anpassungsmaßnahmen im Rahmen von Einführungsprojekten denken. Dennoch wählen Standardsoftwareanbieter in vielen Fällen einen indirekten Vertriebsweg. Die Voraussetzung hierfür sind Vertriebspartner mit einem entsprechend hohen (zum Teil branchenspezifischen) Know-how. Für die Softwareindustrie gilt somit lediglich, dass eine hohe Spezifität tendenziell für einen direkten Vertrieb spricht. Ein weiterer Vorteil eines direkten Vertriebs besteht natürlich darin, dass sich auf diese Weise grundsätzlich eine größere Kundennähe und -loyalität aufbauen lässt.

Darüber hinaus ist die Entscheidung zwischen direktem und indirektem Vertrieb auch von der Anzahl der Kunden abhängig. Tendenziell gilt, dass der Vorteil eines indirekten Vertriebs mit der Kundenanzahl steigt. Begründen lässt sich dies erneut mit der Höhe der Transaktionskosten. Wie wir auch in Abschn. 2.4.4 zum Thema Intermediäre gesehen haben, wächst das Einsparpotenzial mit der Anzahl der Marktteilnehmer.

Man könnte nun argumentieren, dass sowohl die Verfügbarkeit des Internets als auch die Eigenschaften des Gutes Software den direkten Vertrieb begünstigen. Warum sollte der Anbieter die Software nicht einfach über das Netz direkt an die Kunden vertreiben? Dies ist immer dann problemlos möglich, wenn es sich um eine wenig erklärbungsbedürftige Software handelt, wie z. B. ein Anti-Virenprogramm. Völlig anders sieht es aus, wenn wir an die Einführung einer komplexen Softwarelösung denken, etwa mit umfassenden Funktionalitäten zum Supply Chain Management. In solchen Fällen ist in aller Regel eine umfangreiche Beratung und Unterstützung der Kunden notwendig.

Wir hatten bereits eingangs des Buchs darauf hingewiesen, dass die Softwareindustrie eine internationale Branche ist. Dies hat entsprechende Auswirkungen auf die Gestaltung des Vertriebs. So war ein Ergebnis einer von Lünendonk durchgeführten Studie, dass der indirekte Vertrieb im Ausland am häufigsten über eigene Tochtergesellschaften erfolgt. Am zweithäufigsten wurde der Vertrieb über Kooperationspartner, wie z. B. IT-Berater und Systemhäuser, genannt (Lünendonk 2007, S. 57).

Generell gilt aber, dass sich ein Anbieter nicht unbedingt zwischen einem ausschließlich direkten und einem ausschließlich indirekten Vertrieb entscheiden muss. Vielmehr können in vielen Fällen direkte und indirekte Vertriebswege miteinander kombiniert werden. Dies soll im Folgenden am Beispiel des Vertriebs der SAP AG verdeutlicht werden. So wird der Markt von der SAP und damit auch die Strategie differenziert nach

- Global Enterprises mit mindestens 2.500 Mitarbeitern,
- Local Enterprises mit 1.000–2.499 Mitarbeitern,
- Medium Enterprises mit 100–999 Mitarbeitern und
- Small Enterprises mit 1–99 Mitarbeitern.

Dabei schätzt SAP gemäß einer Investorpräsentation im Januar 2007 den Markt wie folgt ein:

- 20.000 Firmen im Global Enterprise Segment,
- 1.300.000 Firmen im Local und Medium Enterprise Segment,
- 55.400.000 Firmen im Small Enterprise Segment.

Der Großkundenvertrieb der SAP ist direkt ausgestaltet, d. h., es werden in der Regel keine Partner in den Vertriebsprozess eingeschaltet. Die installierte Basis in diesem Segment sorgt für wiederkehrende Umsätze und einen stetigen Strom an Lizenzneinnahmen durch Upgrades oder Relasewechsel sowie Wartung und Pflege.

Da die SAP in dem Kundensegment der großen (Top 500) Firmen bereits sehr hohe Marktanteile besitzt, kann ein zukünftiges Wachstum insbesondere über eine stärkere Positionierung bei mittleren und kleineren Unternehmen erreicht werden. Diese Zielsetzung wird auch mit der Einführung der SaaS-Lösung SAP Business ByDesign verfolgt. Eine besondere Herausforderung besteht in diesem Bereich darin, dass die Marktsegmente große strukturelle Unterschiede aufweisen und die verschiedenen Branchen im Mittelstand zudem unterschiedliche Anforderungen an Unternehmenssoftwarelösungen haben. So erfolgt etwa die Betreuung durch Sales und Consulting im unteren Mittelstand lokal, im gehobenen Mittelstand deutschlandweit oder sogar international. Diese mittelständischen Firmen unterscheiden sich zum Teil erheblich im Hinblick auf ihre IT-Kompetenz, die durch die in Tab. 3.2 dargestellten Merkmale beschrieben werden kann.

Mit einem Konzept voreingestellter Branchenlösungen zu attraktiven Einstiegspreisen sollen in diesen Segmenten neue Kunden gewonnen werden. Hierbei bedient sich SAP in Kooperation mit ihren Partnern einer neuen Vertriebs- bzw. Implementierungsstrategie: Das „Try-Run-Adopt“-Angebot ermöglicht ein kostenloses Ausprobieren der Software.

Tab. 3.2 IT-Kompetenz in mittelständischen Unternehmen

Ressourcen	Kleinerer Mittelstand	Größerer Mittelstand
IT-Abteilung	Keine	Eigene
Erfahrung mit ERP-Software	Keine	Vorhanden
IT-Budget	Keines	Vorhanden
Entscheider bei Software	Geschäftsführer	IT-Abteilung
Prozesskomplexität	Niedrig	Hoch
Anzahl User	Klein	Groß

Zudem werden die Lösungen gemäß dem SaaS-Konzept angeboten (siehe Kap. 7). Seit Ende 2006 bieten erste SAP-Partner ihre Branchenlösungen als Paket an. Dabei werden mit Unterstützung der SAP-Lizenzen und Wartung sowie Implementierung und Betrieb der SAP-Systeme zu einem monatlichen Mietpreis gebündelt, der derzeit deutlich unter 200 € pro Nutzer liegt. Dieses Konzept wird seit geraumer Zeit auch durch entsprechende Werbespots im Fernsehen unterstützt.

Darüber hinaus haben Anbieter bei der Gestaltung des Vertriebssystems Entscheidungen in Bezug auf die

- Tiefe des Vertriebsweges,
- Breite des Vertriebsweges sowie
- Breite des Vertriebssystems

zu treffen (Homburg und Krohmer 2006, S. 877–884).

Dabei bezeichnet die Tiefe des Vertriebsweges die Frage, wie viele Vertriebspartner auf dem Weg zwischen Anbieter und Kunden zwischengeschaltet werden. In der Softwareindustrie ist in aller Regel ein mehrstufiger Vertriebsweg wenig sinnvoll. Die gegebenenfalls mit einem mehrstufigen System einhergehenden Vorteile resultieren überwiegend aus logistischen Überlegungen, insbesondere in Bezug auf die Lagerhaltung. Diese Fragestellungen sind für die Softwareindustrie jedoch kaum relevant.

Die Breite des Vertriebsweges bezieht sich auf die Entscheidung, mit wie vielen Vertriebspartnern ein Anbieter zusammenarbeitet. Dabei gilt grundsätzlich: Je komplexer, erklärmungsbedürftiger und hochwertiger die Produkte sind, umso eher bietet es sich an, mit einer kleinen Anzahl von Partnern zusammenzuarbeiten. Wie bereits erwähnt, führen Softwarehäuser aus diesem Grunde Schulungen durch, um die Qualität ihrer Vertriebspartner zu gewährleisten – und zum Teil auch, um eigene Umsätze in diesem Sektor zu generieren.

Die Breite des Vertriebssystems beschreibt, ob ein bestimmtes Produkt lediglich über einen oder über mehrere Vertriebswege distribuiert wird. Von einem Single-Channel-System wird gesprochen, wenn das Produkt ausschließlich auf einem Weg zum Kunden kommen kann. Demgegenüber liegt ein Multi-Channel-System vor, wenn der Anbieter verschiedene Vertriebswege zum Kunden nutzt. Ein Beispiel hierfür ist die Musikindustrie: So werden Musikalben beispielsweise über den klassischen physischen Handel, über Online-Versender oder auf rein digitalem Weg über Distributoren wie iTunes oder Musicload vertrieben.

Die wesentlichen Zielsetzungen des Aufbaus eines Multi-Channel-Systems bestehen in einer breiteren Marktdeckung sowie der Erreichung verschiedener Kundensegmente. Eine zentrale Herausforderung besteht darin, die Kanäle so zu gestalten, dass diese sich möglichst ergänzen und nicht kannibalisieren. So könnte ein Verlag den Abonnenten einer Tageszeitung in Ergänzung zur Printausgabe verschiedene zusätzliche Online-Funktionalitäten zur Verfügung stellen. Dabei kann es sich um Recherchemöglichkeiten oder multimediale Extras wie Filmdokumentationen zu Schwerpunktthemen handeln. In der Softwareindustrie werden insbesondere wenig erklärungsbedürftige Produkte sowohl direkt über das Internet als auch indirekt über Zwischenhändler vertrieben. Dabei ist die Kannibalisierungsproblematik jedoch weniger ausgeprägt als etwa in der Musikindustrie oder im Verlagswesen, da es dem Softwareanbieter in der Regel relativ egal sein wird, ob er die Umsätze über den direkten oder indirekten Weg erzielt.

Eine wesentliche Herausforderung bei der Gestaltung besteht in einer klaren Abgrenzung der Zielgruppen und Aufgaben der verschiedenen Vertriebskanäle.

3.2.2 Gestaltung der Beziehungen zu Vertriebspartnern und Key Accounts

Im Folgenden untersuchen wir die Gestaltung der Beziehungen eines Softwareanbieters zu seinen Vertriebspartnern und Key Accounts. Der Begriff Key Account bezeichnet hierbei Kunden, in der Regel Unternehmen, die für die Anbieter von besonderer Bedeutung sind und denen daher auch spezielle Leistungen angeboten werden.

Betrachten wir zunächst die Beziehungen eines Softwareanbieters zu seinen Vertriebspartnern. Die Beziehung besteht grundsätzlich, wie in einer physischen Supply Chain, darin, dass der Vertriebspartner beim Hersteller Güter und Leistungen – in unserem Fall Softwarelizenzen – einkauft und diese an seine Kunden weiterverkauft. Im Folgenden stellen wir Vertriebspartnerschaften am Beispiel der SAP vor.

Vertriebspartnerschaften im SAP-Umfeld

Im Partnerumfeld ist der Markt für mittlere Unternehmen stark umkämpft. Um die Qualität der Partner transparent zu machen, hat SAP erstmals auf der Sapphire 2005 ein Konzept verabschiedet, nach dem die Partner in Abhängigkeit eines Punktesystems einen Status (associate, silver oder gold) erhalten (siehe Abb. 3.17). Das Punktesystem bewertet sowohl die Kompetenz als auch die Leistung des Partners. Das wesentliche Kriterium ist der Vertriebserfolg. Doch auch Aspekte wie Kundenzufriedenheit und Ausbildungsmaßnahmen bringen ebenso Bonuspunkte wie Entwicklungen von Branchenlösungen und Add-ons. Das Punktesystem sieht vor, dass rollierend über jeweils vier Quartale die Punkte erreicht sein müssen, die bestimmen, welchen Status der Partner erhält. Der Goldpartnerstatus ist aus mehreren Gründen erstrebenswert. So erhält dieser Partner den höchsten Rabatt auf SAP-Lizenzen und

kann hierdurch im Wiederverkauf die höchste Marge erzielen. Ein immaterieller Vorteil ist die enge Verbindung zur SAP.

Letztlich reflektiert der Partnerstatus die Entwicklung der Geschäftsbeziehung zwischen SAP und dem jeweiligen Partner. Ein hoher Status honoriert, dass das betreffende Unternehmen in Ressourcen und Know-how für die Entwicklung bzw. Implementierung von SAP-Lösungen investiert hat.

Darüber hinaus stellt die SAP ihren Partnern umfassende E-Learning-Angebote zur Verfügung und bietet Vertriebs- und Produktschulungen an, die wiederum Punkte generieren. Weitere konkrete Unterstützungsmaßnahmen umfassen quartalsweise Assessments und Reviews, Strategieworkshops und gemeinsame Marketingaktivitäten.

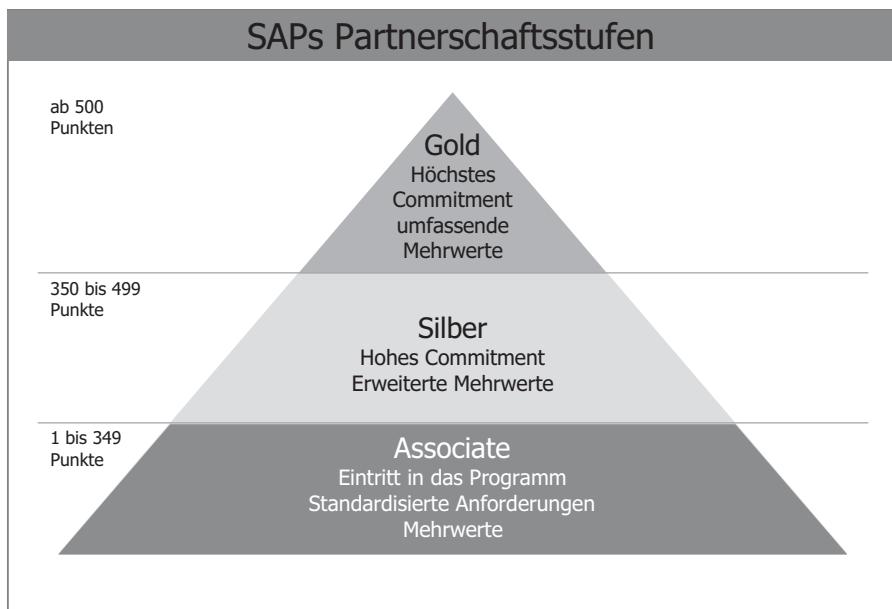


Abb. 3.17 SAPs Partnerschaftsstufen (Quelle: SAP AG)

Sowohl für Softwareanbieter im engeren als auch im weiteren Sinne ist die Betreuung von so genannten Key Accounts von entscheidender Bedeutung. Daher kommt es auch nicht selten vor, dass Vorstandsmitglieder bzw. die Geschäftsführung selbst die Betreuung der Kunden übernehmen. Dabei werden intern häufig auch Entwicklungspläne für die jeweiligen Key Accounts aufgestellt, die beinhalten, welche Umsätze zukünftig mit welchen Lösungen bei den Kunden gemacht werden sollen. Üblicherweise gehen die Rabatte, die diese Key Accounts etwa auf Softwarelizenzen erhalten, deutlich über das normale Maß hinaus.

Ein wichtiges Ziel der Key Account Manager – unabhängig davon, auf welcher Hierarchieebene sie sich befinden – besteht darin, in die Strategieplanung des Kunden ein-

gebunden zu werden. Hierzu gehört, dass etwa Releasewechsel oder der Einsatz neuer innovativer Technologien gemeinsam geplant werden. Auch im operativen Bereich kann eine Abstimmung, etwa im Hinblick auf die Budgetplanung der Kunden, sinnvoll sein. Zur Unterstützung der Aufgaben der Key Account Manager bieten einige CRM-Systeme die Möglichkeit, so genannte „Kundenlandkarten“ anzulegen, die auch Informationen darüber enthalten, welche Mitarbeiter des Kunden dem Anbieter eher wohlgesonnen sind und welche nicht. Diese Information kann insbesondere dann von hoher Bedeutung sein, wenn Probleme kommuniziert werden müssen.

Einer verbesserten Kommunikation sowie einer stärkeren Kundenbindung dient es auch, dass häufig Events für die Key Accounts organisiert werden. Dazu gehören beispielsweise Einladungen zu Sport- und/oder Kulturveranstaltungen.

Bevor wir die Gestaltung von Vertriebsprozessen untersuchen, wollen wir zunächst einige Überlegungen zur Gestaltung der Vertriebsaktivitäten anstellen. Ein wichtiges Instrument hierbei sind Kennzahlensysteme.

3.2.3 Kennzahlensysteme als Instrument für das Vertriebscontrolling in der Softwareindustrie

3.2.3.1 Gegenstände des Vertriebscontrollings

Unter Vertriebscontrolling verstehen wir die zielgerichtete Steuerung und Koordination der Vertriebsaktivitäten eines Unternehmens. Dabei folgen wir der Auffassung, dass es sich bei der Implementierung eines Controllings um mehr als die Kontrolle im Sinne der Ermittlung von Soll-Ist-Abweichungen handelt. Vielmehr wird in der deutschsprachigen betriebswirtschaftlichen Literatur (Weber und Schäffer 2008) die Hauptaufgabe des Controllingsystems in der Koordination des Führungssystems gesehen. Grundlage jeder Koordinationstätigkeit ist die Erzeugung und Verwendung von entscheidungsrelevanten Informationen zur Steuerung von Unternehmen oder Bereichen – in unserem Fall der Vertriebsaktivitäten. Zur Unterstützung der Informationsversorgungsfunktion steht eine Vielzahl von Analyseinstrumenten zur Verfügung (siehe Abb. 3.18).

Analyseinstrument Anwendungsbereich	ABC-Analyse	Portfolio-analyse	Kosten- und Erfolgs-Analyse	Investitions-rechnung	Kennzahlen-/ Kennzahlen- systeme
Informations-versorgung	X	X	X	X	X
Planung	X	X	X	X	X
Kontrolle			X		X

Abb. 3.18 Ausgewählte Analyseinstrumente für das Vertriebscontrolling. (Homburg und Krohmer 2006, S. 1214)

Wir wollen uns in diesem Teil auf die Darstellung der Anwendung von Kennzahlensystemen beschränken, da wir hier einige Besonderheiten für die Vertriebsaktivitäten in der Softwareindustrie herausarbeiten können. Im Gegensatz hierzu ist die Anwendung der anderen Instrumente relativ branchenunabhängig.

3.2.3.2 Anwendung von Kennzahlensystemen in der Softwareindustrie

Kennzahlen dienen allgemein dazu, einen bestimmten Sachverhalt, Zustand oder Vorgang quantitativ zu messen. Mit der Verwendung von Kennzahlen wird in der Betriebswirtschaftslehre das Ziel verfolgt, dem Management entscheidungsrelevante Informationen, insbesondere für Planungs- und Kontrollzwecke, zur Verfügung zu stellen. So eignen sich Kennzahlen dazu, für einen bestimmten Zeitraum Planvorgaben zu erstellen, deren Erreichen am Ende der Periode kontrolliert wird. Treten Plan-Ist-Abweichungen auf, so sind entsprechende Steuerungsmaßnahmen durchzuführen.

In der Betriebswirtschaftslehre wurden hierzu verschiedene Kennzahlensysteme entwickelt, die eine Menge von miteinander in Beziehung stehenden Kennzahlen beinhalten. Der Klassiker unter diesen Kennzahlensystemen ist sicherlich das ROI-DuPont-Kennzahlensystem, das als Beispiel in der folgenden Abb. 3.19 wiedergegeben ist.

Auf dem Grundgedanken der Nutzung von Kennzahlen bzw. Kennzahlensystemen als Controllinginstrument bauen auch moderne Managementsysteme wie die Balanced Scorecard auf.

Während die hier zu Kennzahlensystemen dargestellten Überlegungen weitestgehend anwendungsunabhängig sind, haben Homburg und Krohmer eine Kategorisierung von Kennzahlen für Vertrieb und Marketing entwickelt. Dabei unterscheiden sie zwischen potenzialbezogenen, markterfolgsbezogenen und wirtschaftlichen Kennzahlen einerseits und effektivitäts- bzw. effizienzbezogenen Kennzahlen andererseits (siehe Abb. 3.20).

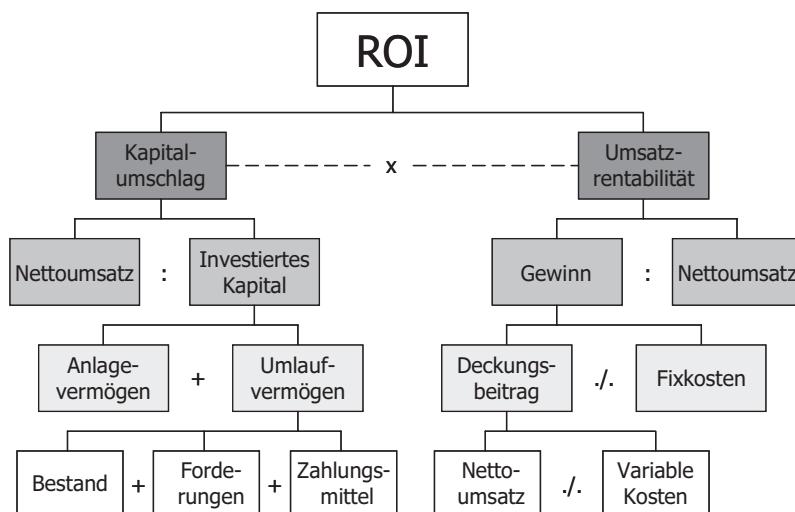


Abb. 3.19 ROI-DuPont-Kennzahlensystem. (Küpper 2008, S. 369)

	Effektivität	Effizienz
	Kategorie I	Kategorie II
potenzial- bezogene Kennzahlen	z. B. <ul style="list-style-type: none"> • Kundenzufriedenheit • Markenimage • Preisimage des Anbieters • Bekanntheitsgrad des Leistungsangebots • Lieferzuverlässigkeit 	z. B. <ul style="list-style-type: none"> • Anzahl erzielter Kontakte / Kosten der Werbeaktion • Kundenzufriedenheit mit der Verkaufsunterstützung / Kosten der Verkaufsunterstützung • Kundenzufriedenheit mit der Lieferbereitschaft / Kosten der Lieferbereitschaft
	Kategorie III	Kategorie IV
markterfolgs- bezogene Kennzahlen	z. B. <ul style="list-style-type: none"> • Anzahl der Kundenanfragen • Anzahl der Gesamtkunden • Anzahl der Neukunden • Anzahl der verlorenen Kunden • Anzahl der zurückgewonnenen Kunden • Marktanteil eines Produkts • am Markt erzieltes Preisniveau • Loyalität der Kunden 	z. B. <ul style="list-style-type: none"> • Anzahl der Kundenanfragen pro Auftrag • Anzahl der Kundenbesuche pro Auftrag • Anzahl der Angebote pro Auftrag (Trefferquote) • Anteil der erfolgreichen Neuprodukteinführungen (Erfolgs- bzw. Floprate) • Anzahl gewonnener Neukunden/Kosten der Aktivitäten der Direktkommunikation
	Kategorie V	Kategorie VI
wirtschaftliche Kennzahlen	z. B. <ul style="list-style-type: none"> • Umsatz • Umsatz bezogen auf Produkt / Produktgruppe • Umsatz bezogen auf Kunden / Kundengruppe • Umsatz aufgrund von Sonderangebotsaktionen • Umsatz aufgrund von Aktivitäten der Direktkommunikation 	z. B. <ul style="list-style-type: none"> • Gewinn • Umsatzrendite • Kundenprofitabilität • Umsatz aufgrund von Rabatten/Kosten in Form von entgangenen Erlösen • Umsatz aufgrund der Messeteilnahme/Kosten der Messeteilnahme

Abb. 3.20 Kennzahlen für das Marketing- und Vertriebscontrolling. (Homburg und Krohmer 2006, S. 1234)

Die hier angegebenen Kennzahlen beziehen sich zwar auf die Bereiche Marketing und Vertrieb, sie sind jedoch branchenunabhängig, d. h., sie können in der Softwareindustrie genauso Anwendung finden wie in anderen Branchen. Die im Folgenden dargestellten Kennzahlen haben sich dagegen speziell für das Vertriebscontrolling in der Softwareindustrie als hilfreich erwiesen (in Klammern ist jeweils die Nummer der Kennzahlen (KPI=Key Performance Indicator) angegeben, die wir in unserem Referenzmodell verwenden, das wir in Abschn. 3.2.4 vorstellen):

- Anzahl von Leads aus dem Direktmarketingprozess (KPI 1),
- Adressvolumen (KPI 2),
- Anzahl Erstbesuche (KPI 3),
- Mit Abschlusswahrscheinlichkeiten bewertete Opportunities (KPI 4),
- Anzahl der Lösungspräsentationen bei den Kunden (KPI 5),
- Anzahl abgegebener Angebote (KPI 6),
- Angebotsvolumen (KPI 7),
- Anzahl der Vertragsverhandlungstermine (KPI 8),
- Anzahl der Abschlüsse (KPI 9),
- Verlorene Opportunities/Angebote (KPI 10) und
- Kundetermine pro Vertriebsmitarbeiter pro Monat (KPI 11).

Die dargestellten Kennzahlen können auch als Grundlage für die variable Vergütung der Vertriebsmitarbeiter herangezogen werden. Folgen wir der Principal-Agent-Theorie, so besteht die Zielsetzung darin, ein anreizkompatibles Vergütungsmodell zu entwickeln, d. h., das Modell ist so auszustalten, dass es den Agent (in diesem Fall Vertriebsmitarbeiter) motiviert, die Zielsetzung der Organisation (Principal) zu verfolgen (siehe Abschn. 2.5 dieses Buches). Ein wesentlicher Bestandteil eines solchen Steuerungssystems ist die Unterteilung des Mitarbeitergehalts in einen fixen und einen variablen Bestandteil. In der folgenden Abb. 3.21 sind einige grundlegende Alternativen dargestellt.

In der Praxis ist das lineare Modell weit verbreitet. Eine sinnvolle Größenordnung für die variablen Gehaltsbestandteile liegt bei etwa 30–40 % des Zielgehaltes. Eine zentrale Fragestellung lautet, welche Parameter als Grundlage zur Bemessung des variablen Gehaltsanteils heranzuziehen sind.

In der Softwareindustrie werden als Zielvorgabe für die variablen Komponenten in erster Linie Umsatzziele oder auch Auftragseingangsziele verwendet. Zu den Fragen, die eindeutig geregelt sein müssen, gehören die Folgenden:

- Welcher Zeitraum des Umsatzes wird als Bemessungsgrundlage verwendet (meist das Geschäftsjahr)?
- Was zählt zur Umsatzzielereichung?

Insbesondere die letzte Frage birgt reichlich Zündstoff. Der heikelste Punkt bei Produkthäusern sind die Regelungen rund um das Wartungsgeschäft. Die Bedeutung dieser Frage

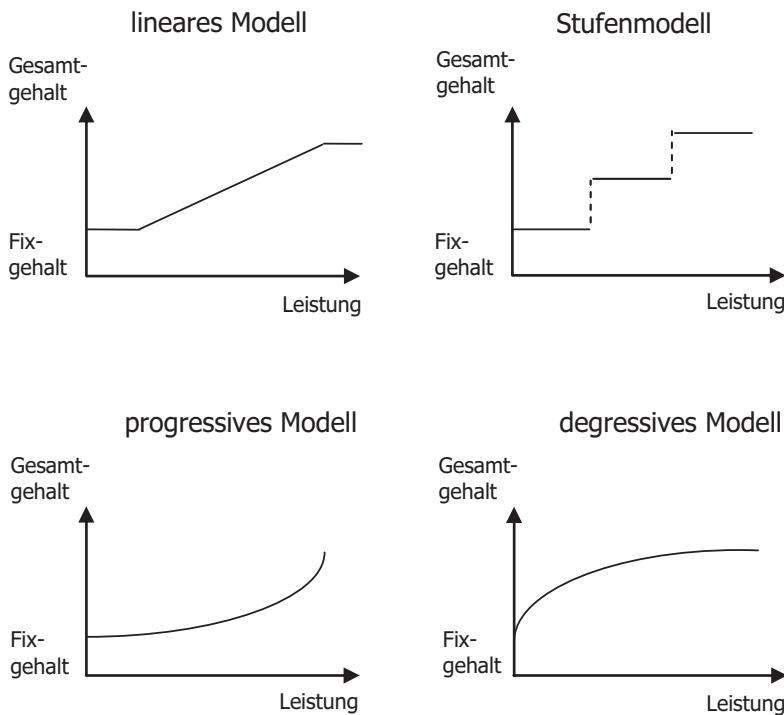


Abb. 3.21 Alternativen leistungsabhängiger Vergütungssysteme. (Homburg und Krohmer 2006, S. 1266)

wird offensichtlich, wenn wir uns erneut klarmachen, dass die Wartungserlöse über einen längeren Zeitraum anfallen und bei Produkthäusern bis zu 80 % des gesamten Jahresumsatzes ausmachen.

3.2.4 Gestaltung der Verkaufsaktivitäten

Unabhängig davon, ob der Vertrieb direkt oder indirekt durchgeführt wird, ist die Gestaltung der Verkaufsaktivitäten ein zentrales Entscheidungsfeld der Vertriebspolitik. Dabei ist es in diesem Bereich in der Regel schwierig, modellbasiert normative Aussagen abzuleiten. Daher gehen wir hier einen anderen Weg und stellen im Folgenden ein Referenzmodell zur Gestaltung von Vertriebsprozessen in der Softwareindustrie vor, das auf der Basis langjähriger Praxiserfahrungen entstanden ist.

3.2.4.1 Der Vertriebsprozess

Die folgenden Abb. 3.22, 3.23, 3.24, 3.25 und 3.26 stellen einen Referenzprozess von der Angebotserstellung bis hin zur Auftragsabwicklung dar, welcher in mehrere Phasen untergliedert ist. Die Abbildungen enthalten neben den Prozessschritten für jede Phase auch Informationen über

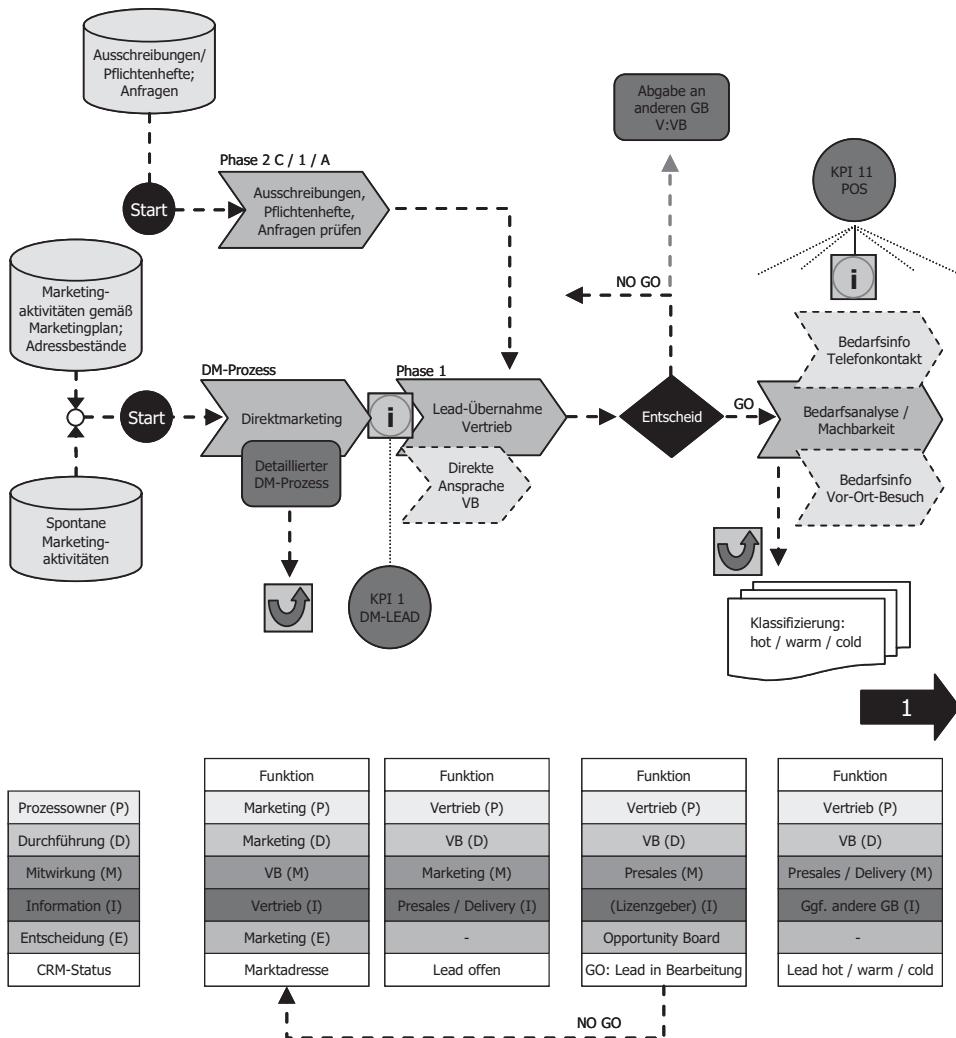


Abb. 3.22 Auslöseimpulse eines Vertriebsprozesses

- den Prozessowner (Marketing oder Vertrieb),
- die für die Durchführung verantwortliche Abteilung bzw. Person (Marketing, Vertriebsbeauftragter (VB) oder Presales),
- die zusätzlich mitwirkenden und unterstützenden Geschäftsbereiche bzw. Personen (VB, Marketing, Presales, Delivery, Management, Rechtsabteilung) und
- die zu informierenden Abteilungen und Personen hinsichtlich der Projektergebnisse (Vertrieb, Presales, Delivery, Lizenzgeber intern/extern, andere Geschäftsbereiche (GB), Rechtsabteilung, Vertriebsleiter (VL), (SL)).

Schließlich gibt die unterste Zeile Aufschluss über den im CRM-System einzupflegenden Stand der aktuellen Akquisebemühung.

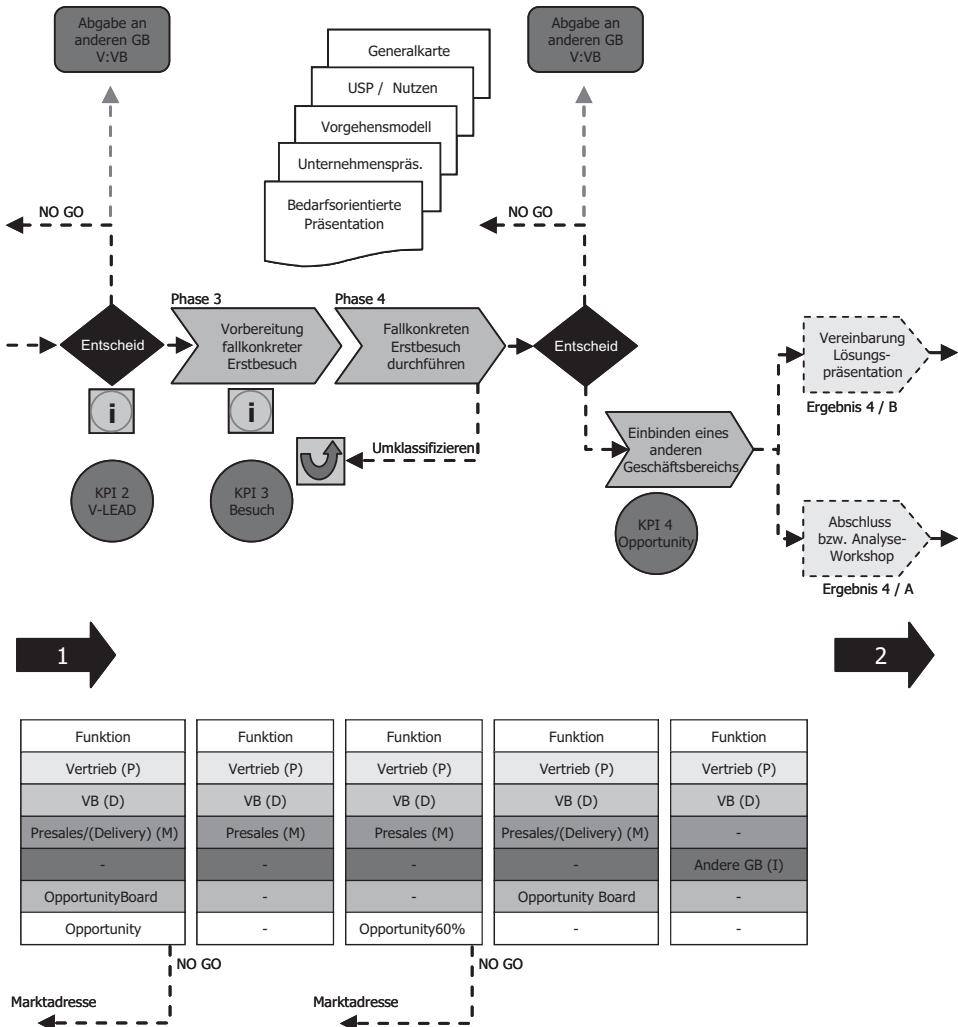


Abb. 3.23 Unternehmensinterne Lösungserarbeitung

Darüber hinaus wird in den fünf Abbildungen, jeweils an den kritischen Prozessphasen, auf die Ermittlung so genannter Key Performance Indicators (KPI) hingewiesen, die wir in Abschn. 3.2.3 bereits genannt haben. Im Folgenden wird nun der Vertriebsprozess anhand der Grafiken kurz skizziert (Reinicke 2007):

Wie in Abb. 3.22 gezeigt, kann der Vertriebsprozess entweder durch Kundeninitiative oder Unternehmensinitiative ausgelöst werden. Danach erfolgt eine Prüfung der Pflichtenhefte und Anfragen bzw. es startet ein Direkt-Marketing-Prozess mit anschließender Lead-Übernahme des Vertriebs. Im Anschluss wird über die Annahme des Projekts entschieden und entweder die Angebotserstellung durch den Vertrieb eingeleitet oder es erfolgt eine Absage an den Kunden. Zu diesem Zeitpunkt wird auch die Machbarkeit durch andere

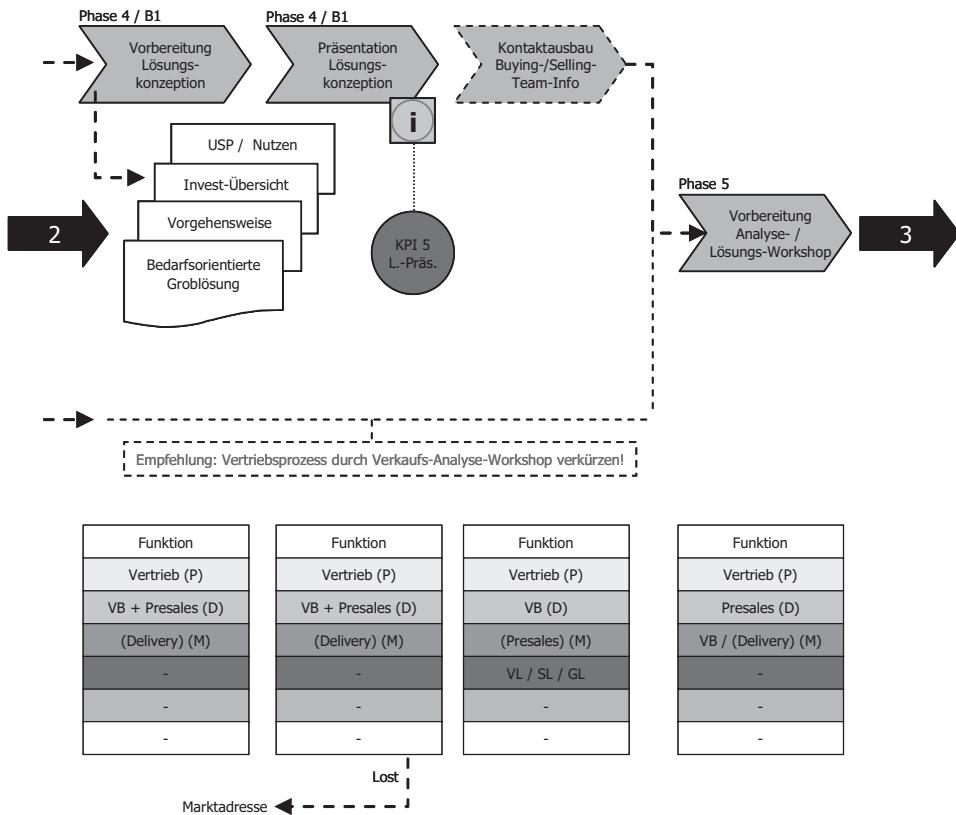


Abb. 3.24 Erstellung und Präsentation der Lösungskonzeption

Geschäftsbereiche geprüft und eine Bedarfs- und Potenzialanalyse beim Kunden erstellt. Diese Phase kann einen längeren Zeitraum in Anspruch nehmen und erstreckt sich mit Beginn des Prozesses in Abb. 3.22 über die Abb. 3.23 und 3.24.

In dieser Phase werden vornehmlich unternehmensinterne Vorbereitungen eines Lösungskonzepts im Hinblick auf den Erstbesuch beim Kunden durchgeführt.

Nach Erstellung des Lösungskonzepts erfolgt die Präsentation beim Kunden zur Analyse der spezifischen Bedürfnisse.

Das daraus resultierende individuelle Lösungskonzept wird im Weiteren zum Feinkonzept ausgearbeitet. Daran anschließend erfolgt die Erstellung eines (Vertrags-) Angebots für den Kunden.

Dieses Angebot wird dem Kunden wiederum präsentiert, um darauf aufbauend Vertragsverhandlungen durchzuführen. Nach erfolgreichem Auftragsabschluss erfolgt die interne Überwachen der Leistungserstellung) mit gleichzeitiger Durchführung einer Win-/Lost-Analyse, bevor es schließlich zum Kunden-Kick-Off kommt.

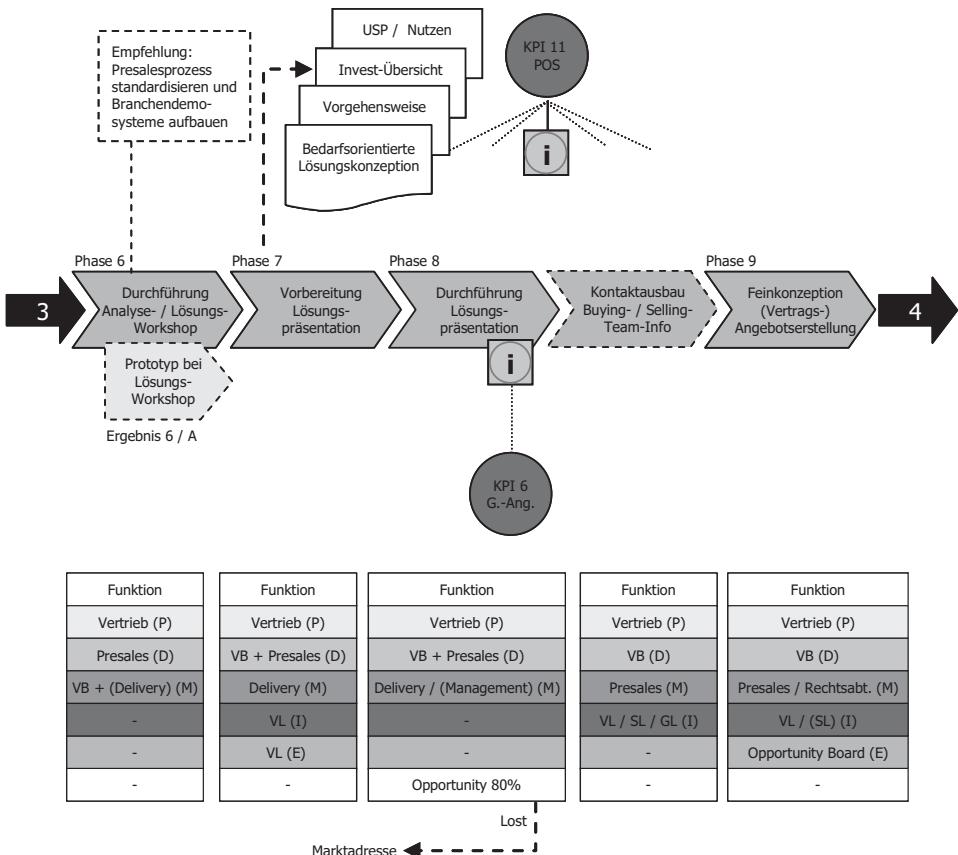


Abb. 3.25 Feinkonzeption und Angebotserstellung

3.2.4.2 Vertriebs-Pipeline

In diesem Abschnitt wollen wir nun auf die Planung der Vertriebsprozesse eingehen. Grundlage hierfür sind die qualitativen und quantitativen Erwartungen des Managements. So sind etwa die Anzahl der Abschlüsse, die Auftragseingangsvolumina aus den Projekten nach Bestands- und Neukunden sowie die Umsatzaufteilung nach Lizenz, Services und Wartung Gegenstand der Planung. Diese Aktivitäten sind insbesondere für börsennotierte Softwareanbieter mit Quartalsreporting von großer Bedeutung.

Daneben haben sich Monats- und Quartalsreviews als nützlich erwiesen, in denen die Vertriebsmannschaft Neues vom Markt und aus der Entwicklungsabteilung des Softwareherstellers erfährt, in denen aber auch aktuelle Verkaufskennzahlen besprochen und gegebenenfalls weitere Maßnahmen festgelegt werden.

Ein in der Praxis weit verbreitetes Planungsinstrument ist hierbei der so genannte „Sales Funnel“, der in der folgenden Abb. 3.27 dargestellt ist.

Ausgehend von den Unternehmenszielen, wie Umsätze oder Anzahl von Lizenzverkäufen, kann rückwärtsgerichtet anhand des „Sales Funnels“ ermittelt werden,

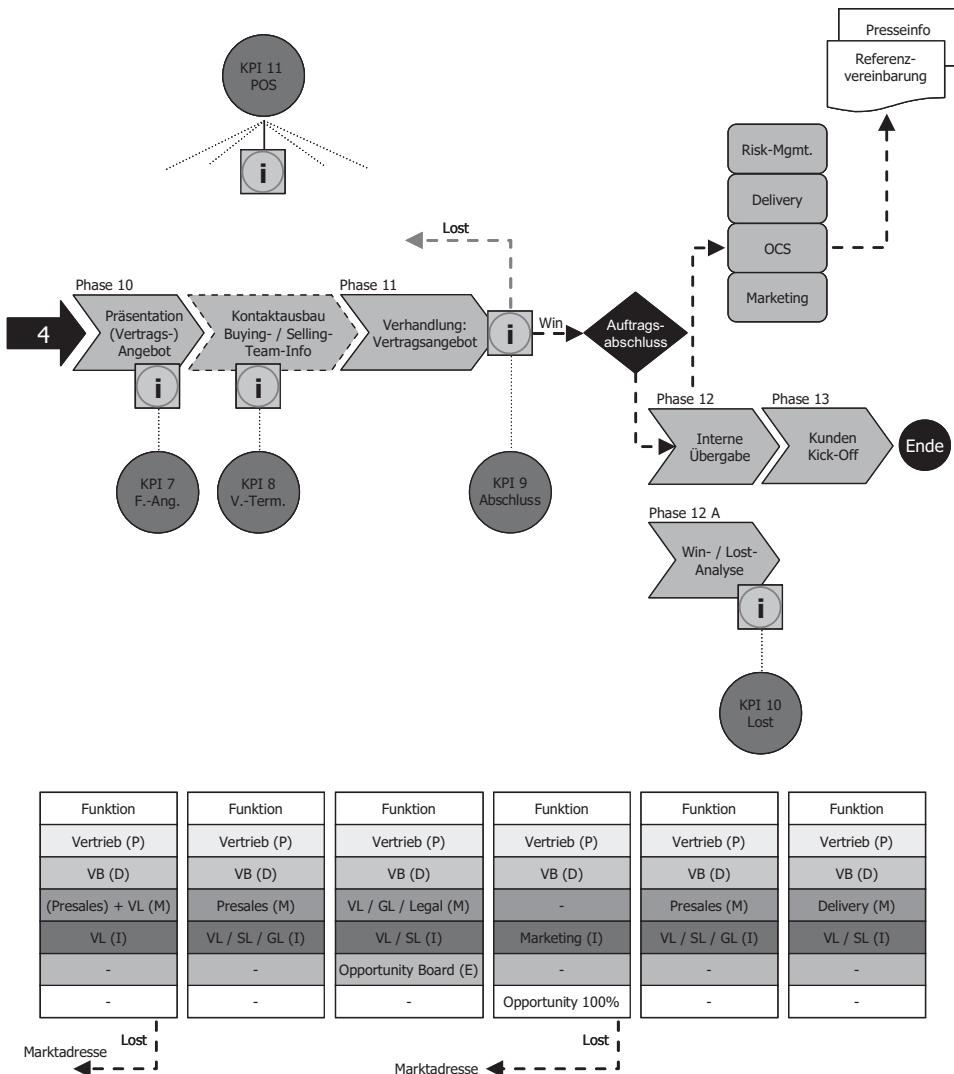


Abb. 3.26 Verhandlungsphase und Kunden-Kick-Off

- wie viele Projekte gewonnen werden müssen, um das Ziel (Umsatz, Anzahl verkaufter Lizzenzen) zu erreichen,
- wie viele Opportunities in den einzelnen Erfolgskategorien vorhanden sein müssen,
- wie viele Leads benötigt werden und
- wie viele Potenzialadressen hierfür erforderlich sind.

Der in der Abbildung dargestellte „Sales Funnel“ basiert auf Erfahrungen der Fujitsu TDS GmbH, die insbesondere mittelständische Kunden hat. Die Anwendung des Konzeptes ist natürlich auch für andere Bereiche möglich; in diesem Fall sind jedoch andere Zahlen in den Trichter einzubeziehen.

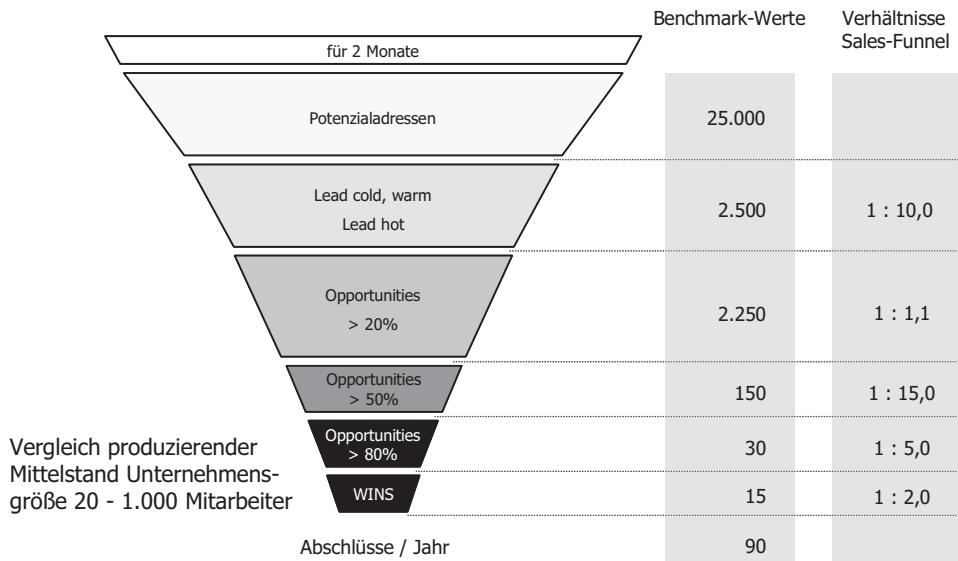


Abb. 3.27 Vertriebspipeline – „Sales Funnel“ im Benchmarkvergleich

3.2.5 Erschließung internationaler Märkte

3.2.5.1 Spezifische Herausforderungen für Softwareunternehmen

Die Anforderungen von Unternehmen zur Software-Unterstützung sind in vielen Feldern recht ähnlich – so erwartet ein italienisches Unternehmen sicherlich ähnliche Dinge von einer Textverarbeitungssoftware wie ein Unternehmen aus Brasilien oder aus Russland. Gleches gilt bezüglich Software für Konsumenten. Zudem lässt sich Software, anders als ein physisches Gut, sehr einfach über das Internet an einem beliebigen Ort bereitstellen. Darüber hinaus existieren in der Softwareindustrie – wie in diesem Buch bereits mehrfach angesprochen – Netzeffekte. All dies legt nahe, dass sich Software-Unternehmen stark um die Erschließung internationaler Märkte bemühen sollten.

3.2.5.2 Nischenstrategie zur Erschließung internationaler Märkte

Um den Schritt ins Ausland zu wagen, stützen sich Softwareunternehmen mit einer Nischenstrategie typischerweise auf etablierte Prozesse im Inland und etablierte Kooperationsbeziehungen im Ausland. Häufig handelt es sich bei diesen Partnern um große und bereits international tätige Unternehmen. Diesem Partner schließt man sich an, indem man beispielweise dessen physische und informelle Infrastrukturen nutzt, um erste internationale Kontakte zu knüpfen. Es ist auch denkbar, gemeinsam mit dem Partnerunternehmen erste Aufträge im Ausland anzunehmen, um anschließend dort eigenständig aufzutreten – gelegentlich wird dies plastisch auch als Huckepack-Ansatz bezeichnet. Wählt ein Softwareunternehmen die Nischenstrategie, dann ist es meist durch kaufmännische Vorsicht sowie dem festen Willen zur unternehmerischen Selbstständigkeit geprägt.

Unternehmen, die diesen Weg der Internationalisierung gehen, spielen ihren Know-how-Vorteil aus. Typischerweise lässt sich dieser Vorteil von Konkurrenten auch nicht so schnell aufholen. Vorteilhaft ist auch, dass die Wettbewerbsintensität in Nischenmärkten oft geringer ist als in Massenmärkten. Gleichwohl ist das Potential einer marktischen definitionsgemäß beschränkt.

Im Ergebnis kann aus einer Nischenstrategie ein langsamer aber stetiger Internationalisierungsprozess hervorgehen. Eine Reihe von B2B-Anbietern aus Deutschland haben diesen Weg gewählt, oft wenig sichtbar für die breite (Fach-)Öffentlichkeit. Typisch für diese Unternehmen ist u. a. dass sie von dem Willen geprägt sind, kein Fremdkapital ins Unternehmen zu holen. Auch agieren sie grundsätzlich eher vorsichtig.

Nischenstrategie eines deutschen Anbieters von Sicherheitssoftware

Das Unternehmen wurde in den 90er Jahren in Deutschland gegründet und ist mit ca. 300 Mitarbeitern in der Branche für Sicherheitssoftware tätig. Die Firma bedient ein hochspezialisiertes Nischensegment im B2B-Bereich der Hochsicherheitstechnik und Kryptologie. Dabei werden individuelle Softwarelösungen (teilweise gekoppelt mit Hardware) entwickelt. Zu den Kunden zählen sowohl Privatunternehmen als auch Regierungen. Aufgrund des Geschäftsfeldes ist langjähriges Vertrauen von zentraler Bedeutung für eine erfolgreiche Zusammenarbeit und für die Akquise von neuen Kunden. Zu den ersten Kunden gehörten unter anderem das Auswärtige Amt sowie weitere Bundesbehörden.

Das B2B-Unternehmen verfolgte zunächst eine klassische Nischenstrategie im Inland, bevor es sich für die Internationalisierung entschied. Das Unternehmen erarbeitete ein strategisches Vorgehen, genaue Planung und detaillierte Analysen der potentiellen Zielmärkte. Die Länderauswahl erfolgte auf Basis umfassender Marktstudien mit einer detaillierten Nutzen-/Risikoabwägung. Zunächst wurden interessante Zielmärkte identifiziert und geeignete Mitarbeiter dorthin entsandt. Denn es hat sich als strategisch sinnvoll erwiesen, Kooperationen mit lokalen Unternehmen vor Ort einzugehen. Die Geschäftstätigkeit innerhalb einer spezialisierten Nische erleichtert dabei die Kontaktaufnahme zu entsprechenden Partnerunternehmen, da sich die relevanten Akteure oftmals direkt oder indirekt kennen. Über diese Ansprechpartner werden Kontakte zu ersten ausländischen Kunden und Regierungen hergestellt. Bei dem gesamten Prozessverlauf von der ersten Kontaktaufnahme zum Networking vor Ort und dem Finden von geeigneten Kooperationspartnern greift das Unternehmen auf Ressourcen wie Know-how, Netzwerke, Personen und Niederlassungen von einem großen und etablierten Partnerunternehmen zurück.

Trotz des ausländischen Wettbewerbs mit langjährig etablierten und regierungsnahen Konkurrenzunternehmen, gelang es dem Unternehmen in ausländischen Märkten EU-weite Ausschreibungen zu gewinnen. Dabei spielte nicht nur der Zugang zu den Ressourcen des Partners eine entscheidende Rolle, sondern auch

die Qualität der Softwareprodukte, die bis dahin erlangte Reputation und der Preis. Diese Internationalisierungsstrategie bedeutete für das Softwareunternehmen, dass ausländische Märkte nicht schnell angegriffen werden konnten, sondern dass es sich Schritt für Schritt an internationale Märkte annähern musste. Dies ging aber mit stetigem Unternehmenswachstum einher.

Best Practice Beispiel aus Picot et al. (2014)

3.2.5.3 Breitenstrategie zur Erschließung internationaler Märkte

Besonders Softwareunternehmen, die in Märkten agieren, welche durch starke Netzefekte geprägt sind, wenden die Breitenstrategie zur Erschließung internationaler Märkte erfolgreich an. Denn gerade wenn der Nutzen eines Gutes durch die steigende Anzahl von weiteren Nutzern beeinflusst wird, ist man gezwungen, schnell international zu denken. Gelegentlich hat man sogar die Chance, einen Standard zu setzen und sich so zumindest über eine gewisse Zeit eine komfortable Marktposition aufzubauen.

Softwareunternehmen mit einer Breitenstrategie richten beispielsweise von Anfang an ihren Fokus auf standardisierte Produktkomponenten. Lokale Adaptionen an den jeweiligen Markt oder sprachliche Anpassungen werden ohne Veränderung des Kernprodukts vorgenommen. Teilweise werden sogar ganze Internationalisierungsprozesse vereinheitlicht, so dass man jedes Land möglichst auf die gleiche Art und Weise angehen kann. Somit können Softwareunternehmen, die eine Breitenstrategie verfolgen, in mehreren Märkten parallel internationalisieren. In den Phasen, in denen das Unternehmen stark auf die Internationalisierung setzt, werden verstärkt Outsourcing, Offshoring und flexible Beschäftigungsmodelle (z. B. Praktikanten und Freelancer) genutzt. Teil der Breitenstrategie ist es folglich, Nicht-Kernaufgaben auszulagern und sich auf die Kernkompetenz zu fokussieren. Insbesondere für B2C-Unternehmen, die auf eine schnelle Internationalisierung setzen, ermöglichen Standorte in Großstädten einen guten Zugang zu qualifizierten und international orientierten Mitarbeitern. Bei der Breitenstrategie handelt es sich folglich um eine Internationalisierungsstrategie mit hoher Geschwindigkeit, die häufig bei B2C-Unternehmen aus den USA zu beobachten ist.

Die Zusammenarbeit mit Venture-Capital-Gebern ist ein typisches Merkmal der Breitenstrategie. Diese Finanzierungsform ist besonders im Silicon Valley beobachtbar, erfreut sich aber auch in anderen Regionen zunehmender Beliebtheit. In diesem Finanzierungsansatz steht das Erzielen von positiven Cashflows nicht unbedingt im Fokus. Stattdessen wird auf ein schnelles und aggressives Wachstum in den ersten Jahren gesetzt - manche Unternehmen verfolgen dabei auch das Ziel eines gewinnbringenden Exits, also den Verkauf des Unternehmens an einen Konsolidierer am Markt. Jedoch ist nicht nur die Verfügbarkeit von Kapital Teil der Internationalisierungsstrategie. Für Softwareunternehmen im B2C-Bereich, die eine Breitenstrategie verfolgen, spielt vor allem auch die qualitative Dimension des verfügbaren Kapitals eine zentrale Rolle. Das bedeutet, dass die Kapitalgeber sowohl als Treiber der Internationalisierung als auch als Ratgeber fungieren. Man spricht in diesem Zusammenhang auch von Smart Money. Die Softwareunternehmen bekommen

durch ihre Financiers Zugang zu einem breiten Netzwerk an erfahrenen Experten, Zugang zu Kunden, Mitarbeitern und Kooperationspartnern in ausländischen Zielmärkten.

Best Practice – Breitenstrategie für B2C-Unternehmen

Das B2C-Unternehmen wurde nach der Jahrtausendwende in Kalifornien gegründet und ist mit ca. 50 Mitarbeitern in der App-Branche tätig. Das Geschäftsmodell basiert auf der Bereitstellung einer Applikation, die eine verbesserte User Experience bei der Betrachtung von Fotos und Videos ermöglicht. Dabei ist die Anwendung sowohl auf mobilen Endgeräten als auch auf dem Desktop möglich. Die B2C-Firma hatte von Anfang an den Anspruch ein Problem zu lösen, welches Menschen weltweit betrifft, wenngleich der Erfolg im Heimatmarkt Priorität für das Unternehmen hat. Dieser darf durch die Konzentration auf andere Länder nicht gefährdet werden. Aktuell befinden sich 40 % der Kunden außerhalb der USA. Die App wurde von Beginn an so programmiert, dass es leicht war, sie an internationale Märkte anzupassen. So konnte die App zum Beispiel innerhalb von 24 h auf chinesische Kunden umgestellt werden.

Das aggressive Wachstum wird vor allem durch diverse Venture-Capital-Firmen ermöglicht. Die Investoren werden nicht nur als Geldgeber wahrgenommen, sondern viel mehr als Partner und Mentoren. Das B2C-Unternehmen profitiert von deren Netzwerken und Wissen, was bei der Erschließung neuer Partnerschaften in unbekannten Ländern von großem Vorteil war. Die Strategie zur Realisierung von Partnerschaften sieht so aus, dass zunächst eine User Base im jeweiligen Land aufgebaut wird. Das geschieht durch den Download der App im App Store. Wenn eine gewisse Anzahl an Kunden generiert wurde, verfügt das Unternehmen über eine gute Verhandlungsposition mit den Partnern. So konkurrierte das Unternehmen zunächst mit Anbietern wie Picasa, Instagram und Flickr. Diese Firmen wurden jedoch im Laufe der Zeit zu Partnern, indem ihre Angebote in den Service integriert wurden. Für die Serviceintegration erwartete das B2C-Unternehmen im Gegenzug Marketingaktivitäten vom jeweiligen Partner. So entstand eine Win-Win-Situation, die zu einem Kundenzuwachs für beide Partner führte.

Abgesehen von dieser Vorgehensweise greift das Unternehmen bei der Internationalisierung auch verstärkt auf das Netzwerk der Venture-Capital-Geber zurück. So ist beispielsweise auch eine strategische Partnerschaft zu Google entstanden. Diese Kooperation zeichnet sich dadurch aus, dass Googles Betriebssystem Android in den Werkseinstellungen eine Applikation beinhaltet, die es ermöglicht, die Software des Unternehmens direkt anzuwenden. Gerade diese wertvolle strategische Partnerschaft ist auf Smart Money zurückzuführen. Die Breitenstrategie bedeutet folglich für das B2C-Unternehmen, dass ausländische Märkte schnell und aggressiv angegriffen werden. Die Internationalisierung wird nicht nur durch das Unternehmen selbst, sondern auch durch die beteiligten Kapitalgeber vorangetrieben.

Best Practice Beispiel aus Picot et al. (2014).

Tab. 3.3 Gegenüberstellung Nischen- und Breitenstrategie

	Nischenstrategie	Breitenstrategie
<i>Zielkunden</i>	Unternehmen	Konsumenten
<i>Strategie</i>	Tief in eine Nische im Heimatmarkt vorstoßen, Expertise sammeln und Schritt für Schritt internationale Märkte angehen	Schnelles und aggressives Wachstum in mehreren Ländern gleichzeitig, Kunden in einem breiten Markt adressieren
<i>Produkt</i>	Technisch ausgereifte und ggf. individuell angepasste Softwarelösungen	Standardisierte Produkte
<i>Zielgruppe im Ausland</i>	Ausgewählte Kunden	Massenmarkt
<i>Kontakte im Ausland</i>	Über Partnerunternehmen, mit denen man „Huckepack“ ins Ausland geht	Über Kapitalgeber, die durch ihre Erfahrung (Smart Money) bereits Kontakte ins Ausland aufgebaut haben
<i>Finanzierung</i>	Meist eigenfinanziert	Meist durch Venture Capital finanziert

In der folgenden Tabelle ist die Breitenstrategie der Nischenstrategie nochmals gegenübergestellt (Tab. 3.3).

3.2.5.4 Weitere Ansatzpunkte zur Erschließung internationaler Märkte

Softwareunternehmen, die ihre Produkte oder ihre Dienstleistungen von Beginn an für ausländische Märkte entwickeln, richten ihre Produktentwicklung schon sehr früh international aus – trotz mancher kurzfristiger Nachteile. Dabei bedeutet eine internationale Produktausrichtung beispielsweise, Variablennamen und Kommentare im Programmcode direkt in englischer Sprache zu verfassen und die in Masken angezeigten Texte variabel zu halten. Wird dies beachtet, dann sind wichtige Hürden für den Weg aus dem heimatlichen Sprachraum genommen. Ferner werden Fehler bei einer nachträglichen Übersetzung vermieden. Für Softwareunternehmen, die in einem nicht-englischsprachigen Land beheimatet sind, empfiehlt sich ferner Englisch als (zweite) Unternehmenssprache zu etablieren. Dies kann Vorteile bei dem Schritt über die Grenzen des eigenen Sprachraums bieten, kann aber auch bei der Rekrutierung von Mitarbeitern hilfreich sein.

Um eine erfolgreiche Internationalisierungsstrategie voranzutreiben, sind für Unternehmen in der Softwareindustrie zudem multikulturelle Teams hilfreich. Ein derartiges Team vereinfacht den Zugang zum Weltmarkt, da einzelne Sprach- und Kulturrestrieren nicht erst überwunden werden müssen. Ferner können Kontakte ins Ausland schneller geknüpft und internationale Netzwerke schneller aufgebaut sowie effektiver genutzt werden. Auch sind die spezifischen Anforderungen aus internationalen Märkten so einfacher zu erfassen.

Ländermarken können als Katalysator für die eigenen Bemühungen um die Internationalisierung genutzt werden. So weisen deutsche Softwareunternehmen, die die Herkunft ihrer Produkte und Dienstleistungen betonen („Software Made in Germany“) einen höheren Internationalisierungsgrad auf als solche, die darauf verzichten (Picot et al. 2011).

Die „Marke Deutschland“ scheint also im Ausland ein Qualitätssiegel zu sein, das auf die deutsche Softwareindustrie ausstrahlt.

3.2.5.5 Die Gefahr der Komfortzone

Man kann beobachten, dass sich Softwareunternehmen aus kleinen Ländern wie Israel oder den skandinavischen Staaten stark um die Erschließung internationaler Märkte bemühen. US-amerikanische Softwareanbieter haben hier einen Wettbewerbsvorteil, da ein derart großer Wirtschaftsraum aufgrund fehlender Sprachbarrieren zu bedeutend geringeren Koordinationskosten führt. Anders sieht das aus, wenn der Heimatmarkt klein ist. Softwareunternehmen aus kleineren Ländern verspüren daher einen gewissen Druck, ihre Produkte international zu vertreiben, um ausreichend Kunden zu gewinnen – für das Überleben reicht der Heimatmarkt häufig nicht aus. Oft pflegen Softwareunternehmen aus kleineren Ländern auch gute internationale Kontakte, die einen frühen Markteintritt in andere Wirtschaftsräume erleichtern.

Ambivalent ist dagegen der Rahmen für Softwareunternehmen aus Ländern mittlerer Größe. In vielen Segmenten bietet ein mittlerer Heimatmarkt eine ausreichende Größe, um sich erfolgreich am Markt zu positionieren. Eine aktuelle Studie bestätigt diese Beobachtung bezüglich deutscher Softwareunternehmen. Diese sehen den Heimatmarkt häufig als ausreichend komfortabel an und internationalisieren weit weniger stark – unabhängig davon, wie erfolgreich das Unternehmen auf dem deutschen Markt agiert (vgl. Picot et al. 2014). Bei hoch spezialisierten Anbietern kann dies erfolgreich sein, auch wenn dann nennenswerte Potentiale ungenutzt bleiben. Bei Anbietern eher breit ausgelegter Software ist die Fokussierung auf den Heimatmarkt sehr gefährlich.

3.3 Preisstrategien

3.3.1 Grundüberlegungen

Die Preisgestaltung nimmt in der Strategie der meisten Unternehmen eine herausragende Rolle ein (Simon 1992, S. 7; Diller 2008, S. 21–22; Rullkötter 2008, S. 93). Sie bestimmt unmittelbar die Umsatzhöhe und somit langfristig auch die erreichte Rendite. Bei Fehlentscheidungen können Reputation und Kundenbeziehungen des Unternehmens gefährdet sein. Trotz dieser hohen Bedeutung wird bezüglich der Preisgestaltung häufig eine Vielzahl an Defiziten festgestellt, darunter auch Rationalitätsdefizite in Form von ad hoc oder willkürlichen Entscheidungen (Florissen 2008, S. 85). Nicht nur in kleinen und mittelständischen Unternehmen werden Preisentscheidungen häufig aus dem Bauch herausgetroffen. Dabei kann eine Fundierung der Preisstrategie auf Basis empirischer Daten ökonomisch sehr sinnvoll sein: Studien zeigen, dass sich eine gut platzierte Preisanpassung in der Regel stärker als eine Kostenreduzierung auf den Gewinn auswirkt. So kann eine Preisanpassung von nur 1 % zu einem Anstieg des operativen Gewinns von ca. 8 % führen (Marn et al. 2003).

Traditionelle Preiskonzepte lassen sich allerdings nicht ohne Weiteres auf Softwareprodukte anwenden (Bontis und Chung 2000, S. 246). Von den in Abschn. 2.1 dargestellten Merkmalen des Gutes Software ist insbesondere die Eigenschaft der quasi kostenfreien Vervielfältigung für die Preissetzung von Bedeutung. Zur Erinnerung: Die Entwicklung der First Copy eines digitalen Gutes führt in der Regel zu relativ hohen Kosten. Im Gegensatz hierzu gehen die variablen Kosten für jede weitere Kopie jedoch gegen Null. Was bedeutet dies für die Preissetzung? Zunächst scheint es klar zu sein, dass eine kostenorientierte Preissetzung nicht in Frage kommt. Vielmehr ist es sinnvoll, eine nachfrage- oder wertorientierte Preissetzung zu implementieren. Das bedeutet, dass die Anbieter ihre Preise an den Zahlungsbereitschaften ihrer potenziellen Kunden orientieren sollten. Shapiro und Varian beschreiben diesen Zusammenhang wie folgt: „cost-based pricing just doesn't work [...]. You must price your information goods according to consumer value, not according to your production cost“ (Shapiro und Varian 1998, S. 3).

Grundsätzlich ermöglicht die Kostenstruktur von digitalen Gütern auch die Implementierung von Niedrigpreisstrategien. Dies kann beispielsweise sinnvoll sein, wenn ein Softwareanbieter einen etablierten Anbieter auf einem Netzeffektmarkt verdrängen möchte. Da die variablen Kosten der Software gegen Null gehen, wird der Deckungsbeitrag, selbst wenn die Software verschenkt wird, nicht negativ. Anders sieht dies aufgrund der variablen Kosten bei physischen Produkten aus. Im Weiteren werden wir sehen, dass etwa auch die Preisbündelung durch die Kostenstruktur digitaler Güter begünstigt wird.

Allerdings führt die Annahme, dass Software aus ökonomischer Perspektive und speziell für die Preissetzung wie jedes andere digitale Gut zu betrachten ist, in die Irre. Mag es für einen Softwareanbieter, dessen Erlöse sich größtenteils oder ausschließlich aus dem Verkauf von Lizenzn zusammensetzen, noch sinnvoll sein, die Regeln für digitale Güter weitgehend anzuwenden, so lässt sich dies nicht verallgemeinern. Vielmehr fallen bei der Erbringung von Beratungs- und Supportdienstleistungen sehr wohl Kosten an.

Im Folgenden geben wir nun einen Überblick über die möglichen Parameter der Preisgestaltung für Softwareprodukte sowie deren verschiedenen Ausprägungen.

3.3.2 Parameter der Preisgestaltung für Softwareprodukte

3.3.2.1 Überblick

Die Gestaltungsmöglichkeiten für Softwareprodukte sind vielfältig. Im Laufe der Zeit haben sich grundlegende Änderungen der Preismodelle für Software entwickelt. Während in Zeiten der Großrechner überwiegend eine Bepreisung anhand der Rechenleistung erfolgte, haben sich in der Vergangenheit benutzerorientierte Preismodelle („Lizenzmodelle“) durchgesetzt (Bontis und Chung 2000, S. 247–248). Heute bieten Softwareanbieter verstärkt auch nutzungsabhängige Preismodelle an, die wir in Abschn. 3.3.2.3 noch einmal detailliert aufgreifen werden.

Da es kein allgemeingültiges Preismodell für Softwareanbieter gibt (Bontis und Chung 2000, S. 246) und sich Preismodelle aus mehreren Elementen zusammensetzen können, stellen wir im Folgenden verschiedene Preisparameter für Softwareprodukte vor.

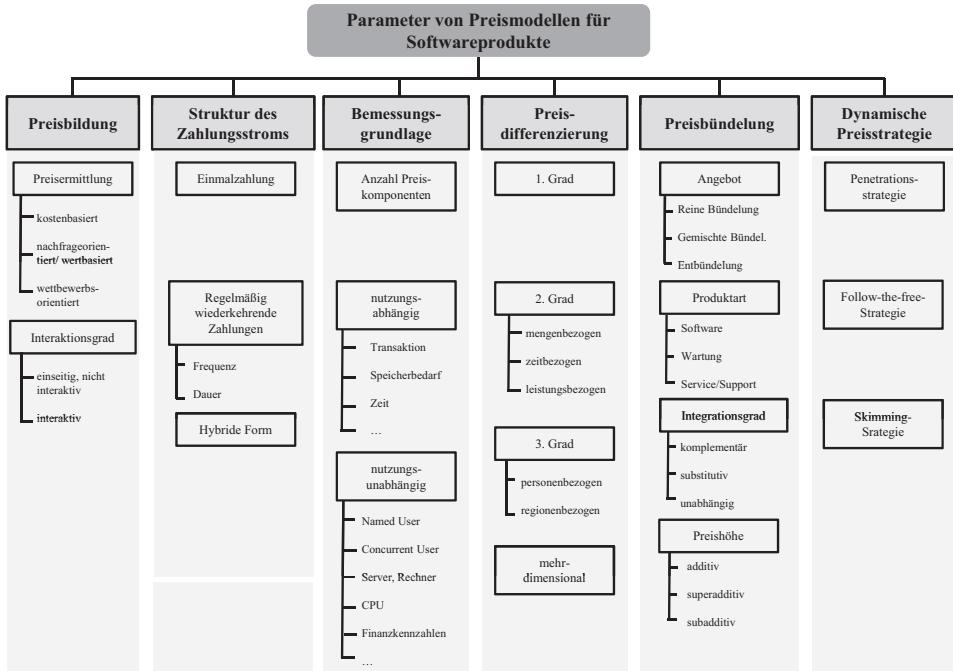


Abb. 3.28 Parameter von Preismodellen für Softwareprodukte

Abb. 3.28 zeigt die im Weiteren betrachteten Parameter der Preismodelle im Überblick (Lehmann und Buxmann 2009).

Die Preismodelle der Softwareanbieter bestehen in der Regel aus einer Kombination der verschiedenen Parameter. Dabei kann das Preismodell auch mehrere Unterpunkte aus jeder Spalte beinhalten.

3.3.2.2 Preisbildung

Bei der Preisbildung legen die Anbieter fest, wie die Höhe der Preise gebildet werden soll. Dabei sind zum einen die Basis der Preisermittlung und zum anderen der Interaktionsgrad zu berücksichtigen.

Für die Preisermittlung kommen prinzipiell drei Formen in Betracht (Homburg und Krohmer 2006, S. 720; Nieschlag et al. 2002, S. 810–814):

- kostenbasiert,
- nachfrageorientiert bzw. wertorientiert und
- wettbewerbsorientiert.

Nach dem Ansatz der kostenorientierten Preisermittlung wird die Höhe des Preises mit Hilfe der Kostenrechnung ermittelt (Diller 2008, S. 310–311). Für Softwarelizenzen gilt jedoch genauso wie für digitale Güter, dass diese Form der Preisermittlung aufgrund der

besonderen Kostenstruktur von geringer Bedeutung ist. Demgegenüber kann die Preisermittlung für SaaS-Lösungen unter Einbeziehung der entstehenden Kosten durchaus sinnvoll sein.

Die nachfrageorientierte bzw. wertorientierte Preisermittlung (Value Based Pricing) orientiert sich im Wesentlichen an der Nachfrage für das Produkt (Homburg und Krohmer 2006, S. 720–721). Dabei sind die Wertschätzungen der Kunden für das Produkt von Bedeutung und nicht, wie bei der kostenorientierten Preisermittlung, die Kosten des Produkts (Harmon et al. 2005, S. 1).

Bei der wettbewerbsorientierten Preisermittlung wird die Preisbildung an den Preisen und dem Verhalten der Wettbewerber ausgerichtet (Homburg und Krohmer 2006, S. 747). Die Attraktivität von Konkurrenzprodukten für Kunden ist u. a. von der Homogenität der Produkte und der Struktur des Marktes abhängig (Nieschlag et al. 2002, S. 813). In der Softwareindustrie ist aufgrund von Netzeffekten und den daraus resultierenden Lock-in-Effekten bei Kunden die Gewinnung eines großen Marktanteils für die Anbieter von entscheidender Bedeutung, insbesondere wenn sich das Produkt nicht oder nur geringfügig von den Produkten der Wettbewerber unterscheidet. Daher spielt die wettbewerbsorientierte Preisermittlung neben der nachfrageorientierten Preisermittlung für Softwareprodukte eine wichtige Rolle.

Ein weiterer Parameter der Preisbildung ist der Grad der Interaktion. Nicht-interaktive Preisbildung ist die einseitige Festlegung des Preises durch den Anbieter ohne Einflussnahme des Kunden. Bei interaktiver Preisbildung kommt der Preis demgegenüber mittels Interaktion von Kunde und Anbieter zustande. Beispiele für interaktive Preisbildung sind Verhandlungen oder (Internet-) Auktionen (z. B. Schmidt et al. 1998). Jedoch sind Auktionen für digitale Güter und daher auch für Software in der Regel ökonomisch wenig sinnvoll (Shapiro und Varian 1999, S. 23).

3.3.2.3 Struktur des Zahlungsstroms

Bei der Gestaltung des Preismodells für Software gibt es grundsätzlich zwei Varianten: Der Kunde kann eine einmalige Zahlung leisten und hiermit das Recht auf eine zeitlich unbegrenzte Nutzungsdauer der Software erwerben oder das Preismodell sieht regelmäßig wiederkehrende Zahlungen des Kunden vor. Natürlich ist auch die Kombination beider Varianten möglich (Kittlaus et al. 2004, S. 82).

Die Einmalzahlung entspricht dem heute weitverbreiteten Modell der Lizenzierung einer Software. Durch den Kauf einer Lizenz erwirbt der Kunde ein in der Regel zeitlich unbegrenztes Nutzungsrecht.

Die Gestaltungsmöglichkeiten regelmäßig wiederkehrender Zahlungen liegen sowohl in der Frequenz als auch in der Dauer der Zahlungen. So könnte ein monatlicher oder jährlicher Abonnementpreis zum Beispiel über einen Zeitraum von zwei Jahren als Preismodell für die Nutzung einer Software vereinbart werden. Anwendung finden diese Preismodelle vor allem in Software-as-a-Service-Lösungen (Cusumano 2007, S. 20), wo Kunden für den Zeitraum der Zahlung (teilweise wird auch von Abonnement- oder Miet-Modellen gesprochen) über das Internet die Software des Anbieters nutzen (Buxmann et al. 2008b). Aus Anwendersperspektive hat diese Art des Preismodells den Vorteil, dass Software auch

für kurze Nutzungsintervalle wirtschaftlich eingesetzt werden kann, da in der Regel die Höhe monatlicher Zahlungen deutlich unter der einer Einmalzahlung für Lizenzen liegt (Cusumano 2007, S. 20). Dieser Vorteil aus Kundensicht stellt allerdings höhere finanzielle Anforderungen an den Anbieter. So haben SaaS-Anbieter nicht selten Schwierigkeiten, das Unternehmen in die Gewinnzone zu bringen (Hill 2008, S. 48). Nach einer Umfrage der SIIA (2006, S. 5) erwarten amerikanische Softwareanbieter in Zukunft eine stärkere Verbreitung des Subskriptionsmodells gegenüber der Einmalzahlung in Form eines Lizenzkaufs (siehe hierzu auch Abschn. 3.3.3).

Darüber hinaus sind hybride Formen aus einmaligen und regelmäßigen Zahlungen denkbar. Verbreitet ist hier beispielsweise der Erwerb einer Softwarelizenz, der mit dem Abschluss eines Softwarewartungsvertrags verbunden ist. Dieser sieht in der Regel jährliche Zahlungen in Höhe eines bestimmten Prozentsatzes der (einmaligen) Lizenzzahlung vor. In aktuellen Preismodellen vieler Softwareanbieter liegt der Wartungsprozentsatz bei ca. 20%. Dieses Modell hat für Anbieter den Vorteil, dass es zu relativ gleichmäßigen Zahlungsströmen führt. In Abschn. 3.3.3 werden wir empirische Ergebnisse zu den Zahlungsformen von SaaS-Lösungen wiedergeben.

3.3.2.4 Bemessungsgrundlage

Eine weitere Gestaltungsmöglichkeit betrifft die Festlegung der Bemessungsgrundlage des Preismodells. Das bedeutet, dass der Preis beispielsweise je Nutzer oder in Abhängigkeit von der in Anspruch genommenen Zeit bestimmt wird. Die Bemessungsgrundlage ist ein wesentlicher Parameter, der darüber entscheidet, ob Kunden das Preismodell des Anbieters als fair empfinden.

Zunächst wird festgelegt, aus wie vielen Preiskomponenten (Skiera 1999b) das Preismodell besteht. Jede Preiskomponente basiert auf einer Bemessungsgrundlage. Beispielsweise kann das Preismodell in einen von der Nutzung unabhängigen Grundbetrag, der monatlich in einer festgelegten Höhe anfällt, und in eine nutzungsabhängige Komponente, z. B. die genutzte Speicherkapazität, aufgeteilt werden. Skiera (1999b) konnte zeigen, dass Dienstleistungsunternehmen durch die Verwendung von zwei Preiskomponenten gegenüber einem Preismodell mit einer Preiskomponente erhebliche Gewinnsteigerungen erzielen können.

Wie bereits im Beispiel angedeutet, kann die Bemessungsgrundlage entweder in Abhängigkeit der Nutzung der Software oder nutzungsunabhängig sein. Prinzipiell ist hier eine Vielzahl von Bemessungsgrößen denkbar. Diese können auch branchenspezifisch sein, wie beispielsweise für eine Software zur Verwaltung von Wohnungen die Anzahl verwalteter Mietobjekte als Bemessungsgrundlage dienen kann.

Beispiele für nutzungsabhängige Bemessungsgrundlagen zeigt Tab. 3.4.

Die Verwendung nutzungsabhängiger Bemessungsgrundlagen kann zu fixen und variablen Administrationskosten führen, z. B. im Bereich des Monitorings der Nutzung und der Abrechnung.

Nutzungsunabhängige Bemessungsgrundlagen sind Größen, die sich nicht an der tatsächlichen Nutzung der Software orientieren. Beispiele können Tab. 3.5 entnommen werden.

Tab. 3.4 Beispiele für nutzungsabhängige Bemessungsgrundlagen. (Lehmann und Buxmann 2009)

Bemessungsgrundlage	Beschreibung
Transaktion	Der Preis richtet sich nach der Anzahl durchgeföhrter Transaktionen der Software. Dies kann sowohl auf technischer (z. B. Webservice-Aufrufe) als auch auf inhaltlicher Ebene (z. B. Anzahl bearbeitete Lieferpositionen) sein
Speicherbedarf	Der Preis wird in Einheiten des Speicherbedarfs des Kunden gemessen (z. B. je GB)
Zeit	Die Höhe des Preises bestimmt sich nach der tatsächlichen Dauer der Softwarenutzung (z. B. minutengenaue Abrechnung)

Tab. 3.5 Beispiele für nutzungsunabhängige Bemessungsgrundlagen. (Lehmann und Buxmann 2009)

Bemessungsgrundlage	Beschreibung
Named user	Die Nutzungsrechte an der Software sind an eine spezifische Person gebunden und somit bezieht sich der Preis auf eine bestimmte Person
Concurrent user	Diese Größe erlaubt eine gleichzeitige Nutzung der Software durch eine im Vorfeld festgelegte Anzahl von Usern
Server/Rechner	Abgerechnet wird je Server oder Rechner. Die Nutzungsrechte sind an einen Server oder Rechner gebunden
CPU	Die Software wird nach der Anzahl der CPUs berechnet, auf denen sie läuft
Stammdaten	Die Bepreisung der Software richtet sich nach der Anzahl der eingepflegten Stammdaten (z. B. Kunden, Lieferanten, Mitarbeiter, Ersatzteilbestand, Mieteinheiten, Flurstück, verwaltetes Vermögen)
Standorte	Der Preis gilt hier je Standort/Produktionsstätte. Dies können beispielsweise auch spezielle Formen des Standorts sein (z. B. Minen)
Produzierte Menge	Die Software wird bepreist in Abhängigkeit von der produzierten Menge des anwendenden Unternehmens (z. B. pro Tag geförderter Barrel Erdöl)
Finanzkennzahlen	Der Preis wird bestimmt aus betriebswirtschaftlichen Kennzahlen (z. B. Umsatz, Ausgaben, Budget)

Für eine nutzungsunabhängige Bemessungsgrundlage spricht aus Anbietersicht, dass Kunden gewöhnlich bereit sind, für die Option der unbegrenzten Nutzung mehr zu zahlen (Sundararajan 2004, S. 1661). Dabei überschätzen viele Kunden ihr Nutzungsverhalten (Flatrate-Bias) (Lambrecht und Skiera 2006, S. 221). Empirische Ergebnisse zur Verbreitung von nutzungsabhängigen bzw. nutzungsunabhängigen Preismodellen im SaaS-Umfeld sind in Abschn. 3.3.3 dargestellt.

Neben der grundsätzlichen Bedeutung der Bemessungsgrundlage für den Kunden wird die Bemessungsgrundlage für die Gestaltung von Preisdifferenzierung benötigt. Hierauf gehen wir im Folgenden ein.

3.3.2.5 Preisdifferenzierungsstrategien

Unter Preisdifferenzierung wird das Anbieten prinzipiell gleicher Produkte an Nachfrager zu unterschiedlichen Preisen verstanden (z. B. Diller 2008, S. 227; Skiera und Spann 2000; Pepels 1998, S. 89). Ziel des Anbieters ist eine verbesserte Abschöpfung der Konsumentenrente. Dies kann gegenüber einem Preismodell mit Einheitspreis durch die Berücksichtigung der unterschiedlichen Zahlungsbereitschaften der Kunden erreicht werden. Bei unterschiedlichen Produktnutzeneinschätzungen der Anwender kann der Anbieter durch differenzierte Preise insgesamt höhere Umsätze erzielen (Diller 2008, S. 227). Ein Beispiel ist eine Musik-CD, die in drei Versionen zu verschiedenen Preisen angeboten wird:

- Die Premium-Version enthält ein umfangreiches Angebot von zusätzlichen Features, z. B. Liederbücher, Multimediacards mit Zugang zu einem exklusiven Internetangebot oder Bonustracks.
- Bei der Standard-Version handelt es sich um eine herkömmliche Albumausstattung.
- Demgegenüber umfasst die Basic-Version lediglich eine einfache Verpackung ohne Booklet.

Schon seit 1929 unterscheidet Pigou drei Formen der Preisdifferenzierung: Preisdifferenzierung ersten, zweiten und dritten Grades.

Die zumindest auf den ersten Blick optimale Strategie für Softwareanbieter besteht grundsätzlich in einer Preisdifferenzierung ersten Grades. Die Idee ist, eine individuelle Preisdifferenzierung durchzuführen, bei der das Produkt den Kunden genau zu deren Reservationspreis, d. h. der maximalen Zahlungsbereitschaft, angeboten wird. Preisuntergrenzen sind bei digitalen Gütern aufgrund der vernachlässigbaren variablen Kosten nicht zu beachten. Eine solche Preisstrategie ist für einen Softwareanbieter, der überwiegend im Business-to-Consumer-Bereich mit Millionen Kunden tätig ist, natürlich nicht praktikabel, da es nicht möglich sein wird, die Zahlungsbereitschaften der einzelnen Kunden auch nur annähernd zu erfassen und abzubilden. Demgegenüber ist bei Standardsoftwareanbietern im Business-to-Business-Geschäft bisweilen eine Tendenz in Richtung einer perfekten Preisdifferenzierung zu beobachten. So sind die Preise in diesem Segment häufig komplex und wenig transparent. Die Preislisten sind oftmals mehrere hundert Seiten dick und es gibt enorme Verhandlungsspielräume bei der Vertragsgestaltung. Dieser Verhandlungsspielraum bezieht sich nicht nur auf die Softwarelizenzen, sondern spielt gerade auch für die komplementären Beratungsdienstleistungen eine Rolle. Jedoch ist auch zu berücksichtigen, dass es sich ein Standardsoftwareanbieter in der Regel nicht leisten kann, bei den Preisunterschieden zu übertreiben. So bestünde beispielsweise die Gefahr, dass die Kunden sich im Rahmen von Anwendervereinigungen austauschen und auf diese Weise von der unterschiedlichen Preissetzung erfahren.

Die Preisdifferenzierung zweiten Grades spielt bei digitalen Gütern eine wichtige Rolle (Linde 2008, S. 209). Sie basiert auf dem Prinzip der Selbstselektion, d. h., der Kunde entscheidet selbst, welche Produkt-Preis-Kombination er auswählt (Varian 1997, S. 193).

Skiera (1999a, S. 287) unterscheidet bei der Preisdifferenzierung mit Selbstselektion die mengen-, zeit-, und leistungsbezogene Preisdifferenzierung¹.

Bei der mengenbezogenen Preisdifferenzierung ändert sich der durchschnittliche Preis pro Einheit mit der insgesamt eingekauften Menge. Hierzu zählt auch die Flatrate, da der Durchschnittspreis pro Einheit von der Gesamtnutzung des Kunden abhängt (Skiera und Spann 2000). Diese Form der Mengenrabatte ist bei Softwarelizenzen, insbesondere für Großkunden, verbreitet. Dabei sind Rabatte von bis zu über 50% keine Seltenheit.

Die zeitbezogene Preisdifferenzierung zielt auf die unterschiedlich hohe Zahlungsbereitschaft der Kunden zu verschiedenen Zeitpunkten ab (Skiera und Spann 1998). Ein Beispiel für eine zeitbezogene Preisdifferenzierung sind Börsenkurse, die kostenfrei mit einer zeitlichen Verzögerung bereitgestellt werden; Echtzeitkurse sind demgegenüber kostenpflichtig. Es gibt viele andere Beispiele, etwa wenn wir an eine saisonale Preisdifferenzierung denken. Eine für die Softwareindustrie relevante Variante ist die Bepreisung des Kundendienstes in Abhängigkeit von der Tageszeit des Einsatzes.

Eine weitere Form der Preisdifferenzierung mit Selbstselektion ist die leistungsbezogene Preisdifferenzierung. Sie liegt vor, wenn relativ geringfügige Änderungen im Leistungsumfang bzw. in der Leistungsqualität vorgenommen werden (Diller 2008, S. 237). Diese Produktvarianten werden zu unterschiedlichen Preisen angeboten. Im Zusammenhang mit einer solchen Produktdifferenzierung wird häufig auch von Versioning gesprochen (Varian 1997; Viswanathan und Anandalingam 2005).

Das Anbieten von unterschiedlichen Versionen eines Produkts wird insbesondere für digitale Güter aufgrund der Kostenstruktur als sehr vorteilhaft eingeschätzt (Viswanathan und Anandalingam 2005, S. 269). Auch vor dem Hintergrund der Netzeffekte können preiswerte Varianten zu einer größeren Marktdurchdringung führen. Dies zeigt, dass Softwareprodukte für diese Form der Preisdifferenzierung generell gute Voraussetzungen aufweisen (Bhargava und Choudhary 2008, S. 1029). Softwarehersteller entwickeln häufig zunächst ein qualitativ hochwertiges und umfangreiches Produkt, um anschließend bestimmte Funktionalitäten herauszunehmen und dem Kunden somit verschiedene Versionen anbieten zu können (Shapiro und Varian 1999, S. 63). Ein Beispiel für leistungsbezogene Preisdifferenzierung sind die verschiedenen Versionen von Microsoft Windows: „Windows 8.1“, „Windows 8.1 Pro“, „Windows 8.1 Enterprise“ und „Windows RT 8.1“, die sich hinsichtlich Funktionsumfang und Preis unterscheiden.

Bei der Anzahl der Versionen ist zu berücksichtigen, dass eine zu große Anzahl einerseits auf Kunden verwirrend wirken kann und andererseits den Aufwand auf Seiten des Anbieters erhöht (Viswanathan und Anandalingam 2005, S. 269). Aufgrund von Extreme-ness Aversion, d. h. der Abneigung gegen Extremes, wird als Daumenregel empfohlen, bei Informationsgütern drei Versionen anzubieten, so dass sich der Kunde als Kompromisslösung für die mittlere Variante entscheiden kann (Varian 1997, S. 200; Simonson und Tversky 1992; Smith und Nagle 1995). Die Grundidee lautet, dass unschlüssige Konsumenten tendenziell eher das Produkt mit einer „mittleren“ Qualität bevorzugen. Das folgende Bei-

¹ Aufgrund ihrer untergeordneten Bedeutung für die Softwareindustrie wurde von der suchkostenbezogenen Preisdifferenzierung abgesehen.

spiel illustriert das Prinzip: Würde eine Schnellimbisskette lediglich die Getränkegrößen „groß“ und „klein“ anbieten, tendierten einige unschlüssige Konsumenten sicherlich dazu, die „kleine“ Variante zu wählen. Wenn wir jedoch annehmen, dass der Anbieter eine zusätzliche „Jumbo-Variante“ anbietet und die neue Variante „medium“ identisch mit der vorherigen Variante „groß“ ist, besteht eine hohe Wahrscheinlichkeit, dass viele Kunden sich für die neue Medium-Variante entscheiden werden (Varian 1997, S. 199 f.).

Bhargava und Choudhary (2008) empfehlen, dass Unternehmen bei sinkenden variablen Kosten weitere Versionen mit geringerer Qualität in Betracht ziehen sollten. Die Autoren zeigen formal, dass mit abnehmenden variablen Kosten Versioning für Anbieter vorteilhafter wird. Dies wird mit dem Zugewinn an Kunden mit geringerer Zahlungsbereitschaft begründet (Bhargava und Choudhary 2008, S. 1031).

Die Preisdifferenzierung dritten Grades basiert auf der durch den Anbieter vorgenommenen Marktsegmentierung (z. B. Diller 2008, S. 229). Im Gegensatz zur Preisdifferenzierung zweiten Grades kann hier der Kunde nicht selbst wählen. Die Preisdifferenzierung dritten Grades lässt sich in personen- und regionenbezogen unterteilen (Skiera und Spann 2000).

Eine gruppenorientierte Differenzierung nach Personen wird häufig eingesetzt, um Lock-in-Effekte zu erzeugen: „Although software producers don't hang around outside of schoolyards pushing their products (yet), the motivation is much the same.“ (Shapiro und Varian 1998, S. 46). Beispielsweise könnte ein Softwareanbieter seine Produkte an bestimmte Konsumentengruppen verschenken, um Lock-in-Effekte aufzubauen. Die Idee ist, dass die im Zitat beschriebenen Schüler beispielsweise lernen, mit der Menüführung dieser Software umzugehen. Werden sie im Laufe der Zeit zu zahlenden Kunden, besteht die Hoffnung, dass sie sich auch dann für diese Software entscheiden.

Eine einfache Variante der räumlichen Preisdifferenzierung besteht darin, Produkte in unterschiedlichen Regionen zu verschiedenen Preisen zu verkaufen, was zum Teil sowohl beim Verkauf von Softwarelizenzen als auch bei der Abrechnung von Service- und Beratungsverträgen stattfindet.

Bei einer Preisdifferenzierung nach mehr als nur einer Dimension wird von mehrdimensionaler Preisdifferenzierung gesprochen (Skiera und Spann 2002, S. 279). Dies findet in der Praxis häufig Anwendung. Beispielsweise ist eine gleichzeitige regionen- und mengenbezogene Preisdifferenzierung denkbar. Dabei hat der Anbieter unterschiedliche Preise je Land oder Region und zusätzlich ein von der Absatzmenge abhängiges Preismodell. Ziel der mehrdimensionalen Preisdifferenzierung ist eine feinere Segmentierung der Kunden. Hierdurch können die vorhandenen Zahlungsbereitschaften noch besser abgeschöpft werden. Zu beachten ist jedoch, dass die Komplexität des Preismodells für den Käufer noch erfassbar ist und die Abrechnung durch den Verkäufer noch gewährleistet werden kann (Skiera und Spann 2002, S. 279).

3.3.2.6 Preisbündelung

Die Preisbündelung ist ein weiterer Parameter der Preisgestaltung für Softwareprodukte. Unter Preisbündelung wird allgemein die „Zusammenstellung mehrerer identifizierbarer Teilleistungen (Produkte, Dienste und/oder Rechte) eines oder mehrerer Anbieter zu ei-

nem Angebotspaket („Set“) mit Ausweis eines Gesamtpreises“ (Diller 2008, S. 240) verstanden. Die Preisbündelung kann auch als Spezialfall der Preisdifferenzierung gesehen werden (Skiera et al. 2005, S. 290; Diller 2008, S. 240). Aufgrund ihrer hohen Bedeutung in der Softwareindustrie betrachten wir das Thema Preisbündelung im Rahmen der Preisgestaltung für Softwareprodukte separat.

Die mit der Preisbündelung verbundenen Ziele sind vielfältig. In erster Linie wird Bündelung zur Preisdifferenzierung (siehe Abschn. 3.3.2.5) eingesetzt (Viswanathan und Anandalingam 2005, S. 264). Während die konventionellen Methoden der Preisdifferenzierung vergleichsweise detaillierte Kenntnisse der Reservationspreise der einzelnen Produkte erfordern, ist dies bei der Preisbündelung in geringerem Umfang notwendig (Adams und Yellen 1976, S. 476). Bakos und Brynjolfsson (1999) erklären dies mit dem Gesetz der großen Zahlen. Hiernach ist es für den Anbieter einfacher, die Zahlungsbereitschaft für ein Bündel mit einer Vielzahl an Produkten zu prognostizieren als für jedes Produkt einzeln, da die Verteilung der Zahlungsbereitschaften für das Bündel eine geringere Anzahl an Extremwerten hat (Viswanathan und Anandalingam 2005, S. 264). Wu et al. (2008, S. 608–609) wenden jedoch ein, dass dies nur bei variablen Kosten von Null gilt, sobald selbst sehr geringe variable Kosten vorliegen, entstehen nicht unerhebliche Kosten für ein Bündel mit sehr vielen Elementen, welche potenzielle Vorteile der Bündelung relativieren.

Insbesondere in der durch Netzeffekte geprägten Softwareindustrie kann die Bündelung für Anbieter vorteilhaft sein, da sie eine stärkere Verbreitung von (zusätzlichen) Produkten im Markt fördert. Beispielsweise enthält die Creative Suite von Adobe neben Software zur Bildbearbeitung und Layoutgestaltung auch Software zur Erstellung von PDF-Dateien. Daher fördert der Absatz der Creative Suite auch die Verbreitung von PDF-Dokumenten. Weiterhin kann die Bündelungsstrategie den Markteintritt potenzieller Konkurrenten erschweren (Nalebuff 2004), z. B. für Anbieter, die nur eines der Produkte aus dem Bündel produzieren. Ferner kann die Bündelung dem Ziel der Kosteneinsparung bei der Fakturierung und Lieferung dienen, da in einem Vorgang mehrere Produkte gleichzeitig verkauft werden (Viswanathan und Anandalingam 2005, S. 264; Adams und Yellen 1976, S. 475–476).

Im Weiteren wollen wir auf die in Abb. 3.29 dargestellten Aspekte der Preisbündelung eingehen, bevor wir im Anschluss die Einflussfaktoren auf die Vorteilhaftigkeit von Bündelungsstrategien diskutieren.

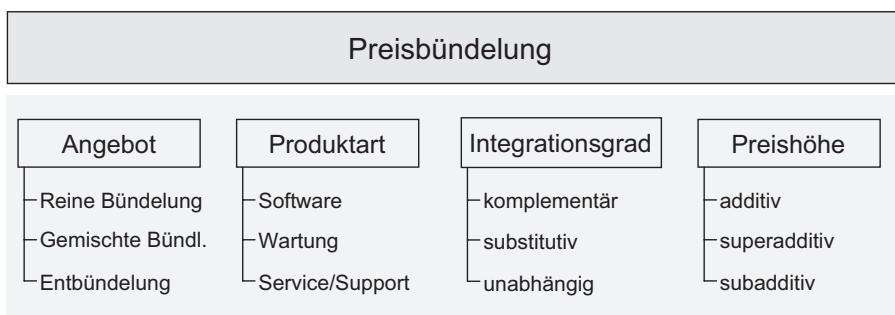


Abb. 3.29 Aspekte der Preisbündelung. (Lehmann und Buxmann 2009)

In Abhängigkeit von der Art des Angebots des Softwareanbieters werden die drei Formen reine und gemischte Bündelung (englisch: Pure Bundling, Mixed Bundling) sowie Entbündelung unterschieden. Bei der reinen Bündelung werden die Produkte ausschließlich im Bündel angeboten. Kann der Kunde wählen, ob er das gesamte Bündel oder die jeweiligen Produkte einzeln kauft, so handelt es sich um gemischte Bündelung. Entbündelung liegt vor, wenn der Kunde die Produkte nur separat erwerben kann (Adams und Yellen 1976; Schmalensee 1984, S. 212, S. 475; Olderog und Skiera 2000, S. 140). Eine weitere Variante ist das Customized Bundling, bei dem der Kunde innerhalb bestimmter Vorgaben selbst wählt, welche Produkte in das Bündel aufgenommen werden. Der Anbieter legt lediglich Preis und Umfang des Bündels fest (Hitt und Chen 2005). Wu und Anandalingam (2002) zeigen, dass das Angebot mehrerer Customized Bundles für einen monopolistischen Anbieter von Informationsgütern vorteilhaft sein kann. Unter der Voraussetzung unvollständiger Informationen ermitteln Wu et al. (2008), dass Customized Bundling für den Anbieter erfolgversprechender ist als reine Bündelung oder Entbündelung.

Ein weiterer Aspekt der Preisbündelung ist die Produktart. Die Teilleistungen des Bündels können ganz unterschiedlicher Art sein. In der Softwareindustrie kommen für die Produktart im Wesentlichen die Software an sich, ihre Wartung und Service- bzw. Supportleistungen in Betracht. Nicht selten teilt sich heutzutage der Umsatz eines Softwareherstellers in drei gleiche Teile Lizenzen, Wartung und Services (Cusumano 2007, S. 19). Die Leistungen aus diesen drei Bereichen können in verschiedenen Formen als Bündel angeboten werden.

Weiterhin lassen sich die Produkte im Bündel auch hinsichtlich ihres Integrationsgrades beschreiben. So können die Teilleistungen komplementär zueinander sein (Diller 2008, S. 241) bzw. in einem substitutiven oder unabhängigen Verhältnis zueinander stehen. Bakos und Brynjolfsson (1999) haben herausgefunden, dass das Bündeln einer größeren Anzahl unabhängiger Informationsgüter profitabel sein kann. Ihr Modell ermöglicht auch die Analyse komplementärer und substitutiver Bündelelemente. Stremersch und Tellis (2002) schlagen vor, im Falle von Produkten, die integriert in ein Bündel eingehen, so dass es zu einem Mehrwert für die Kunden im Vergleich zum Konsum oder zur Anwendung der Einzelprodukte kommt, von einer Produktbündelung (im Gegensatz zu Preisbündelung) zu sprechen.

Die Höhe des Preises für das Bündel kann additiv, superadditiv oder subadditiv festgelegt werden. Bei einem additiven Bündel entspricht der Bündelpreis der Summe der Einzelpreise. Liegt ein superadditives Bündel vor, so liegt der Preis des Bündels über dem der Summe der Teilleistungen, während ein subadditives Bündel einen Preis unterhalb dieser Summe aufweist (Diller 2008, S. 240–241). Die letzte Variante, d. h. ein Bündel mit Preisnachlässen gegenüber den Einzelpreisen, wird als Normalfall bezeichnet (Diller 2008, S. 241, Viswanathan und Anandalingam 2005, S. 264). Günther et al. (2007, S. 139) haben in einer Studie herausgefunden, dass die Mehrheit der Befragten beim Kauf zusammengesetzter Web Services einen niedrigeren Gesamtpreis für das Bündel erwartet. Beispielsweise liegt der Preis für das Bündel Microsoft Office deutlich unterhalb der Summe der Einzelpreise seiner Bestandteile. Als Microsoft das Betriebssystem Windows als Bündel

mit dem Media-Player auslieferte, hatten die Kunden den Eindruck, dass dieses Produkt kostenlos dazugegeben wurde. Diese Form der Bündelungsstrategie kann beispielsweise auch dazu genutzt werden, älteren Produkten neue Applikationen beizulegen, um den Kunden zum Upgrade oder zu Wartungsangeboten zu bewegen (Cusumano 2007, S. 20). Allerdings ist zu berücksichtigen, dass einer Bündelung von Produkten Wettbewerbsgesetze entgegenstehen können. Beispielhaft ist hierfür das Vorgehen der Europäischen Kommission gegen Microsoft aufgrund der Bündelung seines Betriebssystems Windows mit dem Internet Explorer.

Einflussfaktoren auf die Vorteilhaftigkeit von Bündelung wurden u. a. von Olderog und Skiera (2000) auf Grundlage der Untersuchungen von Schmalensee (1984) untersucht. Der Erfolg der Anwendung von Bündelungsstrategien ist im Wesentlichen von zwei Faktoren abhängig: Zum einen davon, in welcher Weise die Reservationspreise korreliert sind, zum anderen von der Höhe der variablen Kosten im Verhältnis zu den Reservationspreisen. Auf diese beiden Faktoren wird im Folgenden eingegangen. Positiv korrelierte Reservationspreise liegen vor, wenn die Konsumenten, die für ein Produkt A viel (wenig) zu zahlen bereit wären, auch viel (wenig) für Produkt B zahlen würden. Eine negative Korrelation liegt vor, wenn die Kunden, die tendenziell viel (wenig) für Produkt A bezahlen würden, eine geringe (hohe) Zahlungsbereitschaft für Produkt B haben. Grundsätzlich gilt, dass eine Bündelungsstrategie für den Anbieter umso vorteilhafter ist, je negativer die Korrelation der Reservationspreise ist. Der Grund ist, dass eine solche Korrelation zu einer Reduktion der Varianz für die Reservationspreise des Bündels und damit zu einer homogeneren Struktur der Zahlungsbereitschaften für das Bündel führt (Olderog und Skiera 2000, S. 142). Dieser Zusammenhang soll im Weiteren grafisch erklärt werden. In Abb. 3.30 sind die Verteilungen der Reservationspreise für zwei Produkte dargestellt.

Abbildung 3.31 verdeutlicht, wie sich die Korrelation der Zahlungsbereitschaften für die Einzelprodukte auf die Homogenität der Reservationspreise für das Bündel auswirkt.

Die Homogenität der Zahlungsbereitschaften für das Bündel nimmt zu (ab), wenn die Korrelation der Reservationspreise negativ (positiv) ist. Abbildung 3.32 zeigt, dass der

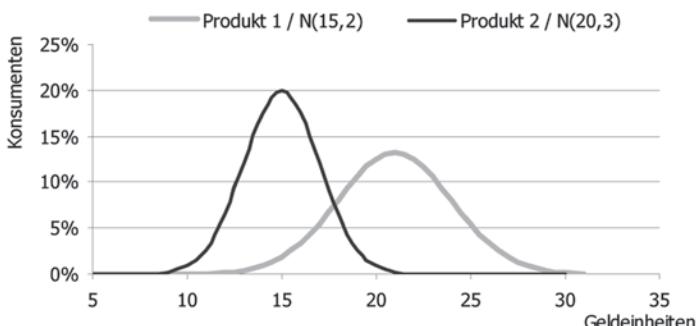


Abb. 3.30 Beispielhafte Verteilungen von Reservationspreisen für zwei Produkte. (Olderog und Skiera 2000, S. 143)

Abb. 3.31 Auswirkung der Korrelationen der Reservationspreise auf die Homogenität der Reservationspreise für das Bündel. (Olderog und Skiera 2000, S. 143)

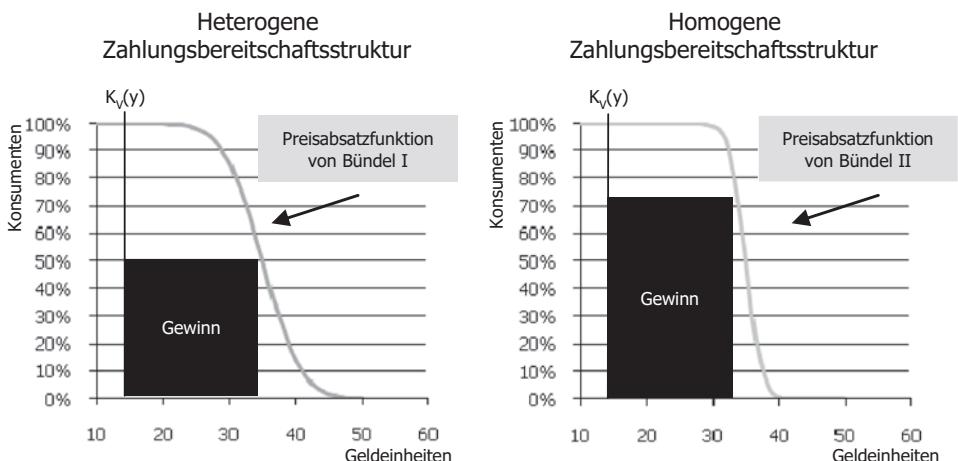
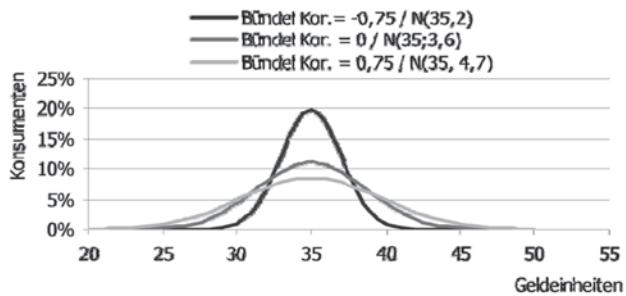


Abb. 3.32 Auswirkungen einer homogenen Zahlungsbereitschaftsstruktur auf die möglichen Gewinne des Anbieters. (Olderog und Skiera 2000, S. 144)

Anbieter seinen Gewinn (durch die dunkle Fläche gekennzeichnet) bei Vorliegen einer homogenen Zahlungsbereitschaftsstruktur erhöhen kann.

Wie bereits erläutert, ist der Erfolg einer Preisbündelung zudem von der Höhe der variablen Kosten im Verhältnis zu den Reservationspreisen der Konsumenten abhängig (Olderog und Skiera 2000, S. 144; Bakos und Brynjolfsson 1999). Dies soll nun anhand eines einfachen Zahlenbeispiels erklärt werden, das in der nachfolgenden Tab. 3.6 dargestellt ist.

Wir gehen von einem Zwei-Produkt-Fall aus und vergleichen, ob ein Anbieter in diesem Fall bei relativ hohen variablen Kosten mit einer Entbündelungs- oder Bündelungsstrategie erfolgreicher ist. In den ersten beiden Zeilen sind die maximalen Zahlungsbereitschaften von zwei Kunden für die Produkte 1 und 2 sowie für das Bündel, das aus genau diesen beiden Produkten besteht, angegeben. Wir sehen auch, dass die Zahlungsbereitschaften beider potenzieller Kunden negativ korrelieren. In der dritten Zeile ist der optimale Preis für die beiden Produkte sowie für das Bündel dargestellt. Da die variablen Kosten für die Herstellung jedes der beiden Produkte annahmegemäß sieben Geldeinheiten betragen, ist es sinnvoll, für beide Produkte einen Preis von acht Geldeinheiten zu

Tab. 3.6 Beispiel für die Auswirkungen einer Bündelungsstrategie bei relativ hohen variablen Kosten

	Entbündelung		Bündelung
	Produkt 1	Produkt 2	Bündel
ZB Käufer i=1	8	4	12
ZB Käufer i=2	4	8	12
Variable Kosten	7	7	14
Optimaler Preis	8	8	/
Deckungsbeitrag	1	1	/
Abgesetzte Menge	1	1	0
Gewinn	1	1	0
Ergebnis	2 >		0

ZB Zahlungsbereitschaft

Tab. 3.7 Beispiel für die Auswirkungen einer Bündelungsstrategie bei digitalen Gütern

	Entbündelung		Bündelung
	Produkt 1	Produkt 2	Bündel
ZB Käufer i = 1	8	4	12
ZB Käufer i = 2	4	8	12
Variable Kosten	0	0	0
Optimaler Preis	8	8	12
Deckungsbeitrag	8	8	12
Abgesetzte Menge	1	1	2
Gewinn	8	8	24
Ergebnis	16 <		24

setzen. Dies führt zu einem positiven Deckungsbeitrag von jeweils einer Geldeinheit für die Produkte 1 und 2. Sieht man von möglichen Fixkosten ab, so beträgt der Gewinn bei einer Entbündelungsstrategie für den Anbieter zwei Geldeinheiten. Demgegenüber ist es für den Anbieter offensichtlich nicht sinnvoll, eine Bündelungsstrategie zu verfolgen – die variablen Kosten sind zu hoch.

Wie aber stellt sich diese Entscheidung bei geringeren variablen Kosten dar? Um dies zu zeigen, rechnen wir unser einfaches Beispiel nun mit variablen Kosten in Höhe von Null durch (siehe Tab. 3.7).

Wir sehen, dass es hier mit einer Bündelungsstrategie aufgrund der homogenen Struktur der Zahlungsbereitschaften möglich ist, die Reservationspreise der Konsumenten voll abzuschöpfen. Die Bündelungsstrategie ist aufgrund der niedrigeren variablen Kosten also vorteilhafter als die Entbündelungsstrategie.

Den Einfluss der Höhe der variablen Kosten auf den Erfolg von Bündelungsstrategien zeigt abschließend Abb. 3.33.

Dabei haben wir für unser Zahlenbeispiel die Gewinne der Bündelungs- oder Entbündelungsalternative in Abhängigkeit von der Höhe der variablen Kosten eingezeichnet. Es

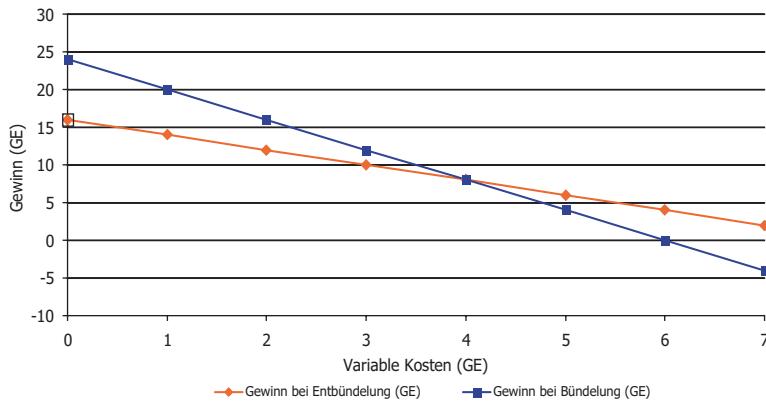


Abb. 3.33 „Break even Point“ in Abhängigkeit der variablen Kosten

zeigt sich, dass variable Kosten in Höhe von vier Geldeinheiten den kritischen Wert darstellen. Übersteigen die variablen Kosten diesen Wert, so ist Entbündelung für den Anbieter die bessere Alternative. Umgekehrt profitiert er bei variablen Kosten, die unterhalb von vier Geldeinheiten liegen, stärker von der Bündelung.

Ein weiterer Vorteil der Bündelung besteht für Softwareanbieter darin, dass sie auf diese Weise zu einer Verbreiterung der installierten Basis ihrer Produkte beitragen, damit also Netzeffekte erzeugen können.

Zusammenfassend lässt sich festhalten, dass bei Vorliegen einer negativen Korrelation der Reservationspreise für die verschiedenen Produkte eine Bündelung tendenziell zu einer homogeneren Nachfragestruktur führt. Außerdem sind Bündelungsstrategien grundsätzlich umso vorteilhafter, je geringer die variablen Kosten sind. Daher ist die Preisbündelung für Softwareprodukte von besonderem Interesse. Zu beachten ist schließlich, dass die optimale Anzahl an Produkten in einem Bündel auch von vorliegenden Budgetbeschränkungen auf Kundenseite bestimmt sein kann (Bakos und Brynjolfsson 1999).

3.3.2.7 Dynamische Preisstrategien

Im Gegensatz zu den bislang untersuchten Ansätzen beruhen dynamische Preisstrategien auf einer mehrperiodigen Betrachtung, die typischerweise durch Preisänderungen bzw. -anpassungen im Zeitablauf gekennzeichnet ist. In den folgenden Abschnitten werden die Penetrationsstrategie, die Follow-the-Free- und die Skimmingstrategie dargestellt. Auf eine Diskussion der Pulsationsstrategie (Anbieter erhöhen und senken die Preise abwechselnd im Zeitablauf) verzichten wir, da wir für die Softwareindustrie im Regelfall keine sinnvollen Anwendungsmöglichkeiten sehen.

Das Ziel der Anwendung einer Penetrationsstrategie besteht darin, durch Niedrigpreise schnell Marktanteile zu gewinnen. Eine solche Niedrigpreisstrategie kann insbesondere auf Netzeffektmarkten von Bedeutung sein, wenn wir etwa an die bereits angesprochene Start-up-Problematik in Verbindung mit Lock-in-Effekten denken.

Die Softwarebranche ist aufgrund von Netzeffekten für die Anwendung einer Penetrationsstrategie insofern gut geeignet. Zudem begünstigt die Kostenstruktur mit den ver-

nachlässigbaren variablen Kosten die Anwendung einer solchen Niedrigpreisstrategie: Selbst das Verschenken einer Software führt noch nicht zu negativen Deckungsbeiträgen. Es sind Situationen denkbar, in denen sogar eine „Draufzahl“-Strategie notwendig ist, um den Vorsprung des Konkurrenten einzuholen. Die Sinnhaftigkeit einer solchen Strategie hängt jedoch wesentlich von dem Netzeffektfaktor (d. h. der Höhe der Netzeffekte im Verhältnis zum Gesamtnutzen) ab. Dabei sind Niedrigpreisstrategien umso notwendiger, je höher der Netzeffektfaktor ist (Buxmann 2002), den wir bereits in Abschn. 2.2 eingeführt haben. Darüber hinaus kann eine Penetrationsstrategie insbesondere sinnvoll sein, wenn der entsprechende Anbieter ein zum Marktstandard inkompatibles Produkt anbietet.

Ein zweiter Schritt einer solchen Penetrationsstrategie kann darin bestehen, dass der Anwender nach Erreichen einer kritischen Masse die Preise wieder erhöht. Ahtiala (2006) hat in Experimenten gezeigt, dass es für Softwareanbieter vor dem Hintergrund der Softwarepiraterie vorteilhaft ist, Software anfangs sehr günstig zu verkaufen, um einen Lock-in-Effekt zu erzeugen, und erst später an Upgrades einen höheren Preis anzusetzen. In der Praxis kann diese Strategie aber auch insofern beobachtet werden, als einige Anbieter – und zwar auch große Player – in den letzten Jahren zum Teil Lizenzen verschenkt haben.

Die Follow-the-Free-Strategie umfasst zwei Schritte: Zuerst werden die Produkte verschenkt, um einen Lock-in-Effekt zu erzeugen, anschließend werden dann durch den Verkauf von Komplementärprodukten oder Premiumversionen an den bereits existierenden Kundenstamm Erlöse erzielt (Zerdick et al. 1999, S. 191–194). Denkbar ist z. B. das kostenlose Anbieten eines Softwareprodukts, während die dazugehörigen Services wie Installation, Wartung, Schulung und Customizing kostenpflichtig angeboten werden (Cusumano 2007, S. 21). Ein Beispiel aus der Softwareindustrie ist die Strategie des Softwareanbieters Adobe, dem es auf diese Weise zudem gelang, den PDF-Standard durchzusetzen.

Im Kontext von Softwareprodukten ist teilweise auch eine Skimming-Strategie zu beobachten, d. h., der Anbieter startet mit einem sehr hohen Preis, den er im Zeitverlauf senkt. Diese Strategie zielt im Kern darauf ab, die Heterogenität der Reservationspreise der Konsumenten dadurch auszunutzen, dass in der hochpreisigen Phase genau diejenigen Konsumenten das Produkt erwerben, die einen sehr hohen Reservationspreis haben. Anschließend soll im Verlauf der Preissenkung schrittweise die Zahlungsbereitschaft der übrigen Konsumenten abgeschöpft werden. Ein Beispiel sind Computerspiele, die nach der Markteinführung zuerst sehr teuer verkauft werden und nach einer gewissen Zeit bisweilen sogar kostenlos Zeitschriften beilegen.

3.3.3 Ansätze zur Preissetzung für Individualsoftwareanbieter

Die zuvor dargestellten Überlegungen gelten überwiegend für die Anbieter von Standardsoftware. Denken wir etwa an Methoden zur Produkt- und Preisdifferenzierung oder auch an die dynamische Bepreisung, so wird deutlich, dass diese Strategien für Individualsoftwareanbieter eher eine untergeordnete Rolle spielen. Im Folgenden werden die Möglich-

keiten, aber auch die Grenzen einer Preisfindung für Individualsoftwareanbieter dargestellt.

Die Grundlage einer solchen Analyse kann die Bestimmung bzw. Schätzung von Preisuntergrenzen sein. Hierzu können traditionelle Methoden zur Ermittlung des Aufwands von Softwareentwicklungsprojekten herangezogen werden, wie zum Beispiel COCOMO (II) bzw. die Function-Point-Methode (Balzert 2000).

Die Anwendung solcher Methoden ist auch insofern geboten, als keineswegs davon ausgegangen werden kann, dass sich Individualsoftwareprojekte präzise und verlässlich planen lassen. Projektkosten und -laufzeiten, die über den geplanten Werten liegen, sind eher die Regel als die Ausnahme.

Die gemeinsame Grundidee der Aufwandschätzmethoden besteht darin, dass bestimmte Parameterausprägungen vor Beginn eines Softwareentwicklungsprojektes Aussagen über den Projektaufwand ermöglichen sollen. Bei COCOMO wird in der einfachsten Variante die geschätzte Anzahl der Lines of Code herangezogen. Auf Basis von empirischen Vergangenheitsdaten wird mit Hilfe von Tabellen daraus der Aufwand des Projektes geschätzt. Demgegenüber sind bei Anwendung der Function-Point-Methode – bei der es sich um die am weitesten verbreitete Methode zur Aufwandschätzung von Softwareentwicklungsprojekten handelt – im ersten Schritt Parameter, wie z. B. externe Ein- und Ausgaben, Benutzertransaktionen, Schnittstellen zu externen Datenquellen oder die Anzahl der verwendeten Dateien, im Hinblick auf ihre Komplexität zu bewerten. Dazu können etwa Pflichtenhefte, Entity-Relationship-Diagramme, Use-Case-Diagramme, Bildschirm-Layouts und andere Dokumente herangezogen werden. Auf Basis der Bewertung der Kriterien werden über Tabellen die so genannten Function Points einer zu entwickelnden Lösung geschätzt. Es handelt sich bei diesem Wert um einen Indikator für den Umfang und die Komplexität einer zu erstellenden Softwarelösung.

Insbesondere große Individualsoftwareanbieter setzen auf eine Anwendung solcher Methoden, während kleine Softwarehäuser sich eher auf ihr „Bauchgefühl“ und ihr Erfahrungswissen verlassen. Ein Vorteil der Methoden ist, dass sie relativ einfach anzuwenden sind. Allerdings basieren auch sie letztlich auf Erfahrungswerten und empirischen Zusammenhängen und setzen zum Teil subjektiv zu ermittelnde Parameter voraus. Insbesondere für solche Projekte, bei denen es um Großaufträge geht, kann die Anwendung dieser Methoden jedoch in vielen Fällen wertvolle zusätzliche entscheidungsrelevante Informationen für die Anbieter bereit stellen. Da die Anwendung der Methoden im Vergleich zu den Projektkosten relativ kostengünstig ist, wird dieser Weg häufig sinnvoll sein. Als Ergebnis erhält man den geschätzten Projektaufwand, etwa gemessen in Personenmonaten, der in ein Projekt zu investieren ist, sowie die geschätzte Entwicklungszeit.

Bei dem Aufwand in Personenmonaten handelt es sich zweifelsohne um den bedeutendsten Kostenblock, den ein Softwareanbieter zu tragen hat. Zur Bestimmung einer Preisuntergrenze für ein solches Entwicklungsprojekt liegt es also nahe, zunächst aus den Personenmonaten die Personalkosten zu ermitteln. Zum Personal gehören in diesem Kontext sowohl die angestellten Mitarbeiter als auch Freelancer, die in der Regel zur Deckung von Bedarfsspitzen eingesetzt werden. Dabei umfassen die Personalkosten für festangestellte Mitarbeiter fixe sowie variable Gehaltsbestandteile, Arbeitgeberleistungen in die

Sozialversicherungen sowie Kosten für Schulungsmaßnahmen. Die Vergütung freier Mitarbeiter erfolgt entweder auf Basis von Stunden- bzw. Tagessätzen oder Festpreisen für bestimmte (Teil)-Projekte.

Wird der ermittelte Projektaufwand in Personenmonaten mit den entsprechenden Kosten für angestellte und freie Mitarbeiter gewichtet, so erhält der Anwender dieser Methode ein grobes Gefühl für die Projektkosten – nicht mehr, aber auch nicht weniger. Insbesondere in der industriellen Produktion werden auf Basis der Zuschlagskalkulation anteilige Gemeinkosten erfasst, um die Herstellkosten eines Produktes zu bestimmen. Dieses Prinzip kann in der Softwareindustrie auch angewendet werden, jedoch ist dieser Schritt in der Regel von nicht so großer Bedeutung, da diese anteiligen Gemeinkosten, beispielsweise für Räume, anteilige Nutzung von Softwareentwicklungsumgebungen etc., im Vergleich zu den Personalkosten keine sonderlich wichtige Rolle spielen.

Nachdem diese Kosten – mit oder ohne Zuschlagskalkulation – bestimmt worden sind, besteht die Aufgabe „nur“ noch darin, den Gewinnaufschlag festzulegen. Hierbei muss eine Vielzahl von Aspekten berücksichtigt werden. Dazu gehört etwa, dass die Function-Point-Methode tendenziell die tatsächlich anfallenden Kosten unterschätzt.

Das hier geschilderte Vorgehen entspricht dem einer kostenorientierten Preispolitik. Es liegt auf der Hand, dass diese Preise nicht zwingend in Einklang mit der Wettbewerbssituation sowie den Zahlungsbereitschaften der Kunden stehen müssen. Aus diesem Grund wird in der Literatur vorgeschlagen, dass es vorteilhafter sei, sich bei der Preisgestaltung an der Marktlage zu orientieren. Dies ist jedoch gerade im Projektgeschäft sehr schwierig, da Projekte per Definition einen einmaligen Charakter haben und ein Marktpreis daher kaum bestimmbar ist.

Bei der Festlegung des Angebotspreises sind darüber hinaus auch psychologische Aspekte zu beachten. Hier bietet es sich an, auf Erkenntnisse des Behavioral Pricing zurückzugreifen. In diesem noch relativ jungen Forschungsgebiet wird untersucht, wie potenzielle Kunden Preise bzw. Preisinformationen aufnehmen und verarbeiten, wie sie auf Preisangebote reagieren und wie sie Preisinformationen bei ihren Urteilen und Entscheidungen nutzen (Homburg und Koschate 2005). Dass Personen Preise häufig nicht absolut, sondern relativ zu einer Referenzgröße beurteilen, sich oft nicht an Preise erinnern können oder die Preissuche vorzeitig abbrechen, weicht von den Annahmen der klassischen Preistheorie ab. Beim Behavioral Pricing wird primär ein deskriptiver Forschungsansatz verfolgt, der sich insbesondere auf die kognitiven Prozesse, die von der klassischen Preistheorie nicht thematisiert werden, konzentriert.

Auch wenn eine kostenorientierte Preispolitik die oben dargestellten Nachteile besitzt, kann sie dennoch zumindest im Rahmen einer kurzfristigen Betrachtung eine gute Grundlage für die Beantwortung der Frage darstellen, ob ein bestimmtes Projekt wirtschaftlich durchführbar ist. Daneben ist es in der Regel sinnvoll, weitere Zielgrößen in die Entscheidung einzubeziehen, wie etwa Möglichkeiten der Gewinnung wichtiger Kunden, die Verteidigung und Steigerung von Marktanteilen oder die Verhinderung des Eintritts von Wettbewerbern.

Nachdem wir in den letzten beiden Abschnitten marktseitige Strategien untersucht haben, wollen wir uns im Folgenden auf Entwicklungsstrategien konzentrieren.

3.4 Entwicklungsstrategien

Schon lange beschäftigen sich Wissenschaft und Praxis mit der effizienten und effektiven Entwicklung von Software – früher überwiegend im Kontext anwendender Unternehmen, heute zunehmend auch aus der Perspektive von Softwareanbietern. Zahlreiche Konzepte wurden entwickelt, erprobt und wieder verworfen. Nachfolgend geben wir einen Überblick über die wichtigsten Konzepte mit Bestand.

3.4.1 Strukturierung der Softwareentwicklung

Von zentraler Bedeutung für die Softwareentwicklung ist die Frage nach der Strukturierung des Entwicklungsprozesses. Nach einer kurzen Darstellung der Anfänge der Softwareentwicklung werden zunächst die planbasierten Ansätze vorgestellt. Im Anschluss daran werden die als Kritik auf diese entstandenen agilen Entwicklungsansätze diskutiert. Abschließend vergleichen wir die relativen Vorteilhaftigkeiten der verschiedenen Ansätze in Abhängigkeit der jeweiligen Rahmenbedingungen.

Ad-hoc Entwicklung In die Zeit des Aufkommens der ersten Programmiersprachen in den fünfziger und Anfang der sechziger Jahre fällt auch die Entstehung einer der ersten Softwareentwicklungsmethodiken. Im Rahmen dieser – auch als „*Stagewise Model*“ bezeichneten – Methodik wird Software in aufeinander folgenden, streng sequenziellen Phasen entwickelt. Insgesamt glich die Entwicklung von Software in dieser Zeit aber eher einem Kunsthhandwerk („*Software Crafting*“), das von der individuellen Fähigkeit der einzelnen Entwickler abhing (Boehm 1986, S. 22; Dogs und Klimmer 2005, S. 15).

Die Softwareentwicklung lief nach dem so genannten „*Code and Fix*“-Ansatz ab: Hierbei handelt es sich um keine Methodik im eigentlichen Sinne. Vielmehr wurde ohne größere vorherige Planung Funktionalität entwickelt und bei auftauchenden Fehlern der Code solange geändert, bis das Programm schließlich ohne Fehlermeldung lief. Dieses Vorgehen führte zwangsläufig zu Programmcode, der unübersichtlich und damit schlecht zu warten war – es entstand der so genannte „*Spaghetti Code*“ (Boehm 2006, S. 13).

Planbasierter Ansatz Mit zunehmender Verbreitung von Computern stiegen auch die Anforderungen an die zugehörige Software. Da die Komplexität der Software allerdings immer weniger beherrschbar wurde, konnten Kosten-, Zeit- und Qualitätsziele häufig nicht eingehalten werden. In Folge dieser „*Software-Krise*“ wurde 1968 auf einer Konferenz der Begriff „*Software Engineering*“ geboren. Die zunehmende Orientierung der Softwareentwicklung an der strukturierten Vorgehensweise der Ingenieurswissenschaften sollte einen Beitrag dazu leisten, die Software-Krise zu überwinden (Dogs und Klimmer 2005, S. 16 f.). Im Jahr 1970 wurde schließlich erstmals das „*Wasserfallmodell*“ der Softwareentwicklung beschrieben. Dieses baute auf dem bereits erwähnten „*Stagewise Model*“ auf und galt fortan als Archetyp der so genannten planbasierten Entwicklungs-methodiken (Boehm 1986, S. 22).

Charakteristisch für planbasierte Methodiken ist, dass die Entwicklung der Software in standardisierten, hauptsächlich sequenziell ablaufenden Prozessschritten durchgeführt wird. Ausgehend von der Erhebung der Anforderungen und einer detaillierten Planung zu Beginn entstehen in den einzelnen Phasen umfangreiche Artefakte, wie eine Dokumentation der Nutzeranforderungen oder das technische Design. Diese bilden die Ausgangsbasis für die jeweils nachfolgenden Phasen. Nach der eigentlichen Programmierung wird die Software auf korrekte Umsetzung getestet.

In der Folgezeit kam es zur Kritik am Wasserfallmodell und den planbasierten Methoden im Allgemeinen. So können etwa Änderungen der Kundenanforderungen im Entwicklungsverlauf durch die sequenzielle Vorgehensweise nur schwer berücksichtigt werden. Diese führen zudem zu umfangreichen und aufwändigen Anpassungen der Artefakte. Durch Einbezug neuer Methoden, wie beispielsweise Rapid Prototyping, wurden neue Ansätze, wie etwa das Spiral-Modell (Boehm 1986, S. 21 ff.) und das V-Modell (Hindel et al. 2004, S. 16), entwickelt, ohne jedoch an den grundlegenden Einschränkungen der planbasierten Methodiken etwas zu ändern.

Bis auf die Etablierung des Begriffs Software Engineering blieb der erwartete Erfolg der verschiedenen Ansätze allerdings weitgehend aus. Die alten Probleme – überhöhte Kosten, lange Projektlaufzeiten und schlechte Qualität – existierten weiterhin. Hierfür werden u. a. die folgenden Gründe aufgeführt (Dogs und Klimmer 2005, S. 19 ff.):

- Planungen veralten schnell, da sich ständig neue Kundenanforderungen ergeben.
- Es wird viel Overhead (wie etwa Spezifikationen und Designs) geschaffen, der am eigentlichen Ziel vorbeigeht, für den Kunden qualitativ hochwertige Software zu entwickeln.
- Dem Faktor Mensch wird nur geringe Aufmerksamkeit im Entwicklungsprozess geschenkt.
- Tests erfolgen erst zu spät im Software-Entwicklungsprozess.
- Aus Fehlern wird nicht ausreichend für kommende Projekte gelernt.

Darüber hinaus führten eine zunehmende Globalisierung und ein dynamischeres Wirtschaftsumfeld zu immer volatileren Anforderungen. Kritiker führten an, dass die Inflextibilität der planbasierten Ansätze häufig dazu führe, dass Software entwickelt werde, welche die Nutzer zum Zeitpunkt ihrer Fertigstellung bereits nicht mehr benötigten.

Agiler Ansatz Aus der Unzufriedenheit mit den planbasierten Ansätzen entstanden zu Beginn der neunziger Jahre unabhängig voneinander verschiedene so genannte „leichtgewichtige“ Methodiken. Der Begriff „leichtgewichtig“ soll hierbei einen Gegensatz zu den bisher üblichen „schwergewichtigen“ Methodiken (im Sinne von prozess- und dokumentenlastig) ausdrücken und es einem Team erlauben, möglichst flexibel auf Änderungen der Nutzeranforderungen zu reagieren. Typische Vertreter dieser Ansätze sind etwa Scrum, eXtreme Programming (XP) und Adaptive Software Development (ASD).

Leichtgewichtige Methoden folgen hierbei einem „Just enough method“-Ansatz. Sie versuchen, die Prozesse zu vermeiden, die nur einen geringen Mehrwert für die fertige Software schaffen. Hierbei werden erzeugte Artefakte, wie etwa Spezifikationen, auf das Notwendigste beschränkt, um bei Änderungen den Anpassungsaufwand möglichst gering zu halten. Ein wesentliches Ziel der leichtgewichtigen Ansätze besteht darin, den Kunden die von ihnen priorisierte Funktionalität inkrementell (aufeinander aufbauend) in möglichst kurzen, iterativen Entwicklungszzyklen zur Verfügung zu stellen. Darüber hinaus erfolgen Tests der Software nicht erst zum Ende des Projekts, sondern sind fortlaufend in den Entwicklungsprozess integriert.

Anstatt den ständigen Wandel von Nutzeranforderungen als negativen Einfluss anzusehen, betrachten leichtgewichtige Methodiken diesen vielmehr als die Möglichkeit, Innovatives zu entwickeln und dadurch Wert für den Kunden zu schaffen. Auch ändert sich die Rolle der beteiligten Entwickler: Anstatt auf umfangreiche Dokumentation setzen leichtgewichtige Methodiken auf eine umfangreiche Kooperation sowie den Wissens- und Erfahrungsaustausch der Entwickler untereinander, wodurch dem Faktor Mensch eine stärkere Bedeutung als bei den planbasierten Ansätzen zukommt.

Bereits im Februar 2001 kamen Vertreter unterschiedlicher leichtgewichtiger Methodiken zusammen, um Gemeinsamkeiten und ein einheitliches Verständnis zu erarbeiten. Zum einen verständigten sich die Teilnehmer des Treffens darauf, von nun an den Begriff „agil“ anstatt leichtgewichtig zu verwenden, da „leichtgewichtig“ mit negativen Assoziationen belegt war. Zum anderen einigte man sich auf das so genannte „Agile Manifest“, das von den Anwesenden unterzeichnet wurde und welches das gemeinsame Verständnis bezüglich der agilen Methodiken ausdrückt (Fowler und Highsmith 2001, S. 28 ff.).

Die im „Agilen Manifest“ beschriebenen Werte drücken eine Verschiebung der Bedeutung aus, die verschiedenen Aspekten des Softwareentwicklungsprozesses beigemessen wird. So formulieren Fowler und Highsmith (2001, S. 29 f.):

We value:

- Individuals and interactions over processes and tools.
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Responding to change over following a plan.

Neben diesen vier Werten werden im Agilen Manifest darüber hinaus zwölf Prinzipien festgelegt, die sich an den Werten orientieren. Es wird allerdings bewusst darauf verzichtet, konkrete Programmierpraktiken vorzugeben. Die Prinzipien sind vielmehr hinreichend allgemein gehalten, um den Anwendern bei der Nutzung noch ausreichend Gestaltungsspielräume zu lassen.

Kritiker führen an, dass die der agilen Entwicklung zugrunde liegenden Prinzipien weder neu sind noch einen radikalen Paradigmenwechsel darstellen. So können die Anfänge der iterativen und inkrementellen Entwicklung bereits bis zum Ende der sechziger bzw. Anfang der siebziger Jahre zurückverfolgt werden (Fitzgerald et al. 2006, S. 200; Larman

und Basili 2003, S. 48). Neu ist allerdings das Zusammenwirken der verschiedenen agilen Praktiken, die sich gegenseitig verstärken sollen. Außerdem propagieren Vertreter agiler Methodiken, durchgängig Iterationen einzusetzen, um möglichst schnell Code und neue Softwareversionen zu erzeugen. Außerdem wird das ständige Umstellen lediglich grober Pläne propagiert, um Problemen und neuen Anforderungen flexibel zu begegnen, die aus der zunehmenden Geschwindigkeit resultieren, mit der sich das wirtschaftliche und technologische Umfeld ändern.

Dennoch stehen die Grundsätze der agilen Methodik zunächst im Gegensatz zu den Werten der Vertreter der planbasierten Softwareentwicklung, insbesondere das Erstellen vorab möglichst detaillierter Pläne, das Arbeiten in genau identifizierbaren Phasen und das Erstellen umfangreicher Dokumentationen. Im Zuge der scheinbaren Unvereinbarkeit der beiden Methodiken kam es zu einer Lagerbildung, die auch als „Method War“ bezeichnet wurde. In deren Rahmen wurden Vertreter der agilen Ansätze als „Hacker“ bezeichnet und Anhänger der planbasierten Ansätze von der Gegenseite mit „Dinosauriern“ verglichen (Boehm und Turner 2003, S. xiii; Boehm 2002, S. 3; Beck und Boehm 2003, S. 45).

Tabelle 3.8 stellt zusammenfassend anhand ausgewählter Bereiche dar, wie sich planbasierte und agile Ansätze voneinander unterscheiden.

Zunehmend wird in der Literatur allerdings betont, dass sowohl agile als auch planbasierte Methodiken ihre Berechtigung und ihre Stärken in unterschiedlichen Bereichen haben. Keine der beiden Methodiken kann also prinzipiell der anderen als überlegen angesehen werden (Boehm und Turner, S. 148 ff.). Tabelle 3.9 führt anhand ausgewählter Einflussgrößen die jeweiligen Anwendungsfelder an, in welchen die jeweilige Methodik für einen Einsatz am besten geeignet zu sein scheint.

Eine neuere Entwicklung stellen Ansätze dar, welche sowohl planbasierte als auch agile Elemente enthalten, um dadurch die Stärken beider Methodiken nutzen zu können. Die-

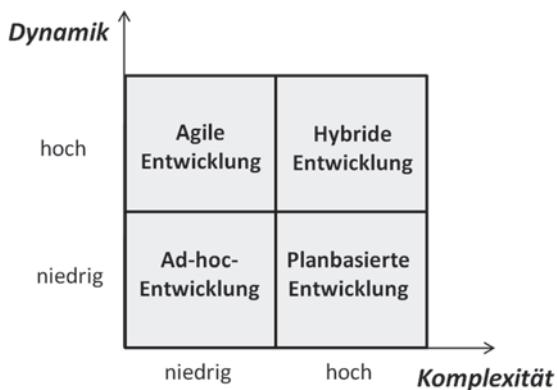
Tab. 3.8 Vergleich planbasierter und agiler Ansätze der Softwareentwicklung

Bereich	Planbasiert	Agil
Umgang mit Änderungen	Störgröße	Chance
Prozess	Sequenziell	Iterativ
Artefakte	Umfassend	Soviel wie nötig
Rolle der Entwickler	Ressource	Entscheidungsträger
Releases	Vollständig	Inkrementell

Tab. 3.9 Einflussgrößen auf die Auswahl der Methodik

Einflussgröße	Planbasiert	Agil
Nutzeranforderungen	Stabil	Instabil
Größe Entwicklungsteam	Eher groß	Eher klein
Sicherheitskritische Systeme	Geeignet	Ungeeignet
Art der entwickelten SW	Standard-SW	Individual-SW

Abb. 3.34 Vorteilhaftigkeit unterschiedlicher Entwicklungsmethodiken



se hybriden Methodiken planen etwa detailliert das Zusammenspiel einzelner Softwarekomponenten. Die Entwicklung der einzelnen Komponenten selbst erfolgt dann allerdings nach einem agilen Ansatz.

Aufbauend auf den in diesem Abschnitt angestellten Überlegungen wird in Abb. 3.34 anhand der beiden Dimensionen Dynamik und Komplexität eine Matrix dargestellt, welche Entwicklungsmethodik sich in welchem Kontext tendenziell am ehesten eignet. Die Achse Dynamik steht für ein Umfeld mit hoher wirtschaftlicher oder technologischer Unsicherheit, in dem sich Nutzeranforderungen häufig ändern. Die Dimension Komplexität beinhaltet u. a. Aspekte wie die Anzahl der Softwareentwickler bzw. die Größe des beteiligten Teams oder auch die Notwendigkeit der Integration von Software und Hardware von Drittanbietern.

Diese Darstellung kann natürlich nur ein erster Ansatzpunkt für die Unterstützung einer Auswahlentscheidung sein, da in einem organisatorischen Kontext auch andere Faktoren, wie etwa die Erfahrung mit einer Methodik oder die spezifische Art der entwickelten Software, eine wichtige Rolle spielen.

3.4.2 Personalführung in der Softwareentwicklung

Nicht erst nach dem Aufkommen, der im letzten Abschnitt diskutierten agilen Entwicklungsmethodiken setzt sich die Erkenntnis durch, dass der Faktor Mensch im Rahmen des Software-Entwicklungsprozesses eine entscheidende Rolle spielt. Der Einfluss eines solchen „weichen“ Faktors kann hierbei eine größere Rolle spielen als die Bedeutung technischer Faktoren, wie etwa die Art der eingesetzten Technologie oder der Einsatz eines speziellen Werkzeugs. Ausdruck findet dies u. a. in der Entwicklung des People Capability Maturity Models (CMM) des Software Engineering Institute, welches die Überlegungen des CMMI-Reifegradmodells eines Entwicklungsprozesses auf das Management des Personals in einem Unternehmen überträgt (SEI 2010).

Vor dem Hintergrund der Bedeutung eines guten Personalmanagements für den Erfolg von Software-Entwicklungsprojekten wird im Folgenden genauer betrachtet, inwiefern Softwareentwickler spezifische Charakteristika besitzen, durch die sie sich von anderen Berufsgruppen unterscheiden und von diesen abgrenzen lassen. Aufbauend darauf wird der Ansatz des „Person-Job-Fit“ als eine Möglichkeit der Zuordnung von Mitarbeitern zu Rollen und Aufgaben vorgestellt. Abschließend wird das Thema Mitarbeitermotivation kurz diskutiert.

Durch die Möglichkeit, Persönlichkeitseigenschaften und -typen zu bestimmen und zu messen, nahm in den letzten Jahrzehnten das Interesse an Persönlichkeitstheorien in der Wissenschaft stark zu. Da Einstellungen, Überzeugungen, Wahrnehmung und Verhalten einer Person alles Aspekte sind, die zumindest zu einem bestimmten Grad von deren Persönlichkeit beeinflusst werden, hat die Persönlichkeitsstruktur von Menschen auch einen erheblichen Einfluss auf in der Softwareentwicklung ablaufende Prozesse.

Unter Persönlichkeit werden die einzigartigen psychologischen Merkmale eines Individuums verstanden, die eine Vielzahl von charakteristischen Verhaltensmustern prägen. So können Persönlichkeitseigenschaften zumindest teilweise das Interesse an bestimmten Berufen erklären (Costa et al. 1984). Softwareentwickler unterscheiden sich anhand ihres Persönlichkeitsprofils vom Durchschnitt der Bevölkerung. So lassen sie sich tendenziell weniger stark von ihren Gefühlen leiten, sind eher introvertiert und arbeiten in vielen Fällen lieber alleine als im Team (Capretz 2003).

Trotz der ähnlichen Wesenszüge von Softwareentwicklern existieren natürlich auch innerhalb dieser Gruppe große individuelle Unterschiede; es lässt sich also nicht von „dem einen“ typischen Softwareentwickler sprechen. Außerdem fallen im Rahmen eines Softwareentwicklungsprojekts unterschiedliche Aufgaben (z. B. Teamleiter, Qualitätsmanager, Tester, Programmierer etc.) an, die unterschiedliche Anforderungen stellen.

Mit der Zuordnung von Personen mit ihren spezifischen Fähigkeiten auf bestimmte Aufgaben beschäftigt sich der Ansatz des „Person-Job-Fit“. Das in Abb. 3.35 dargestellte Modell besagt, dass die persönlichen Eigenschaften und Fähigkeiten einer Person den An-

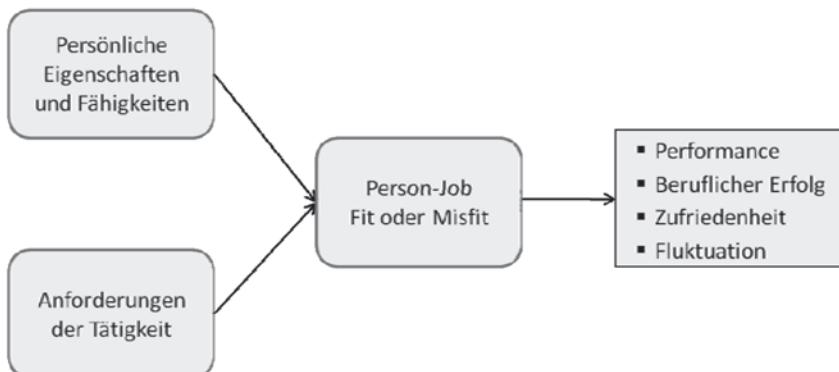


Abb. 3.35 Person-Job-Fit. (in Anlehnung an Lauver und Kristof-Brown 2001)

Software roles	Capabilities										
	Analysis	Decision-making	Intrapersonal	Innovation and creativity	Judgment	Tenacity	Stress tolerance	Organizational	Environment and knowledge	Interpersonal	Management
Team leader	✓	✓		✓		✓		✓	✓	✓	✓
Quality manager	✓	✓	✓	✓		✓		✓		✓	✓
Requirements engineer	✓			✓	✓			✓	✓	✓	✓
Designer	✓	✓	✓		✓	✓		✓	✓	✓	✓
Programmer	✓	✓	✓		✓	✓		✓	✓	✓	✓
Maintenance and support specialist				✓	✓	✓		✓	✓	✓	✓
Tester		✓	✓	✓	✓	✓		✓	✓	✓	✓
Configuration manager		✓	✓	✓	✓	✓		✓	✓	✓	✓

Abb. 3.36 Matching von Entwicklerrollen auf Anforderungsprofile. (Acuña et al. 2006, S. 98)

forderungen einer Tätigkeit entsprechen müssen. Ein hoher Fit führt tendenziell zu einer hohen Performance und beruflichem Erfolg des Mitarbeiters sowie zu hoher Zufriedenheit und geringer Fluktuation.

Konkret angewandt auf die Softwareindustrie leiten Acuña et al. (2006) ausgehend von verschiedenen Persönlichkeitseigenschaften diejenigen Fähigkeiten („Capabilities“) ab, welche typisch für Personen mit diesen Charaktereigenschaften sind. In einem zweiten Schritt werden diese Fähigkeiten mit den spezifischen Anforderungen von acht verschiedenen Rollenprofilen im Rahmen eines Softwareentwicklungsprojekts in einer Matrix in Übereinstimmung gebracht (siehe Abb. 3.36).

Um die Eignung eines Mitarbeiters für eine bestimmte Rolle zu beurteilen, werden also etwa mittels eines Persönlichkeitstests dessen Charakterzüge bestimmt. Die daraus resultierenden Fähigkeiten werden schließlich mit der Matrix auf Übereinstimmung geprüft. Hierbei ist zu bemerken, dass dieses Matching nur einen ersten Hinweis auf die Eignung eines Mitarbeiters für eine Tätigkeit geben kann – so bleiben bei diesem Modell etwa spezifische Fertigkeiten der Mitarbeiter, Berufserfahrung, Besonderheiten der jeweiligen Rolle oder auch organisatorische Spezifika außen vor.

Eng verwandt mit der Zuordnung von Personen zu Aufgaben und Rollen ist der Punkt, wie sich Softwareentwickler am besten motivieren lassen. Allerdings lässt sich diese Frage nur schwer allgemeinernd beantworten, da Faktoren, die motivierend oder demotivierend auf Entwickler wirken, häufig vom jeweiligen Kontext sowie den individuellen Bedürfnissen einer Person abhängen. Abbildung 3.37 stellt ein allgemeines Modell zum Zustandekommen von Motivation und Demotivation bei Softwareentwicklern dar.

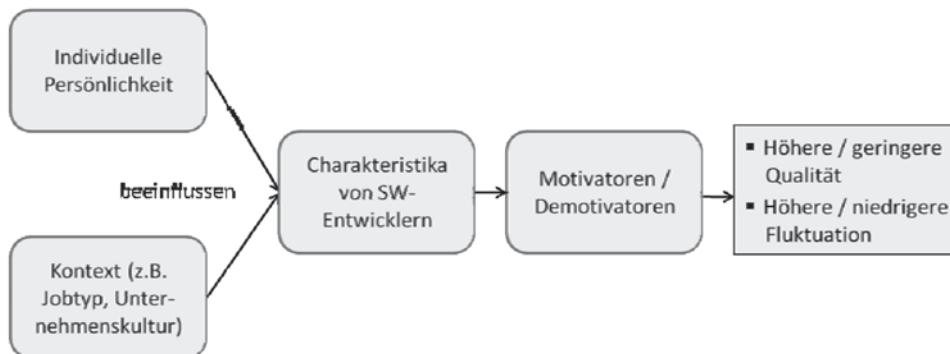


Abb. 3.37 Zustandekommen von Motivation. (eigene Darstellung in Anlehnung an Beecham et al. 2008)

Trotz der individuellen und kontextspezifischen Umstände wird in der Literatur vor allem eine Identifikation mit der anstehenden Aufgabe als genereller Motivator von Softwareentwicklern genannt: Neben einem persönlichen Interesse an der Aufgabe müssen die verfolgten Ziele klar abgesteckt werden und die Bedeutung der Aufgabe für das ganze Projekt verständlich sein. Zudem wirken das Vorhandensein klarer Karrierepfade sowie ein abwechslungsreiches und herausforderndes Tätigkeitsspektrum motivierend (vgl. Beecham et al. 2008).

Besonders häufig werden eine schlechte Arbeitsumgebung, etwa das Fehlen von entsprechenden Ressourcen, wie Hard- und Software, als demotivierend für Softwareentwickler genannt. Weiterhin wirken ein schlechtes Management, z. B. das Einberufen überflüssiger Meetings, sowie eine schlechte Bezahlung demotivierend (vgl. Beecham et al. 2008).

Bei der Implementierung von Maßnahmen zur Steigerung der Motivation ist allerdings der bereits diskutierte Fit zwischen der Aufgabe und der jeweiligen Person zu beachten, da ein Misfit sich zwangsläufig demotivierend auswirkt. Anders ausgedrückt lässt sich die „falsche“ Zuordnung einer ungeeigneten Person zu einer nicht passenden Aufgabe nur begrenzt durch andere Faktoren korrigieren.

3.4.3 Add-on: Fachkräftemangel in der Software- und IT-Industrie

Die zuvor dargestellten Überlegungen zu motivationssteigernden Maßnahmen für Softwareentwickler sind auch vor dem Hintergrund des IT-Fachkräftemangels einzuordnen. So fehlten im Jahr 2013 laut dem Branchenverband Bitkom alleine in Deutschland rund 39.000 IT-Fachkräfte. Überraschend ist jedoch, dass sich dieser Fachkräftemangel nicht in den Gehältern niederschlägt. So sind die Jahreseinkommen 2013 gemäß einer Studie der Computerwoche und der Vergütungsberatung Personalmarkt lediglich um 1 % gestiegen. Überdurchschnittliche Gehaltszuwächse gab es nur in den Bereichen SAP-Beratung, Webdesign und IT-Coaching (Königes 2013).

Um ein besseres Verständnis für die Folgen insbesondere aus Sicht der Software- und Serviceanbieter zu gewinnen, wurden im Rahmen einer Studie von Lünendonk (Streicher et al. 2013) die Probleme bei der Gewinnung von IT-Fachkräften näher untersucht. Darüber hinaus wurden die Konsequenzen thematisiert. Grundlage ist die bereits im ersten Kapitel dieses Buches durchgeführte Unterscheidung zwischen IT-Beratungs- und IT-Serviceunternehmen.

Die teilnehmenden 99 Unternehmen wurden im Rahmen der Umfrage gebeten, die Probleme bei der Gewinnung von qualifizierten IT-Arbeitskräften auf Basis einer Skala (+2 = „sehr schwierig“ bis -2 = „gar nicht schwierig“) zu beschreiben. Rund vier von fünf Teilnehmern (79,1 %) aus dem Kreis der IT-Beratungsunternehmen bezeichnen die Gewinnung als „sehr schwierig“ oder „schwierig“. Nur 6,4 % empfinden diese Aufgabe als „weniger schwierig“ oder „gar nicht schwierig“. Die mittlere Bewertung aller IT-Beratungsunternehmen liegt bei 1,06 (Streicher et al. 2013, S. 65).

Zudem wurde im Rahmen der Studie das durchschnittliche Alter der Berater und IT-Experten analysiert. Der Mittelwert der genannten firmenbezogenen Durchschnittswerte liegt bei den IT-Beratungsunternehmen bei 37,6 Jahre (Median 38). Die unternehmensbezogenen Extremwerte liegen bei 46 Jahren am oberen Rand und 30 Jahren am unteren Rand (Streicher et al. 2013, S. 66).

Im Rahmen der Studie untersuchte Lünendonk auch, wie hoch der Frauenanteil bei den Beratern und IT-Experten ausfällt. Der Mittelwert aus den von den IT-Beratungsunternehmen genannten Durchschnittswerten beträgt 15,9 %, der Median liegt bei 17 %.

Interessant ist, dass die 25 nach Inlandsumsatz führenden IT-Beratungs- und Systemintegratoren einen deutlich höheren Frauenanteil haben als die kleineren Unternehmen. Während der Frauenanteil bei den Top 25 im statistischen Mittel 21,0 % beträgt, liegt er bei den übrigen untersuchten Anbietern mit 13,6 % deutlich darunter.

Betrachten wir nun den Frauenanteil unter den IT-Experten der IT-Service-Unternehmen: dieser liegt mit durchschnittlich 18,3 % über dem ermittelten Wert der untersuchten IT-Beratungsfirmen, der Median beträgt 17 % (Streicher et al. 2013, S. 66–67).

Der Mittelwert der firmenbezogenen Durchschnittswerte für das Alter der IT-Experten bei den IT-Service-Unternehmen beträgt 40,9 Jahre (Median 41 Jahre). Die Durchschnittswerte der IT-Beratungs- und Systemintegrations-Unternehmen liegen mit einem Mittelwert von 37,6 und einem Median von 38 Jahren etwas niedriger (Streicher et al. 2013, S. 67).

Im Rahmen der ergänzend durchgeführten Befragung von Anwenderunternehmen wurde auch das Durchschnittsalter der internen IT-Mitarbeiter erfasst. Der Mittelwert aus den unternehmensbezogenen Durchschnittswerten ergibt ein Alter von 38,7 bei einem Median von 40 Jahren. Damit sind die Altersstrukturen der IT-Dienstleister auf ähnlichem Niveau wie die der großen Anwenderunternehmen.

Die Schwierigkeiten, geeignete IT-Fachkräfte in der notwendigen Anzahl zu rekrutieren, erfordern hohe Anstrengungen in der Personalsuche und -rekrutierung. Im Rahmen der Studie wurde daher gefragt, wie die Teilnehmer ihre IT-Fachkräfte rekrutieren. Auf

einer Skala von +2 = „sehr häufig“ bis -2 = „gar nicht“ sollten die Teilnehmer die Häufigkeiten der vorgegebenen Rekrutierungswege nennen.

Am häufigsten rekrutieren alle befragten IT-Beratungsunternehmen ihre Mitarbeiter über „Empfehlungen durch Mitarbeiter“ (1,00). Es folgt die „Eigene Homepage“ (0,97) vor „Jobbörsen/Jobmessen“ (0,82) und „Hochschulkooperationen“ (0,57) (Streicher et al. 2013, S. 69). Demgegenüber werden „Personalberater/Headhunter“ (0,23) und „Direktansprache“ (0,22) deutlich seltener eingesetzt. Auch „Initiativbewerbung“ (0,23) und „Social Media“ (0,05) spielen eher eine untergeordnete Rolle. Gar keine gängigen Rekrutierungswege sind nach Auskunft der Unternehmen hingegen Mitarbeiter „von Kunden“ (-1,29), „von Wettbewerbern“ (-0,33) und „Fachmessen“ (-0,53).

Zudem wurden auch die IT-Service-Anbieter befragt, wie sie ihr Personal rekrutieren. Auf Basis einer Skala von +2 = „sehr häufig“ bis -2 = „gar nicht“ sollten auch die teilnehmenden IT-Service-Unternehmen die Häufigkeit der vorgegebenen Rekrutierungswege bewerten. Im Durchschnitt wird der Weg über „Jobbörsen/Jobmessen“ (1,25) am häufigsten gewählt. Mit einem Durchschnittswert von 1,21 folgt dahinter die unternehmenseigene Homepage, vor „Empfehlung durch Mitarbeiter“ (0,82) und „Hochschulkooperationen“ (0,80). Während „Initiativbewerbungen“ (0,52) auch ein beträchtlicher Stellenwert zu kommt, sind „Social Media“ (0,07) und das Überwechseln von Mitarbeitern der Kundenunternehmen (-0,84) und „von Wettbewerbern“ (-0,23) ebenso selten wie Personalkontakte auf „Fachmessen“ (-0,33) (Streicher et al. 2013, S. 69).

Die eingesetzten HR-Instrumente und Medien der 25 nach Inlandsumsatz führenden IT-Beratungs- und Systemintegrations-Unternehmen bei der Personalsuche haben eine durchgängig höhere Gewichtung. Alle von Lünendonk abgefragten Rekrutierungswege werden von den Top-25-IT-Beratern wesentlich häufiger genutzt als bei den übrigen Anbietern. Deutliche Abweichungen finden sich bei der „Eigenen Homepage“, der „Initiativbewerbung“ und der Abwerbung von Mitarbeitern „von Wettbewerbern“. Die tendenziell höhere Markenbekanntheit der nach Umsatz in Deutschland führenden IT-Beratungs-Unternehmen schlägt sich auch im HR-Branding und der Attraktivität als Arbeitgeber nieder, so dass diese drei Kommunikationskanäle von Bewerbern und Interessenten konsequenterweise häufiger genutzt werden. Vergleichsweise kleinere IT-Berater müssen ihre Anstrengungen und ihre HR-Budgets deutlich vergrößern, um ihre Markenbekanntheit bei Bewerbern zu erhöhen.

Diese Sichtweise der Anbieter korrespondiert mit der angespannten Lage am IT-Arbeitsmarkt. Die Branche benötigt demnach dringend gut ausgebildete IT-Nachwuchskräfte, um die Vielzahl an anspruchsvollen IT-Projekten für ihre Kunden zu bewältigen. Aufgrund des Kostendrucks sowie der Konzentration auf bestimmte Kernaufgaben haben viele Kundenunternehmen in der Vergangenheit immer mehr IT-Aufgaben an externe IT-Dienstleistungsunternehmen ausgelagert und werden es auch weiterhin tun (siehe hierzu auch Kap. 5 dieses Buches).

Literatur

- Acuña S, Juristo N, Moreno A (2006) Emphasizing human capabilities in software development. *IEEE Softw* 23:94–101
- Adams W, Yellen J (1976) Commodity bundling and the burden of monopoly. *Q J Econ* 90:475–498
- Ahtiala P (2006) The optimal pricing of computer software and other products with high switching costs. *Int Rev Econ Finance* 15:202–211
- Bakos Y, Brynjolfsson E (1999) Bundling information goods: pricing, profits, and efficiency. *Manage Sci* 45:1613–1630
- Balzert H (2000) Lehrbuch der Software-Technik. Software-Entwicklung, 2. Aufl. Spektrum Akademischer Verlag, Heidelberg
- Beck K, Boehm B (2003) Agility through discipline: a debate. *Computer* 36:44–46
- Beecham S, Baddoo N, Hall T, Robinson H, Sharp H (2008) Motivation in software engineering: a systematic literature review. *Inf Softw Technol* 50:860–878
- Beitel P, Schiereck D (2003) Zum Erfolg von Akquisitionen und Zusammenschlüssen zwischen Banken: Eine Bestandsaufnahme der empirischen Forschung. *Z Gesamte Bank- Börsenwes* 51(Sonderdruck):501–516
- Bhargava K, Choudhary V (2008) When is versioning optimal for information goods? *Manage Sci* 54:1029–1035
- Blohm H (1980) Kooperation. In: Grochla E (Hrsg) Handwörterbuch der Organisation. Schäffer Poeschel, Stuttgart, S 1111–1117
- Boehm B (1986) A spiral model of software development and enhancement. *ACM SIGSOFT Softw Eng Notes* 11:14–24
- Boehm B (2002) Get ready for agile methods, with care. *Computer* 35:1–7
- Boehm B (2006) A view of 20th and 21st century software engineering. In: Proceedings of the 28th international conference on software engineering, ICSE '06. S 12–29
- Boehm B, Turner R (2003) Balancing agility and discipline: a guide for the perplexed. Addison-Wesley Longman, Boston
- Bontis N, Chung H (2000) The evolution of software pricing: from box licenses to application service provider models. *Electron Netw Appl Policy* 10:246–255
- Brandenburger A, Nalebuff B (1996) Co-opetition. A revolutionary mindset that combines competition and cooperation. The game theory strategy that's changing the game of business. Doubleday, New York
- Bruner RF (2001) Does M&A pay? A survey of evidence for the decision-maker. Batten Institute, Charlottesville
- Budzinski O, Christiansen A (2007) The Oracle/PeopleSoft case: unilateral effects, simulation models and econometrics in contemporary merger control. *Legal Issues Econ Integr* 34(2):1–34
- Buxmann P (2002) Strategien von Standardsoftware-Anbietern: Eine Analyse auf Basis von Netzeffekten. *Z betriebswirtsch Forsch* 54:442–457
- Buxmann P, König W, Fricke M, Hollich F, Martín Diaz L, Weber S (2004) Inter-organizational cooperation with SAP solutions – design and management of supply networks, 2. Aufl. Springer, Berlin
- Buxmann P, Strube J, Pohl G (2007) Cooperative pricing in digital value chains – the case of online music. *J Electron Commer Res* 8:32–40
- Buxmann P, Hess T, Lehmann S (2008b) Software as a service. *Wirtschaftsinformatik* 50:500–503
- Capretz L (2003) Personality types in software engineering. *Int J Hum-Comput Stud* 58:207–214
- Costa PT, McCrae RR, Holland JL (1984) Personality and vocational interests in an adult sample. *J Appl Psychol* 69:390–400
- Cusumano MA (2007) The changing labyrinth of software pricing. *Commun ACM* 50:19–22
- Datta DK, Pinches GE, Narayanan VK (1992) Factors influencing wealth creation from mergers and acquisitions: a meta-analysis. *Strateg Manage J* 13(1):67–84

- Diller H (2008) Preispolitik. Kohlhammer, Stuttgart
- Dogs C, Klimmer T (2005) Agile Software-Entwicklung kompakt. Mitp-Verlag, Bonn
- Fitzgerald B, Hartnett G, Conboy K (2006) Customising agile methods to software practices at Intel Shannon. *Eur J Inf Syst* 15:200–213
- Florissen A (2008) Preiscontrolling – Rationalitätssicherung im Preismanagement. *Z Control Manage* 52:85–90
- Fowler M, Highsmith J (2001) The agile manifesto. *Softw Develop* 28–32
- Friedewald M, Rombach HD, Stahl P, Broy M, Hartkopf S, Kimpeler S, Kohler K, Wucher R, Zoche P (2001) Analyse und Evaluation der Softwareentwicklung in Deutschland. Studie für das Bundesministerium für Bildung und Forschung (BMBF). GfK Marktforschung, Nürnberg
- Gao L, Iyer B (2006) Analyzing complementarities using software stacks for software industry acquisitions. *J Manage Inf Syst* 23:119–147
- Günther O, Tamm G, Leymann F (2007) Pricing web services. *Int J Bus Process Integr Manage* 2:132–140
- Halebian J, Devers CE, McNamara G, Carpenter MA, Davison RB (2009) Taking stock of what we know about mergers and acquisitions: a review and research agenda. *J Manage* 35(3):469–502
- Harmon R, Raffo D, Faulk S (2005) Value-based pricing for new software products: strategy insights for developers. <http://www.cpd.ogi.edu/MST>
- Hill S (2008) SaaS economics seem to favor users more than vendors. *Manuf Bus Technol* 48
- Hindel B, Hörmann K., Müller M, Schmied J (2004) Basiswissen Software-Projektmanagement. Dpunkt.Verlag GmbH, Heidelberg
- Hirnle C, Hess T (2007) Investing into IT infrastructures for inter-firm networks: Star Alliance's move to the common platform. *Electron J Virtual Organ Netw* 8:124–143
- Hitt LM, Chen P (2005) Bundling with customer self-selection: a simple approach to bundling low marginal cost goods. *Manage Sci* 51:1481–1493
- Homburg C, Koschate N (2005) Behavioral Pricing – Forschung im Überblick. *Z Betriebswirtsch* 75:383–423. u. 501–524
- Homburg C, Krohmer H (2006) Marketingmanagement. Gabler, Wiesbaden
- Hutzschenreuter T, Stratigakis G (2003) Die feindliche Übernahme von PeopleSoft durch Oracle – der Beginn einer Konsolidierungswelle. *Signale* 18(3):10–11
- Izci E, Schiereck D (2010) Programmierte Wertgenerierung durch M&A in der Business-Software-industrie? *Mergers Acquis Rev* 2:69–74
- King DR, Dalton DR, Daily CM, Covin JG (2004) Meta-analyses of post-acquisition performance: indications of unidentified moderators. *Strategic Manage J* 25(2):187–200
- Kittlaus HB, Rau C, Schulz J (2004) Software-Produkt-Management: Nachhaltiger Erfolgsfaktor bei Herstellern und Anwendern. Springer, Heidelberg
- Königes H (2013) Gehaltsrunde für IT-Profis fällt mager aus. <http://www.computerwoche.de/a/gehaltsrunde-fuer-it-profis-faellt-mager-aus,2548666>
- Kude T, Dibbern J, Heinzl A (2012) Why do complementors participate? An analysis of partnership networks in the enterprise software industry. *IEEE Trans Eng Manage* 59:250–265
- Küpper H-U (2008) Controlling. Konzeption, Aufgaben, Instrumente. Schäffer-Poeschel, Stuttgart
- Laamanen T, Brauer M, Junna O (2013) Performance of acquirers of divested assets: evidence from the U.S. software industry. *Strategic Manage J* 1–22
- Lambrecht A, Skiera B (2006) Paying too much and being happy about it: existence, causes, and consequences of tariff-choice biases. *J Mark Res* 43:212–223
- Larman C, Basili VR (2003) Iterative and incremental development: a brief history. *Computer* 36:47–56
- Lauver KJ, Kristof-Brown A (2001) Distinguishing between employees' perceptions of person-job and person-organization fit. *J Vocat Behav* 59:454–470

- Léger P-M, Quach L (2009) Post-merger performance in the software industry: the impact of characteristics of the software product portfolio. *Technovation* 29:704–713
- Lehmann S, Buxmann P (2009) Preisstrategien von Softwareanbietern. *Wirtschaftsinformatik* 51:519–529
- Liebowitz SJ, Margolis SE (1994) Network externality: an uncommon tragedy. *J Econ Perspect* 8:133–150
- Liebowitz SJ, Margolis SE (2001) Winners, losers & microsoft competition and antitrust in high technology. Independent Institute, Oakland
- Linde F (2008) Pricing-Strategien bei Informationsgütern. *WISU* 2:208–214
- Loefert C (2007) Unternehmensreputation und M&A-Transaktionen – Bewertung strategischer Entscheidungen in der US-amerikanischen Finanz- und Telekommunikationsindustrie. Gabler, Wiesbaden
- Lünendonk T (2007) Führende Standard-Software-Unternehmen in Deutschland. Bad Wörishofen
- MacKinlay CA (1997) Event studies in economics and finance. *J Econ Lit* 35(1):13–39
- Marn MV, Roegner EV, Zawada CC (2003) The power of pricing. *The McKinsey Quarterly* 1
- Martín Díaz L (2006) Evaluation of cooperative planning in supply chains. An empirical approach of the European automotive industry. Gabler, Wiesbaden
- Mcnamara GM, Halebian JJ, Dykes BJ (2008) The performance implications of participating in an acquisition wave: early mover advantages, bandwagon effects, and the moderating influence of industry characteristics and acquirer tactics. *Acad Manage J* 51(1):113–130
- Meglio O, Risberg A (2011) The (mis)measurement of M&A performance – a systematic narrative literature review. *Scand J Manage* 27(4):418–433
- Meyer R (2008) Partnering with SAP vol. 1: business models for software companies. Books on Demand, Norderstedt
- Nalebuff BJ (2004) Bundling as an entry barrier. Working Paper. School of Management, Yale University, New Haven
- Nieschlag R, Dichtl E, Hörschgen H (2002) Marketing, 19. Aufl. Duncker & Humblot, Berlin
- Olderog T, Skiera B (2000) The benefits of bundling strategies. *Schmalenbach Bus Rev* 1:137–160
- o. V. (2005) Mit Siebel kauft Oracle Marktanteile. Computerwoche. <http://www.computerwoche.de/nachrichten/566287>
- o. V. (2013) Oracle kauft Netzwerkkomponentenanbieter für 1,7 Mrd. Dollar. <http://www.computerbase.de/news/2013-02/oracle-kauft-netzwerkkomponentenanbieter-fuer-1.7-mrd.-dollar/>
- Parbel M (2005) Symantec übernimmt Veritas: Security und Storage wachsen zusammen. <http://www.crn.de/showArticle.jhtml?articleID=184423707>
- Pepels W (1998) Einführung in das Preismanagement. Oldenbourg, München
- Picot A, Bub U, Krcmar H (2011) Die Zukunft der Telekommunikation. *Wirtschaftsinformatik* 5:253–255
- Picot A, Hess T, Hörndlein C, Jablonka C, Kaltenecker N, Neuburger R, Schreiner M, Werbik A, Gold B (2014) Deutsche Software Champions: Bestandsaufnahme – Stellschrauben – Perspektiven, Abschlussbericht zum gleichnamigen, vom BMBF geförderten Projekt, im Erscheinen
- Raymond ES (1999) The cathedral and the bazaar. O'Reilly, Sebastopol
- Reinicke A (2007) Internes Dokument zum Management von Vertriebsprozessen. erstellt von w+p consulting AG, Mannheim
- Rullkötter L (2008) Preismanagement – Ein Sorgenkind? *Z Control Manage* 52:92–98
- Schief M (2014) Business models in the software industry: the impact on firm and M&A performance. Springer, Wiesbaden
- Schief M, Buxmann P, Schiereck D (2013) Fusionen und Unternehmensübernahmen in der Softwareindustrie – Forschungsergebnisse im Bereich Erfolgsdeterminanten. *Wirtschaftsinformatik* 421–433

- Schiff C (2007) What's driving the latest wave of business performance management mergers and acquisitions? <http://www.b-eye-network.com/view/4120>
- Schmalensee R (1984) Gaussian demand and commodity bundling. *J Bus* 57:211–230
- Schmidt C, Weinhardt C, Horstmann R (1998) Internet-Auktionen – Eine Übersicht für Online-Versteigerungen im Hard- und Softwarebereich. *Wirtschaftsinformatik* 40:450–457
- SEI (2010) People CMM. <http://www.sei.cmu.edu/cmmi/tools/peoplecmm>
- Shapiro C, Varian HR (1998) Information rules: a strategic guide to the network economy. Harvard Business School Press, Boston
- Shapiro C, Varian HR (1999) Information rules. Harvard Business School Press, Cambridge
- Sieg G (2005) Spieltheorie. Oldenbourg, München
- SIIA (2006) Key trends in software pricing and licensing: a survey of software industry executives and their enterprise customers. Washington
- Simon H (1992) Preismanagement. Gabler, Stuttgart
- Simonson I, Tversky A (1992) Choice in context: tradeoff contrast and extremeness aversion. *J Mark Res* 29:281–295
- Skiera B (1999a) Preisdifferenzierung. In: Albers S, Clement M, Peters K (Hrsg) Marketing mit interaktiven Medien. Strategien zum Markterfolg. F.A.Z. Institut, Frankfurt a. M., S 283–296
- Skiera B (1999b) Mengenbezogene Preisdifferenzierung bei Dienstleistungen. Gabler, Wiesbaden
- Skiera B, Spann M (1998) Gewinnmaximale zeitliche Preisdifferenzierung für Dienstleistungen. *Z Betriebswirtsch* 68:703–718
- Skiera B, Spann M (2000) Flexible Preisgestaltung im Electronic Business. In: Weiber R (Hrsg) Handbuch Electronic Business: Informationstechnologien – Electronic Commerce – Geschäftsprozesse. Gabler, Wiesbaden, S 539–557
- Skiera B, Spann M (2002) Preisdifferenzierung im Internet. Roadmap to E-Business – Wie Unternehmen das Internet erfolgreich nutzen. In: Schögel M, Tomczak T, Belz C (Hrsg) Thexis, St. Gallen, S 270–284
- Skiera B, Spann M, Walz U (2005) Erlösquellen und Preismodelle für den Business-to-Consumer-Bereich im Internet. *Wirtschaftsinformatik* 47:285–293
- Smith GE, Nagle TT (1995) Frames of reference and buyers' perception of price and value. *Calif Manage Rev* 38:98–116
- Statista W (2014b) Verteilung der weltweiten M&A Deals im Jahr 2013 nach Branchen (in Milliarden US-Dollar). <http://de.statista.com/statistik/daten/studie/234097/umfrage/verteilung-der-weltweiten-munda-deals-nach-branchen/>
- Streicher H, Lünendonk T, Zillmann M, Bochtler R (2013) Der Markt für IT-Beratung und IT-Service in Deutschland: eine unabhängige Analyse führender IT-Dienstleister sowie Anwenderunternehmen. Lünendonk-Studie 2013:1–97
- Stremersch S, Tellis GJ (2002) Strategic bundling of products and prices: a new synthesis for marketing. *J Mark* 66:55–72
- Sundararajan A (2004) Nonlinear pricing of information goods. *Manage Sci* 50:1660–1673
- Thanos IC, Papadakis VM (2011) The use of accounting-based measures in measuring M&A performance: a review of five decades of research. In: Cooper CL, Finkelstein S (Hrsg) Advances in mergers & acquisitions. Emerald, Bradford, S 103–120
- Varian HR (1997) Versioning information goods. Working Paper. University of California, Berkeley. <http://people.ischool.berkeley.edu/~hal/Papers/version.pdf>
- Viswanathan S, Anandalingam G (2005) Pricing strategies for information goods. *Sadhana – J Indian Acad Sci* 30:257–274
- Weber J, Schäffer U (2008) Einführung in das Controlling. Schäffer-Poeschel, Stuttgart
- Wirtz BW (2003) Mergers & acquisitions management. Strategie und Organisation von Unternehmenszusammenschlüssen. Gabler, Wiesbaden

- Wu S, Anandalingam G (2002) Optimal customized bundle pricing for information goods. In: Proceedings workshop on information technology and systems, Barcelona
- Wu S, Hitt LM, Chen P, Anandalingam G (2008) Customized bundle pricing for information goods: a nonlinear mixed-integer programming approach. *Manag Sci* 54:608–622
- Zerdick A, Picot A, Schrage K, Artopé A, Goldhammer K, Lange UT, Vierkant E, López-Escobar E, Silverstone R (1999) Die Internet-Ökonomie: Strategien für die digitale Wirtschaft. Springer, Berlin

Aufbauend auf den Strategien für Softwareanbieter wollen wir nun Geschäftsmodelle in der Softwareindustrie behandeln, deren Gestaltung und Analyse von hoher Bedeutung für das Management ist: So gab in einer Umfrage der Economist Intelligence Unit die Mehrheit der befragten Senior Manager an, dass sie die Bedeutung innovativer Geschäftsmodelle höher einschätzt als das Anbieten neuer Produkte oder Services (Amit und Zott 2012, S. 41). Ebenso hat die Betrachtung von Geschäftsmodellen in der Wissenschaft in den letzten Jahren erheblich zugenommen (Veit et al. 2014).

Wie so häufig, existiert eine Vielzahl von Definitionen. Wir wollen uns eine ausführliche Erörterung der verschiedenen Definitionen des Begriffs Geschäftsmodell hier sparen und uns stattdessen an der klassischen Definition von Osterwalder et al. (2005, S. 5) orientieren:

A business model is a conceptual tool that contains a set of elements and their relationships and allows expressing the business logic of a specific firm. It is a description of the value a company offers to one or several segments of customers and of the architecture of the firm and its network of partners for creating, marketing, and delivering this value and relationship capital, to generate profitable and sustainable revenue streams.

Die Beschreibung der Wertschöpfungskette eines Unternehmens wird dabei häufig als die Grundlage der Gestaltung und Analyse von Geschäftsmodellen angesehen. Daher wollen wir in diesem Kapitel mit der Vorstellung einer auf die Softwareindustrie angepassten „Software Value Chain“ beginnen (Abschn. 4.1). Darauf aufbauend stellen wir in Abschn. 4.2 ein Framework für die Analyse und Gestaltung von Geschäftsmodellen in der Softwareindustrie vor. Das Kapitel schließt in Abschn. 4.3 mit der Vorstellung eines Software-Werkzeuges zur Analyse und Gestaltung solcher Geschäftsmodelle.

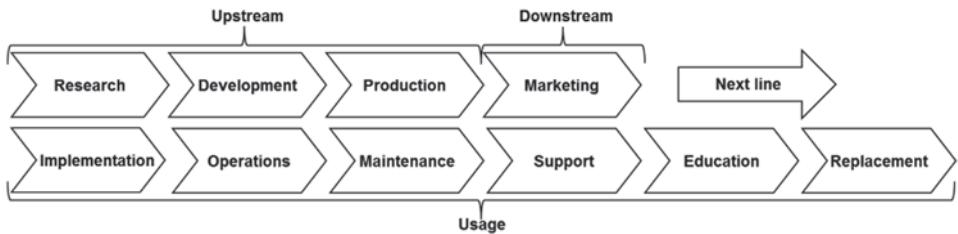


Abb. 4.1 Software value chain. (Pussep et al. 2011, S. 5)

4.1 Die Wertschöpfungskette in der Softwareindustrie

Das Konzept der Wertschöpfungskette wurde ursprünglich von Porter (1985) vorgestellt. Das Ziel dieses Konzepts besteht in erster Linie darin, Unternehmen ein Instrument zur Analyse und Verbesserung ihrer Wettbewerbsfähigkeit an die Hand zu geben. Darauf aufbauend können Strategien, wie etwa Kostenführerschaft oder Differenzierung, entwickelt werden (Porter 1985, S. 64).

Jedoch sind generische bzw. branchenunabhängige Wertschöpfungsketten in der Regel wenig pragmatisch (Stabell und Fjeldstad 1998, S. 419). Daher wurden industriespezifische Wertschöpfungsketten, beispielsweise für die „Mobile Commerce Value Chain“ (Barnes 2002), die Telekommunikations-Wertschöpfungskette (Li und Whalley 2002) oder das Wertschöpfungsnetzwerk der Grid-Industrie (Stanoevska-Slabeva et al. 2007) konzipiert.

4.1.1 Kernaktivitäten der Software-Wertschöpfungskette

Vor diesem Hintergrund wird in diesem Abschnitt ein Konzept für die Beschreibung der Wertschöpfungskette in der Softwareindustrie entwickelt. Nach Pussep et al. (2011, S. 4–5) lässt sich die Software-Wertschöpfungskette auf Basis von zehn Kernaktivitäten darstellen¹ (Abb. 4.1):

Research Hierbei handelt es sich um grundlegende in der Regel produktnahe Forschungsaktivitäten. Es geht bei diesen Aktivitäten vorrangig um die Nutzung innovativer Technologien. Häufig werden auch Prototypen entwickelt, um ein „Proof of Concept“ durchzuführen. Ebenso fällt die Entwicklung von neuen Algorithmen in diesen Bereich, wohingegen die Entwicklung von Code für das Produkt der Aktivität *Development* zugeordnet wird.

¹ Wir verzichten auf eine Übersetzung, weil sich die hier verwendeten englischen Begriffe auch im deutschen Sprachgebrauch der Softwareindustrie etabliert haben.

Development Das Development umfasst den Softwareentwicklungsprozess, wie er in diesem Buch auch in Abschn. 3.4 kurz behandelt wird. Zur Aktivität Development gehören neben den Methoden zur Entwicklung von Software, wie beispielsweise dem Wasserfallmodell oder SCRUM etc. (siehe hierzu auch Abschn. 3.4.1 dieses Buches), ebenfalls das Integrieren von Programmmodulen in ein lauffähiges Produkt, sowie dem Testen und der Dokumentation.

Production Im Rahmen dieser Aktivität werden Software und Dokumentation gebündelt und zu einem Produktpaket zusammengeschnürt.

Marketing Das Marketing umfasst alle Maßnahmen, die notwendig sind, um das Produkt am Markt zu platzieren und zu vermarkten. Hierzu gehören auch Preis- und Produktstrategien, wie wir sie etwa in Abschn. 3.3 und 3.4 dieses Buches diskutiert haben.

Implementation Die Implementierung umfasst alle erforderlichen Maßnahmen, um das Produkt in das Informationssystem des Kunden zu integrieren bzw. zum Laufen zu bringen. Hierzu gehören ebenfalls Anpassungen, wie das Customizing oder die Parametrisierung der Software (siehe hierzu Abschn. 4.2 des Buches).

Operations Im Rahmen dieser Aktivität wird der Betrieb der Softwarelösung gewährleistet. Hierzu gehört beispielsweise auch das Einspielen neuer Releases.

Maintenance Der Fokus dieser Aktivität liegt in der Erweiterung der Funktionalität bzw. in der Beseitigung von Fehlern eines existierenden Produkts. Bei den Systemänderungen handelt es sich um inkrementelle und nicht disruptive innovative Änderungen.

Support Diese Aktivität umfasst zum einen die Unterstützung von Nutzern bei einfachen Problemen oder Fehlern, zum anderen aber auch „development support“, wofür zum Teil tiefes technologisches Wissen und Kenntnisse des Softwarecodes notwendig sind.

Education Hierbei geht es um die Schulung von Nutzern, aber auch von Partnerfirmen, die komplementäre Softwarelösungen für das eigene Produkt herstellen und vermarkten.

Replacement Diese Aktivität analysiert die Frage, ob ein Produkt von einem neuen System, beispielsweise im Rahmen eines Upgrade, ersetzt werden soll. Aber auch die Entscheidung, ob etwa eine existierende On-Premise-Lösung durch eine On-Demand-Lösung abgelöst werden soll, fällt in diese Aktivität.

Pussep et al. (2011) führen darüber hinaus Expertengespräche mit Mitarbeitern von neun Softwareanbietern (im engeren und weiteren Sinne) durch. Ein interessantes Ergebnis hierbei ist die recht hohe Wertschöpfungstiefe, die zwischen 60 und 10% lag.



Abb. 4.2 Generische Wertschöpfungsstufen von Softwareherstellern. (in Anlehnung an Wolf et al. (2010))

4.1.2 Wertschöpfungsketten in der Softwareindustrie – 3 Fallstudien

Ebenfalls eine hohe Wertschöpfungstiefe in der Softwareindustrie zeigen Wolf et al. (2010) in mehreren Fallstudien auf. Die Autoren legen dabei eine Software-Wertschöpfungskette zugrunde, die sich von unserer nur marginal unterscheidet (so wird beispielsweise „Replacement“ nicht thematisiert und die Aktivität „Production“ wird mit „Dokumentation“ betitelt). Die Grundaussagen bleiben hierbei jedoch identisch. Im Folgenden wollen wir zentrale Ergebnisse von drei Fallstudien wiedergeben.

Ausgangspunkt dieser Studie ist eine achtstufige Wertschöpfungskette in der Softwareindustrie, wie sie in Abb. 4.2 dargestellt ist.

Bezugspunkt sind die Softwareanbieter. Für jede Wertschöpfungsstufe wurde untersucht, ob diese zu einem bestimmten Betrachtungszeitpunkt vom Softwareanbieter selbst übernommen, von Partnerunternehmen erbracht oder über den Markt bezogen wurde. Der Logik der Transaktionskostentheorie folgend (siehe Abschn. 2.4) wird die erste Koordinationsform als hierarchisch, die zweite als hybrid und die dritte als marktlich bezeichnet.

Zur Visualisierung greifen wir auf das Konzept der „Swimlanes“ (Binner 1987) zurück, das in vielen Prozessmodellierungs-Tools angeboten wird. Jede Wertschöpfungsstufe wird dabei entsprechend der in den Fallstudien ermittelten zwischenbetrieblichen Arbeitsteilung auf oder zwischen einer der „Swimlanes“ Hierarchie, Hybrid oder Markt platziert. Befindet sich eine Wertschöpfungsstufe ausschließlich auf der „Swimlane“ Hierarchie, werden sämtliche dazugehörige Teilaktivitäten rein vom jeweiligen Software-Anbieter selbst durchgeführt und es liegt keine zwischenbetriebliche Arbeitsteilung vor. Eine Platzierung auf der Begrenzung zwischen Hierarchie und Hybrid bedeutet hingegen einen höheren Grad an zwischenbetrieblicher Arbeitsteilung, da eine oder mehrere Teilaktivitäten in der Koordinationsform Hybrid und somit arbeitsteilig erbracht werden. Der höchste Grad an zwischenbetrieblicher Arbeitsteilung aus der Sicht des jeweiligen Softwareanbieters liegt – bezogen auf eine Wertschöpfungsstufe – vor, wenn sich diese vollständig auf einer der „Swimlanes“ Hybrid oder Markt oder auf der Grenze dazwischen befindet, da dann keine der jeweiligen Teilaktivitäten vom Fallstudienunternehmen selbst übernommen wird. Vergleicht man den ausgefüllten Untersuchungsrahmen zweier verschiedener Zeitpunkte, so lässt sich feststellen, ob und auf welchen Wertschöpfungsstufen es beim jeweiligen Fallstudienunternehmen zu einer veränderten Arbeitsteilung gekommen ist.

Fallstudie 1 (Wolf et al. 2010)

Die Fallstudie wurde im Mai 2009 bei einem eigenständigen, mittelständischen Standardsoftwarehersteller durchgeführt. Bei dem Produkt, dessen Wertschöpfungskette betrachtet wurde, handelt es sich um eine ERP-Lösung für kleine und mittelgroße Unternehmen, welche die Bereiche Warenwirtschaft und Produktion, Finanzbuchhaltung, Geschäftsanalyse, Personalmanagement und Kundenmanagement abdeckt und dabei helfen soll, die kaufmännischen und betriebswirtschaftlichen Anforderungen von Unternehmen aus den Bereichen Handel, Industrie sowie Dienstleistung zu lösen. Neben der aktuellen Situation wurde vom Unternehmen als zu betrachtender Zeitpunkt in der Vergangenheit das Jahr 2005 festgelegt.

Die *produktbezogene Forschung* wurde 2009 grundsätzlich von internen Mitarbeitern durchgeführt, wobei für bestimmte, den Funktionsumfang erweiternde Schnittstellenprodukte auch Kooperationspartner in die Entwicklung mit einbezogen wurden. Demgegenüber wurde die produktbezogene Forschung im Jahr 2005 ausschließlich intern durchgeführt.

Die *Produktentwicklung* des Kernprodukts wurde zu beiden Betrachtungszeitpunkten innerhalb des Unternehmens durchgeführt. Allerdings gab es 2009 die bei der produktbezogenen Forschung bereits erwähnten Schnittstellenprodukte, die den Leistungsumfang des Kernprodukts erweiterten und auch in Kooperation mit Partnern entwickelt wurden.

Die *Dokumentation* wurde sowohl 2005 als auch 2009 intern von spezialisierten Mitarbeitern aus dem Unternehmensbereich Qualitätsmanagement durchgeführt.

Die generischen *Marketingaktivitäten*, wie Messen, Anzeigen, Online-Medien, Branding oder Brand-Kampagnen, wurden sowohl 2009 als auch 2005 intern durchgeführt.

Der *Vertrieb* erfolgte sowohl 2005 als auch 2009 auf indirektem Weg (siehe Abschn. 3.2.1). Endkunden konnten das Produkt also nicht direkt vom Anbieter, sondern ausschließlich über teilweise auf bestimmte Produktlinien oder Branchen spezialisierte Vertriebspartner und Systemhäuser beziehen, mit denen eine langfristige Kooperationsbeziehung (teilweise bis zu 16 Jahren) bestand. Allerdings gab es 2009 im Gegensatz zu 2005 auch eigene Mitarbeiter des Fallstudienunternehmens im Bereich vertriebliche Beratung, welche die Partner auf Anfrage bei Kundenpräsentationen unterstützten, sodass 2009 neben hybrider auch hierarchische Koordination vorlag.

Die *Implementierung* wurde zu beiden Betrachtungszeitpunkten vollständig von den schon im Bereich vertriebliche Beratung genannten Kooperationspartnern durchgeführt, ohne dass diese – wie es 2009 auf der vorangegangenen Wertschöpfungsstufe der Fall war – von internen Mitarbeitern des Anbieters unterstützt wurden.

Grundsätzlich wurde die *Schulung* insbesondere bei Neukunden zu beiden Betrachtungszeitpunkten von den bereits im Bereich der vertrieblichen Beratung und der Implementierung tätigen Kooperationspartnern durchgeführt. Diese schulten die Key-User des Kunden im Umgang mit der Software, die dann wiederum

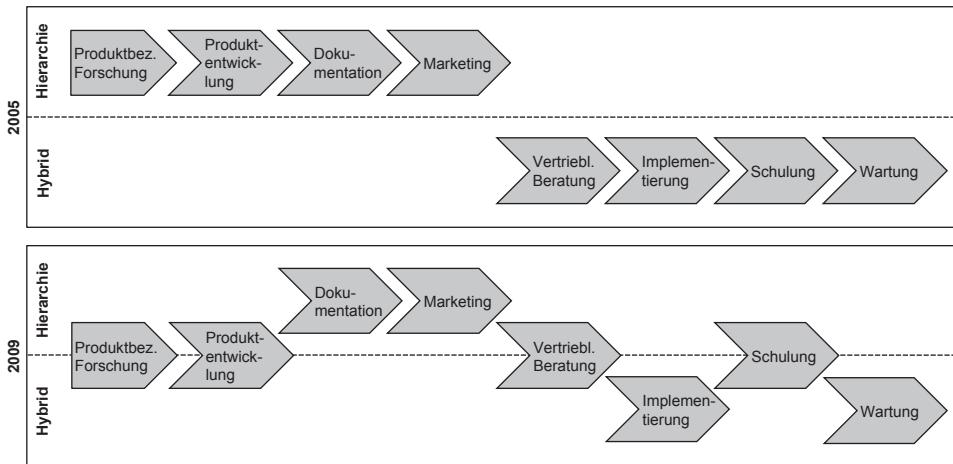


Abb. 4.3 Zwischenbetriebliche Arbeitsteilung der Fallstudie 1 im Vergleich. (Wolf et al. 2010)

intern weiterschulten. Lediglich bei bestimmten neuen Releases oder Änderungen des Softwareprodukts, z. B. im Bereich Finanzwesen oder Lohnbuchhaltung, die ein Partner gegebenenfalls aus inhaltlicher Sicht nicht leisten konnte, bot das Fallstudienunternehmen 2009 auch eigene Schulungen an, die es 2005 noch nicht gab.

Im Bereich des betrachteten Softwareprodukts wurde die *Wartung* sowohl 2005 als auch 2009 vollständig von den Kooperationspartnern des Fallstudienunternehmens übernommen.

Zusammenfassend ergeben sich für 2005 und 2009 die in Abb. 4.3 dargestellten Ausprägungen der zwischenbetrieblichen Arbeitsteilung.

Fallstudie 2 (Wolf et al. 2010)

Die Fallstudie wurde im Mai 2009 bei einem der führenden mittelständischen Beratungs- und Softwareunternehmen für integrierte ERP-Lösungen im deutschsprachigen Raum erhoben. Bei dem Produkt, dessen Wertschöpfungsstruktur betrachtet wurde, handelt es sich um ein modular aufgebautes, mehrsprachiges, mandanten- und mehrserverfähiges ERP-System für Kunden aus dem Bereich der Einzel-, Auftrags- und Variantenfertigung. Neben der aktuellen Situation wurde als zu betrachtender Zeitpunkt in der Vergangenheit das Jahr 2000 festgelegt.

Die *produktbezogene Forschung* und die *Produktentwicklung* wurden vom Fallstudienunternehmen nicht getrennt, sondern als eine integrierte Stufe betrachtet. Neue Produktideen und aktuelle Kundenbedürfnisse im Sinne der produktbezogenen Forschung wurden zu beiden Betrachtungszeitpunkten von eigenen Mitarbeitern ermittelt, festgesetzt und an die Entwicklungsabteilung gegeben. Innerhalb der Produktentwicklung grenzte das Fallstudienunternehmen ab zwischen der eigent-

lichen Anwendungsentwicklung und der technischen Entwicklung, die sich u. a. damit befasste, welche Tools bei der Softwareentwicklung eingesetzt werden. Die technische Entwicklung und die Entwicklung des Kernprodukts wurden 2000 vollständig von unternehmensinternen Mitarbeitern durchgeführt. 2009 existierten im Gegensatz zu 2000 außerdem verschiedene Möglichkeiten, externe Programme, wie z. B. Outlook, Excel oder Word, aus dem Softwareprodukt heraus anzusteuern oder bestimmte Basisfunktionalitäten der für das Softwareprodukt verwendeten SQL-Datenbank, wie z. B. grafische Darstellungen, zu nutzen, die aus Nutzersicht wie ein integraler Bestandteil aussahen. Da dies jedoch Standardfunktionen der für die Entwicklung verwendeten Microsoft-Technologien waren, können sie nicht als von externen Transaktionspartnern hinzugekaufte Programmmodulen im Sinne dieser Untersuchung verstanden werden, die auf die Koordinationsform Hybrid hindeuteten würden. Jedoch wurden die 2009 verfügbaren und in das System integrierbaren Module für Personalmanagement und Frachtpapiere/Zollabwicklung von externen Softwarepartnern erstellt, sodass neben der hierarchischen Koordination auf der Wertschöpfungsstufe *produktbezogene Forschung und Produktentwicklung* 2009 auch hybride Koordination vorlag.

Um das Fach-, Architektur-, Codier- und Dokumentationswissen nicht auf verschiedene Köpfe zu verteilen, wurde die *Dokumentation* zu beiden Betrachtungszeitpunkten von den im Bereich produktbezogene Forschung und Produktentwicklung tätigen internen Mitarbeitern durchgeführt.

2009 verfügte das Fallstudienunternehmen über eine eigene Marketingabteilung, welche die Pressearbeit und CRM-Aktivitäten durchführte sowie das von einer externen Agentur erstellte Markenbild am Markt kommunizierte. 2000 gab es diese Zusammenarbeit mit der externen Agentur noch nicht, sondern auch die später von der Agentur übernommenen Aktivitäten wurden noch intern durchgeführt. Somit lag auf der Wertschöpfungsstufe *Marketing* 2009 sowohl hierarchische als auch hybride Koordination, 2000 hingegen rein hierarchische Koordination vor.

2009 erfolgte der Vertrieb für den deutschen Markt auf direktem Weg, während für alle anderen Länder ein indirektes Vertriebsmodell mit jeweils wenigen Franchisenehmern implementiert wurde, an denen auch Minderheitsbeteiligungen bestanden. 2000 hingegen wurde – auf allen DACH-Märkten – noch viel stärker auf Partner gesetzt, was sich allerdings nicht bewährt hatte, da das Produkt oftmals nicht ordnungsgemäß verkauft und betreut wurde.

Die *Implementierung* wurde, wie schon die vertriebliche Beratung, 2009 auf dem deutschen Markt vom Anbieter selbst und auf internationalen Märkten von den bereits erwähnten Franchisenehmern erbracht. Auch 2000 wurde die Implementierung auf internationalen Märkten von Partnern und somit in der Koordinationsform Hybrid erbracht, allerdings führten Partnerunternehmen auch in Deutschland Implementierungsprojekte durch. Da die mit dem Softwareprodukt gelösten Probleme in der Regel sehr kundenspezifisch sind, existierte zu keinem der beiden Betrachtungs-

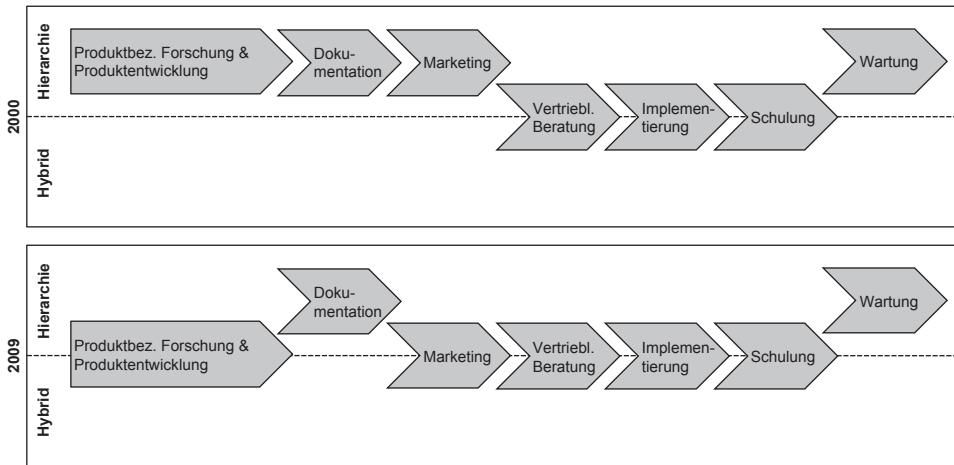


Abb. 4.4 Zwischenbetriebliche Arbeitsteilung der Fallstudie 2 im Vergleich. (Wolf et al. 2010)

zeitpunkte eine eigene Schulungsabteilung, die standardisierte Seminare oder ähnliches anbot.

Die *Schulung* wurde in der Regel jeweils von den die Implementierung durchführenden Mitarbeitern des Anbieters oder der Partnerunternehmen auf das jeweilige Unternehmen zugeschnitten und durchgeführt. Die Koordinationsformen stimmen daher für beide Betrachtungszeitpunkte mit denen bei der Wertschöpfungsstufe Implementierung beschriebenen überein.

Das Fallstudienunternehmen sah sich 2009 insbesondere bei beratungsintensiveren Projekten nicht nur als Lieferant des eigenen Softwareprodukts, sondern auch als Systemintegrator von Lösungen anderer Anbieter. Um die Kunden zufrieden zu stellen, wurde daher die *Wartung* nicht nur für die herstellereigenen Module, sondern auch für die zu Beginn des Kapitels erwähnten Module von verschiedenen Softwarepartnern, die in das System integriert werden konnten, sowie darüber hinaus für weitere, in der Rolle als Systemintegrator betreute Software intern durchgeführt. 2000 gab es die Wartung für die Module fremder Unternehmen in dieser Form noch nicht; die Wartung für die eigene Software wurde auch schon komplett intern durchgeführt.

Zusammenfassend ergeben sich für 2000 und 2009 die in Abb. 4.4 dargestellten Ausprägungen der zwischenbetrieblichen Arbeitsteilung.

Fallstudie 3 (Wolf et al. 2010)

Die Erhebung der Fallstudie fand im April 2009 statt. Das Produkt, dessen Wertschöpfungsstruktur betrachtet wurde, ist eine ERP-Software für Maschinen- und

Anlagenbauer sowie Automobilhersteller und -zulieferer, die sich aus einem ERP-Standard-Modul sowie verschiedenen Modulen für spezielle Probleme der genannten Zielbranchen zusammensetzt. Neben der aktuellen Situation wurde vom Unternehmen als Betrachtungszeitpunkt in der Vergangenheit der Zeitraum 1999–2001 festgelegt.

Der Kern des Softwareprodukts ist vor rund 25 Jahren aus einem Forschungsprojekt hervorgegangen. Damals hatte das Fallstudienunternehmen in seiner Rolle als Softwarehaus gemeinsam mit Kunden ein PPS-System speziell für Auftragsfertiger gesucht. Ein entsprechendes System war allerdings am Markt nicht vorhanden, sodass das Fallstudienunternehmen, die Kunden sowie Forschungseinrichtungen miteinander kooperiert haben, um ein eigenes System zu entwickeln. Diese Vorgehensweise führte das Fallstudienunternehmen bis auf den bereits vorhandenen Kern des Produkts auch 2009 bei produktbezogenem Forschungsbedarf durch, der sich rund um die Zielbranchen (Maschinen- und Anlagenbau; Automobilindustrie, insbesondere Zulieferer) ergab. Dabei wurden gemeinsam mit den Forschungseinrichtungen zusätzliche Lösungen entwickelt und in den Produktstandard integriert. Auf Grund der strategischen Positionierung des Fallstudienunternehmens war die einzige Voraussetzung dabei, dass es sich um etwas handelt, was direkt mit dem Bereich Fertigungsprozesse zu tun hat. Wenn es um Make-or-Buy-Entscheidungen bei der Zusammenstellung des Produktpportfolios ging, wurde grundsätzlich folgende Strategie verfolgt: Handelte es sich um ein Fertigungsthema, was die Position des Fallstudienunternehmens im Fertigungsumfeld stärken konnte, wurde die produktbezogene Forschung selbst durchgeführt. Dabei wurde entweder mit Kunden oder auch mit Forschungseinrichtungen kooperiert, wobei es sich jeweils um längerfristige partnerschaftlich angelegte Beziehungen mit einem ausgewählten Spektrum an Kunden bzw. Forschungseinrichtungen handelte, die sich auf Grund ihrer Branchenausrichtung oder lokalen Nähe anboten. 1999–2001 hingegen versuchte das Fallstudienunternehmen mit dem Softwareprodukt weniger den Spezialisierungsgedanken zu verfolgen, sondern einen deutlich breiteren Markt anzusprechen. Da die „Time to Market“ zu diesem Betrachtungszeitpunkt eine deutlich geringere war, führte das Fallstudienunternehmen die produktbezogene Forschung deutlich häufiger allein intern durch und musste sich stärker technischen Forschungsprojekten widmen. Hinsichtlich der idealtypischen Koordinationsformen lagen auf der Wertschöpfungsstufe *produktbezogene Forschung* demnach 2009 und auch 1999–2001 sowohl hierarchische als auch hybride Koordination vor.

Die *Produktentwicklung* der Kernfunktionalitäten für den Bereich Fertigung wurde 2009 hauptsächlich intern durchgeführt, wobei der Mutterkonzern des Fallstudienunternehmens bestimmte Entwicklungen nach Polen ausgelagert hatte. Diese polnische Einheit war ein hundertprozentiges Tochterunternehmen des Mutterkonzerns und arbeitete ihren Schwestergesellschaften zu. Aus Sicht des Fallstudien-

unternehmens handelte es sich bei der von der polnischen Schwestergesellschaft durchgeführten Produktentwicklung jedoch um hybride Koordination.

Des Weiteren wurden Funktionalitäten aus Randbereichen, in denen das Fallstudienunternehmen nicht tätig war (wie beispielsweise Dokumenten- oder Qualitätsmanagement), von ausgewählten Produktpartnern bezogen, die auf bestimmte Lösungsangebote spezialisiert waren. Diese Ergänzungen des Funktionsumfangs wurden über Schnittstellen integriert, wobei das Fallstudienunternehmen bei Kundenprojekten als Generalunternehmer auftrat. Somit lag 2009 bezüglich der Kernproduktentwicklung sowohl hierarchische als auch hybride (polnisches Schwesterunternehmen) Koordination und bezüglich der Entwicklung von Ergänzungen rein hybride (Produktpartner) Koordination vor. 1999–2001 erfolgte die Entwicklung des Kernprodukts hingegen ausschließlich intern. Allerdings wurden auch damals schon – sogar in einem größeren Maße als 2009 – ergänzende Funktionalitäten von Drittanbietern bezogen: Es gab deutlich mehr Drittanbieter als 2009 und die Partnerschaft zu diesen war weniger eng, sodass häufig zur Realisation des Pflichtenhefts von Endkunden eine bestimmte Software nur einige wenige Male hinzugekauft und integriert wurde, was der idealtypischen Koordinationsform Markt entspricht.

Deshalb ist für den Betrachtungszeitraum 1999–2001 wie folgt zu unterscheiden: Im Bereich der Kernproduktentwicklung lag ausschließlich hierarchische Koordination vor, im Bereich der Entwicklung ergänzender Funktionalitäten hingegen sowohl hybride als auch marktliche Koordination. Obwohl sich die Position der Teilaktivität „Ergänzungsentwicklung“ in unserer Darstellung daher von der Grenze zwischen den „Swimlanes“ Hybrid und Markt (1999–2001) auf die „Swimlane“ Hybrid (2009) verschiebt, bedeutet dies keine Änderung des Grades an zwischenbetrieblicher Arbeitsteilung aus Sicht des Fallstudienunternehmens. Dieses war nämlich weder 1999–2001 noch 2009 an der Teilaktivität „Ergänzungsentwicklung“ beteiligt.

Die *Dokumentation* wurde 2009 mittels Wikis von eigenen Mitarbeitern aus dem Unternehmensbereich Marketing und somit in der Koordinationsform Hierarchie durchgeführt. Dabei konnten auch die Dokumentationen der in das Kernprodukt integrierten Drittprodukte in das Wiki aufgenommen werden. Auch 1999–2001 wurde die Dokumentation schon vollständig intern durchgeführt, allerdings noch nicht auf Wiki-Basis, sodass die Integration von Drittdokumentationen insbesondere in die Online-Hilfe oftmals zu Problemen führte.

Die Aktivitäten der Wertschöpfungsstufe *Marketing* wurden vom Fallstudienunternehmen zu beiden Betrachtungszeitpunkten nahezu vollständig intern durchgeführt.

Der Hauptzugang zum Kunden war zum Betrachtungszeitpunkt 2009 ein direkter Vertriebskanal, der mit der Komplexität sowohl der Anforderungen der Endkunden als auch des Produkts selbst begründet wurde. Da das Fallstudienunternehmen

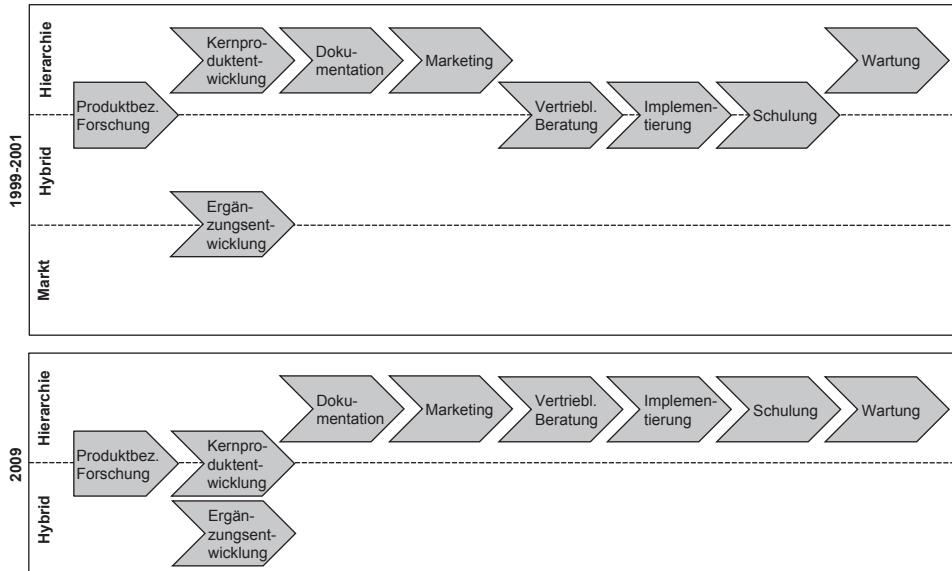


Abb. 4.5 Zwischenbetriebliche Arbeitsteilung der Fallstudie 3 im Vergleich. (Wolf et al. 2010)

1991–2001 einen breiteren Markt ansprechen wollte und dafür mehrere Vertriebskanäle benötigte, war neben dem direkten Vertrieb die indirekte Variante über Partnerunternehmen stark ausgeprägt. Zusammenfassend kann festgehalten werden, dass auf der Wertschöpfungsstufe *vertriebliche Beratung* 2009 die hierarchische Koordination dominierte. 1999–2001 hingegen lag neben hierarchischer in hohem Maße auch hybride Koordination vor.

2009 wurde die *Implementierung* im Kern vom Fallstudienunternehmen selbst durchgeführt. Ebenso wie bei der vertrieblichen Beratung spielten die Partnerunternehmen hingegen 1999–2001 auf Grund der gewollt breiteren Marktansprache eine größere Rolle, sodass auch hier sowohl hierarchische als auch hybride Koordination vorlag.

Die *Schulung* wurde 2009 als zusätzliche Aktivität im Rahmen der Implementierung beim Kunden vor Ort durchgeführt. 1999–2001 hingegen existierte auf Grund der deutlich breiteren Branchenausrichtung eine eigene Schulungsabteilung am Hauptsitz des Fallstudienunternehmens. Des Weiteren boten auch die in einer großen Anzahl vorhandenen Partnerunternehmen eigene Schulungen an, sodass sowohl hierarchische als auch hybride Koordination vorlag.

Das Fallstudienunternehmen sah die Aktivitäten der Wertschöpfungsstufe *Wartung* als wichtigste Erlösquelle von Standardsoftwareanbietern an. Diese wurden zu beiden Betrachtungszeitpunkten rein intern durchgeführt.

Zusammenfassend ergeben sich für 1999–2001 und 2009 die in Abb. 4.5 dargestellten Ausprägungen der zwischenbetrieblichen Arbeitsteilung.

	Produktbez. Forschung	Produkt-entwick- lung	Doku- mentation	Marketing	Vertriebl. Beratung	Implemen- tierung	Schulung	Wartung
Fallstudie 1: 2005 vs. 2009	↑	↑	✗	✗	↓	✗	↓	✗
Fallstudie 2: 2000 vs. 2009	↑	↑	✗	↑	✗	✗	✗	✗
Fallstudie 3: 99/01 vs. 2009	✗	↑ [*]	✗	✗	↓	↓	↓	✗

* Bezüglich Kernproduktentwicklung (Keine Veränderung bei "Ergänzungsentwicklung", siehe dazu Erläuterungen im Text auf S. 11)

Abb. 4.6 Fallstudienergebnisse im Überblick

In Abb. 4.6 ist überblicksartig dargestellt, in welchen der von uns erhobenen Fällen es auf welchen Wertschöpfungsstufen aus Sicht des Softwareanbieters zu einem erhöhten (↑) oder reduzierten (↓) Grad an zwischenbetrieblicher Arbeitsteilung gekommen ist bzw. wo sich keine Veränderung (✗) feststellen ließ.

Im Ergebnis zeigt sich eine zunehmende zwischenbetriebliche Arbeitsteilung vor allem auf den Upstream-Wertschöpfungsstufen produktbezogene Forschung (Fälle 1, 2) sowie Produktentwicklung (Fälle 1, 2, 3).

Dies ist insofern interessant, als die Wertschöpfungstiefe in reiferen Branchen, wie etwa der Automobilindustrie, deutlich niedriger liegt. Wir werden im folgenden Abschnitt darauf zurückkommen.

4.2 Geschäftsmodelle in der Softwareindustrie – ein Framework

Im Folgenden soll ein Framework für die Beschreibung und Analyse von Geschäftsmodellen vorgestellt werden, das auf der zuvor vorgestellten Software-Wertschöpfungskette aufbaut. Das im Folgenden vorgeschlagene Framework besteht aus insgesamt fünf Bausteinen (siehe Abb. 4.7).²

Der Baustein „Strategy“ beschreibt die strategischen Entscheidungen der Softwareanbieter, „Revenue“ beinhaltet u. a. die Ausgestaltung der Preismodelle sowie der daraus resultierenden Finanzflüsse. Sowohl „Upstream“ als auch „Downstream“ basiert auf den im letzten Abschnitt vorgestellten Überlegungen zur Wertschöpfungskette in der Softwareindustrie. Baustein „Usage“ beschreibt Komponenten, die wichtig für den Betrieb und die Wartung von Softwarelösungen sind.

Dabei besteht jeder dieser Bausteine wiederum aus verschiedenen Komponenten, die wir im Folgenden vorstellen. Für alle diese Komponenten existieren vorgegebene Parameterausprägungen, um eine standardisierte Beschreibung und Analyse der Geschäftsmodelle zu ermöglichen. Das Framework ist in der folgenden Abb. 4.8 dargestellt.

² Auf eine Übersetzung wurde aus den gleichen Gründen wie im letzten Abschnitt verzichtet.

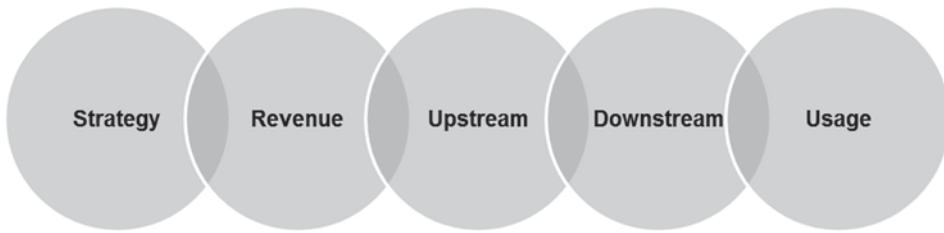


Abb. 4.7 Bausteine des software business model framework. (Schief 2014, S. 70)

Im Folgenden werden die Bausteine des Frameworks mit ihren Komponenten und Parameterausprägungen beschrieben.

Baustein 1: Strategy

Value Proposition: beschreibt alternative Optionen, wie Softwareanbieter Wettbewerbsvorteile erzielen können (Osterwalder und Pigneur 2010, S. 44). Hierzu gehören verschiedene Faktoren, wie etwa Image, Qualität sowie Funktionalität.

Weitere Möglichkeiten bieten sich beispielsweise durch die Innovation Leadership Position, die insbesondere durch neue und disruptive Innovationsstrategien erreicht wird. Die Intimate Customer Relationship stellt ebenfalls eine Chance dar und beschreibt die Beziehungsintensität zwischen einem Softwareanbieter und seinen Kunden. Andere mögliche Erfolgsfaktoren wie Design & Usability hängen von der Einfachheit der Nutzung, der Intuition und der Benutzeroberfläche ab. Darüber hinaus ermöglichen es One Stop Shops, dass Kunden die Komponenten einer Lösung (z. B. Software und Services) aus einer Hand auf einfache Weise erwerben können. Zuletzt gilt es als Option verschiedene Preisstrategien zu nennen, die bereits in Abschn. 3.3 dieses Buchs ausführlich behandelt wurden.

Investment Horizon: beschreibt den Zeithorizont des Geschäftsmodells. Die vorgeschlagenen Parameterausprägungen wurden von Morris et al. (2005, S. 730) entwickelt.

Value Chain: fasst die Hauptaktivitäten der Software Wertschöpfungskette zusammen, wie wir sie in Abschn. 4.1 vorgestellt haben.

Degree of Vertical Integration: beschreibt die Wertschöpfungstiefe. Sie misst den Anteil der Aktivitäten, die selbst erstellt werden. Prinzipiell kann jede Aktivität im Rahmen der Wertschöpfungskette selbst erledigt oder fremd vergeben werden. Die Ausprägungen *Low*, *Medium* und *High* repräsentieren eine qualitative Bewertung des Grades der Wertschöpfungstiefe.

Number of Cooperation Partners: Die Ausprägungen dieser Komponente basieren auf Arndt et al. (2008, S. 77–78) und sind selbsterklärend.

Baustein 2: Revenue

Sales Volume: beschreibt die Absatzmenge, die mit einem bestimmten Softwareprodukt erzielt wird. Zur Quantifizierung wird normalerweise die Anzahl der Implementierungen bei Kunden herangezogen. Die Parameterwerte *Low*, *Medium* und *High* stammen von Morris et al. (2005, S. 730) und repräsentieren eine qualitative Bewertung des Parameters.

Strategy										
Value Proposition	Image	Quality	Functionality		Innovation Leadership		Intimate Customer Relationship	Design & Usability		
Investment Horizon	Subsistence Model		Income Model		Growth Model		Speculative Model	Social Model		
Value Chain	Research	Development	Production	Marketing	Implementation	Operations	Maintenance	Support		
Degree of Vertical Integration	Low			Medium			High			
# of Cooperation Partners	None		One		Few		Many			
Revenue										
Sales Volume	Low			Medium			High			
Revenue Source	Direct			Advertising			Commission			
Pricing Assessment Base	Usage-based			Hybrid Combination			Usage-independent			
Payment Flow Structure	Upfront			Hybrid Combination			Recurring			
Revenue Distribution Model	Low			Medium			High			
Upstream										
Software Stack Layer	Application Software			Systems Software		Hardware Control & Embedded Software		(Web) Content		
Platform	Desktop Computers & Notebooks		Servers		Mobile		Cloud Computing	Embedded Systems		
License Model	Proprietary: Sell Usage Rights			Proprietary: Sell all Rights to Customers			Open Source: Copyleft Licenses (e.g. GPL)	Open Source: Permissive Licenses (e.g. BSD)		
Degree of Standardization	Individual Production			Batch Production			Bulk Production			
Key Cost Driver	Research & Development		Marketing & Sales		Services		Third Party Software Licenses	Hardware		
Downstream										
Localization	All		Local		EMEA (Europe, Middle East, Africa)	AMERICAS (North-, Central-, and South America)		APJ (Asia, Pacific, Japan)		
Target Customer	Small Organizations			Medium Organizations			Large Organizations			
Target Industry	All	Consumer	ICT	Manufacturing	Finance & Insurance	Wholesale & Retail	Services (e.g. Health)	Pharma & Chemicals		
Target User	Business - Broad Workforce		Business - Dedicated Specialists		Business - Managers		Consumers			
Channel	Sales Agents		Events		Telesales		Online Shops			
Usage										
Implementation Effort	Low			Medium			High			
Operating Model	On Premise			Hybrid Combination			On Demand			
Maintenance Model	Daily		Weekly		Monthly		Quarterly	Biyearly		
Support Model	Standard Support			Hybrid Combination			Customer Specific Support			
Replacement Strategy	One Release			Few Releases			Many Releases			

Abb. 4.8 Software business model framework. (Schief und Buxmann 2012, S. 3335)

Revenue Source: analysiert, welcher Akteur schließlich für die Nutzung der Softwarelösung bezahlt. Die Antwortoptionen wurden von Abdollahi und Leimstoll (2011, S. 3) entwickelt. „*Direct*“ bedeutet in diesem Kontext, dass der Nutzer für die Lösung bezahlt. *Advertising* steht für ein Modell, bei dem die Umsätze von „third parties“ und nicht von den Nutzern stammen, wie wir es beispielsweise von Anbietern wie Google kennen. Zuletzt bedeutet „*Commission*“, dass Firmen, wie beispielsweise eBay, Kommissionen erhalten, etwa in Form einer Umsatzbeteiligung am Transaktionsvolumen.

Pricing Assessment Base: gibt die Bestimmungsgrößen der Preismodelle wieder (siehe auch Abschn. 3.3.2 dieses Buches).

Payment Flow Structure: beschreibt, zu welchem Zeitpunkt sich Erlöse ergeben, also ob sie beispielsweise auf Basis von einmaligen oder laufenden Zahlungen entstehen. Wir haben diesen Punkt bereits in Abschn. 3.3.2.3 dieses Buches behandelt.

Revenue Distribution Model: beschreibt, inwieweit ein Softwareanbieter Umsätze oder Gewinne mit Drittanbietern teilt. Die Parameterausprägungen *Low*, *Medium* und *High* wurden von Bieger und Reinhold (2011, S. 49–52) entwickelt. Ein solches „*Revenue Sharing*“ findet beispielsweise häufig zwischen Anbietern von Apps und Betreibern von Plattformen statt (siehe hierzu Abschn. 6.3.1.2 dieses Buchs).

Baustein 3: Upstream

Software Stack Layer: Diese Klassifikation basiert auf einer Einteilung von Gao und Iyer (2006, S. 125) sowie Forward und Lethbridge (2008, S. 8–10).

Platform: Diese Komponente wird im 6. Kapitel dieses Buchs ausführlich diskutiert.

License Model: beschreibt die Lizenzbedingungen, unter denen Anwender die Softwarelösungen nutzen können. Verschiedene Lizenzmodelle werden in diesem Buch insbesondere in Abschn. 3.3.2.4 diskutiert.

Degree of Standardization: beschreibt den Standardisierungsgrad einer Softwarelösung. Die Parameterausprägungen wurden von Rajala und Westerlund (2007, S. 118–120) entwickelt. *Individual Production* beschreibt die Entwicklung von Individualsoftware, die sich an den spezifischen Anforderungen eines Kunden orientiert. Unter *Batch Production* wird dagegen verstanden, dass ein Anbieter eine Lösung für eine bestimmte Anzahl von Kunden wiederverwenden kann, während *Bulk Production* beschreibt, dass eine Lösung oder ein Modul an alle Kunden ausgeliefert wird, der Standardisierungsgrad also weiter zunimmt.

Key Cost Driver: analysiert die zentralen Kostentreiber eines Geschäftsmodells. Die Parameterausprägungen reichen von personalkostenintensiven Geschäftsmodellen bis hin zu Modellen, deren Aufwände insbesondere von Beschaffungs- und „Third Party“-Kosten bestimmt werden.

Baustein 4: Downstream

Localization: beschreibt die geographische Eingrenzung des Marktes. Hierbei steht *EMEA* für Europa, Middle East und Afrika, *AMERICAS* für Nord-, Zentral- und Süd-Amerika sowie *APJ* für Asien, Pazifik und Japan.

Target Customer: analysiert die Kundengröße. Die Parameterausprägungen stammen von Benlian und Hess (2010, S. 183–184), die – in Anlehnung an die Richtlinien der

Europäischen Union – Small Organizations als solche definieren, die maximal 50 Arbeitnehmer haben, Medium Organizations beschäftigen zwischen 51 und 250 und Large Organizations mehr als 250 Arbeitnehmer. Bei Private Individuals handelt es sich um Privatpersonen, die die Software kaufen.

Target Industry: beschreibt die Zielbranchen, die eine Softwarefirma adressiert. Die Ausprägungen wurden der Standard Industrial Classification (SIC 2013) entnommen. *All* bedeutet in diesem Zusammenhang, dass ein Anbieter horizontale Lösungen an Kunden aus verschiedenen Branchen vertreibt. *Consumer* beschreibt Lösungen für das B2C-Geschäft. Neun weitere Branchen (*Information and Communication Technology (ICT) Manufacturing, Finance & Insurance, Wholesale & Retail, Services (e.g. Health), Pharmaceuticals & Chemicals, Construction & Utilities, Transport & Storage, Public Sector*) wurden explizit als Zielmärkte aufgeführt. *Others* beschreibt alle weiteren nicht explizit aufgeführten Branchen.

Target User: beschreibt die Zielgruppe an Nutzern, für die eine Lösung entwickelt wurde. Die Parameterausprägungen wurden von Cotterman und Kumar (1989, S. 1316) vorgeschlagen. Für „Business Users“ wurde zwischen drei Typen unterschieden: *Broad Workforce, Dedicated Specialists* (z. B. Controlling), and *Managers* (z. B. Executive Information Systems oder Dashboards). Zudem nutzen *Consumers* die Software für Anwendungen, während *Software Developers* Software zur Entwicklung von Software verwenden.

Channel: beschreibt die Vertriebskanäle. Die Parameterausprägungen stammen von Osterwalder und Pigneur (2010, S. 44). *Sales Agents* verkaufen die Software direkt an ihre Kunden (siehe hierzu auch Abschn. 3.2 dieses Buchs). *Events* könnten genutzt werden, um größere Zielgruppen zu adressieren. *Telesales* erlaubt ebenfalls – in engeren Grenzen – den persönlichen Kontakt zu den Kunden. *Online Shops* beschreiben den Vertriebskanal über das Internet, während *Retail Stores* einen persönlichen Kundenkontakt ermöglichen.

Baustein 5: Usage

Implementation Effort: untersucht den Aufwand für die Implementierung und Anpassung von Softwarelösungen. Diese fallen insbesondere bei der Integration einer Softwarelösung in das Informationssystem der Kunden an.

Operating Model: systematisiert den Betrieb von Softwarelösungen. Dabei wird insbesondere zwischen *On Premise* und *On Demand* unterschieden (siehe hierzu Abschn. 7.3 dieses Buchs).

Maintenance Model: beschreibt die Häufigkeit neuer Releases. Die Parameterausprägungen *Daily, Weekly, Monthly, Quarterly, Biyearly* und *Yearly* wurden von Greer und Ruhe (2004, S. 244) entwickelt.

Support Model: untersucht, welche Art von Support die Kunden benötigen. *Standard Support* bedeutet ein „one size fits all“-Angebot. Demgegenüber bedeutet *Customer Specific Support* eine stärkere kundenindividuelle Differenzierung.

Replacement Strategy: beschreibt die Anzahl verfügbarer Produkte zu einem Zeitpunkt (Jansen et al. 2011, S. 2–4). *One Release* bedeutet, dass alle Kunden die gleiche Software-

version benutzen, während *Few Releases* bzw. *Many Releases* die Anzahl der den Kunden zur Verfügung gestellten Releases einer Softwarelösung bezeichnet.

Das Framework lässt sich grundsätzlich auf alle Bereiche der Softwareindustrie übertragen. Im Folgenden wollen wir die Anwendung am Beispiel der mobilen Betriebssysteme iOS von Apple sowie Google's Android exemplarisch darstellen. Tabelle 4.1 gibt eine

Tab. 4.1 Software business models: Apple iOS vs. Google Android. (Scholz 2012, S. 43)

		Mobile operating systems	
		Apple – iOS	Google – Android
Strategy	Value proposition	Design & usability	Price
	Investment horizon	Growth model	Cross finance model
	Value chain	Development	Development
	Degree of vertical integration	High	Medium
	# of cooperation partners	Many	Many
Revenue	Sales volume	High	High
	Revenue source	Direct	Advertising
	Pricing assessment base	Usage-independent	Hybrid combination
	Payment flow structure	Upfront	Hybrid combination
	Revenue distribution model	Low	Low
Upstream	Software stack layer	Systems software	Systems software
	Platform	Mobile	Mobile
	License model	Proprietary: sell usage rights	Open source: permissive licenses
	Degree of standardisazion	Bulk production	Bulk production
	Key cost driver	Research & development	Research & development
Downstream	Localization	All	All
	Target customer	Private individuals	Private individuals
	Target industry	All	All
	Target user	Consumers	Consumers
	Channel	Retail stores	Retail stores
Usage	Implementation effort	Low	Low
	Operating model	On premise	On premise
	Maintenance model	Biyearly	Biyearly
	Support model	Standard support	Standard support
	Replacement strategy	Few releases	Few releases

Zusammenfassung des Vergleichs wieder und verdeutlicht die wesentlichen Unterschiede der Betriebssysteme. Während eine wesentliche Strategie von Apple darin besteht, Wettbewerbsvorteile durch Differenzierung zu erlangen, insbesondere im Hinblick auf Anwendbarkeit und Design, verfolgt Google im Segment der mobilen Betriebssysteme die Strategie der Preis- bzw. Kostenführerschaft, indem Android kostenlos vertrieben wird. Die Überlegung dahinter ist bekannt: es sollen möglichst viele Nutzer gewonnen werden, um letztlich höhere Werbeeinnahmen zu generieren. Des Weiteren trägt die Strategie offener Schnittstellen auch zu einer Erweiterung des Ökosystems bei (siehe hierzu Kap. 8 dieses Buchs). Somit steigt die Anzahl der Teilnehmer, während die Wertschöpfungstiefe sinkt. Im Gegensatz dazu verkauft Apple sein Betriebssystem den Kunden über gebündelte Produktverkäufe (etwa als Teil von iPhone oder iPad). Apple verfolgt damit eher eine proprietäre Produktstrategie.

4.3 Software Business Model Tool

In diesem Abschnitt stellen wir nun ein Tool zur Analyse von Geschäftsmodellen in der Softwareindustrie vor. Es basiert auf dem zuvor vorgestellten Framework. Hierbei können wir die folgenden beiden Anwendungsmöglichkeiten bzw. -szenarien unterscheiden:

- **Nutzer entwickeln und analysieren ihr eigenes Geschäftsmodell:** Diese Anwendungsmöglichkeit kann sowohl für etablierte Unternehmen als auch für Startups von Interesse sein. Ein Geschäftsmodell kann dabei entweder „from scratch“ entwickelt oder ein existierendes Modell kann analysiert werden.
- **Nutzer können Geschäftsmodelle in der Softwareindustrie allgemein analysieren:** Wir gehen davon aus, dass verschiedene Stakeholder Interesse an der Performance von Softwareunternehmen haben. Hierzu gehören u. a. Analysten. So können z. B. verschiedene Lizenzmodelle und ihre Auswirkungen auf die Performance untersucht werden (Weise et al. 2011, S. 30–34). Weitere Stakeholder sind beispielsweise Kunden, Journalisten, Berater, Politiker oder Studierende.

Im Folgenden sollen nun die Architektur sowie die Anwendungsmöglichkeiten beschrieben werden. Diese umfassen fünf Ebenen. Die oberste Ebene ermöglicht hierbei die Konfiguration des Geschäftsmodells, d. h. die Festlegung der Ausprägungen der Modellparameter unseres Frameworks, den Vergleich mit anderen Geschäftsmodellen sowie die Analyse.

Die erste Softwarekomponente ist die **Business Modeling Environment**. Diese unterstützt eine schrittweise Konfiguration der Geschäftsmodelle, einschließlich einer Visualisierung der Konfiguration. Abbildung 4.9 zeigt die Konfiguration am Beispiel des „Software Stack Layer“. Eine entsprechende Unterstützung der Konfiguration wird für alle 25 Parameter des Frameworks angeboten.



Example: Software designed to help the user to perform specific **applications**, e.g. ERP, accounting, office, media, games; **systems software** (Software designed to integrate information systems, e.g. operating systems, middleware, engineering, security, servers), **embedded software** (e.g. hardware control, firmware), or **(web) content** (e.g. legal information or IBAN banking numbers).

Abb. 4.9 Business modeling environment



Abb. 4.10 Detaillierte Übersicht zu business modeling environment

Nach der Eingabe der Parameter zur Spezifikation des Geschäftsmodells kann der Nutzer sich die gesamte Konfiguration auf einen Blick anschauen. Abbildung 4.10 stellt ein Beispiel dar.

Die Komponente **Benchmarking Environment** unterstützt die Analyse von ähnlichen Geschäftsmodellen auf Basis eines Cluster-Algorithmus (Chakrabarti 2002, S. 84). Grundlage hierfür sind die Daten aus dem German Software Industry Survey, den wir in den Jahren 2012 und 2013 erhoben haben (Pussep et al. 2012a, 2013).

Die Resultate werden in vier Grafiken bzw. Tabellen dargestellt (siehe Abb. 4.11). Die Tabelle links oben stellt zehn Geschäftsmodelle dar, die dem konfigurierten Modell am ähnlichsten sind. Für jedes dieser Modelle wird der Grad der Ähnlichkeit, die Umsatzrendite sowie das Wachstum im Vergleich zu den Wettbewerbern angezeigt. Die Grafik rechts

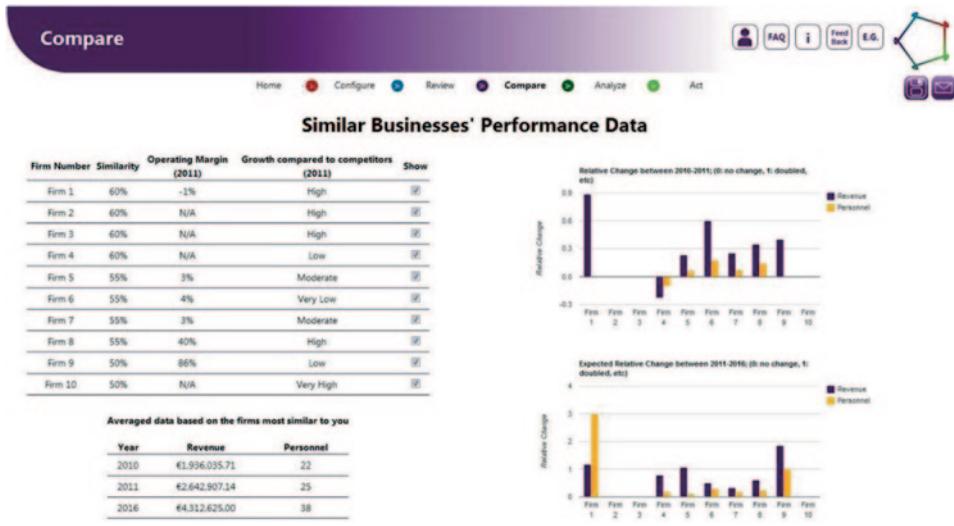


Abb. 4.11 Benchmarking environment

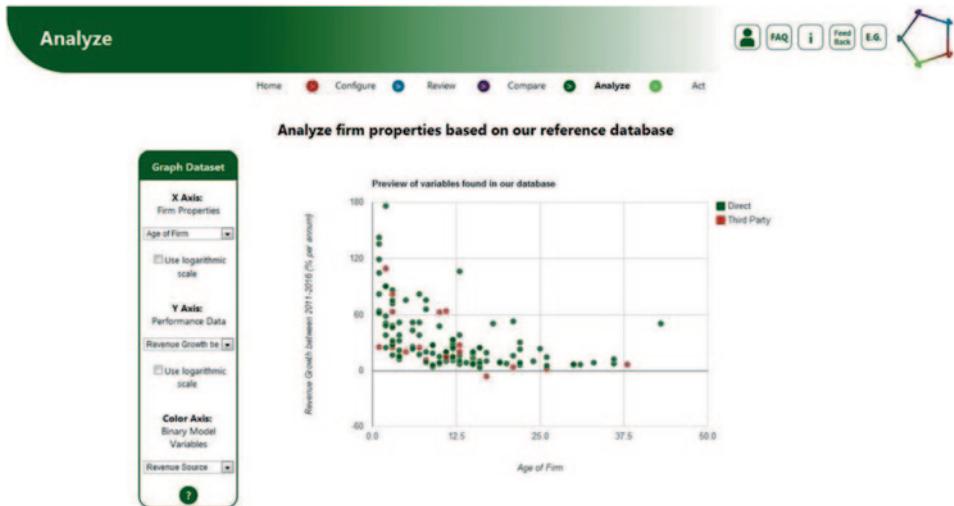


Abb. 4.12 Business intelligence & analytics

oben stellt die relative Änderungen in Bezug auf Umsatz und Mitarbeiteranzahl von 2010 bis 2011 dar. Die dritte Grafik (rechts unten) zeigt die prognostizierten Änderungen von Umsatz und Personal von 2011 zu 2015. Die Tabelle links unten bildet die durchschnittlichen absoluten Werte der ähnlichsten Geschäftsmodelle ab.

Die dritte Komponente **Business Intelligence & Analytics** unterstützt Analysen auf der Basis von drei Datenquellen (Geschäftsmodelle, Firmencharakteristika sowie Finanzperformancedaten).

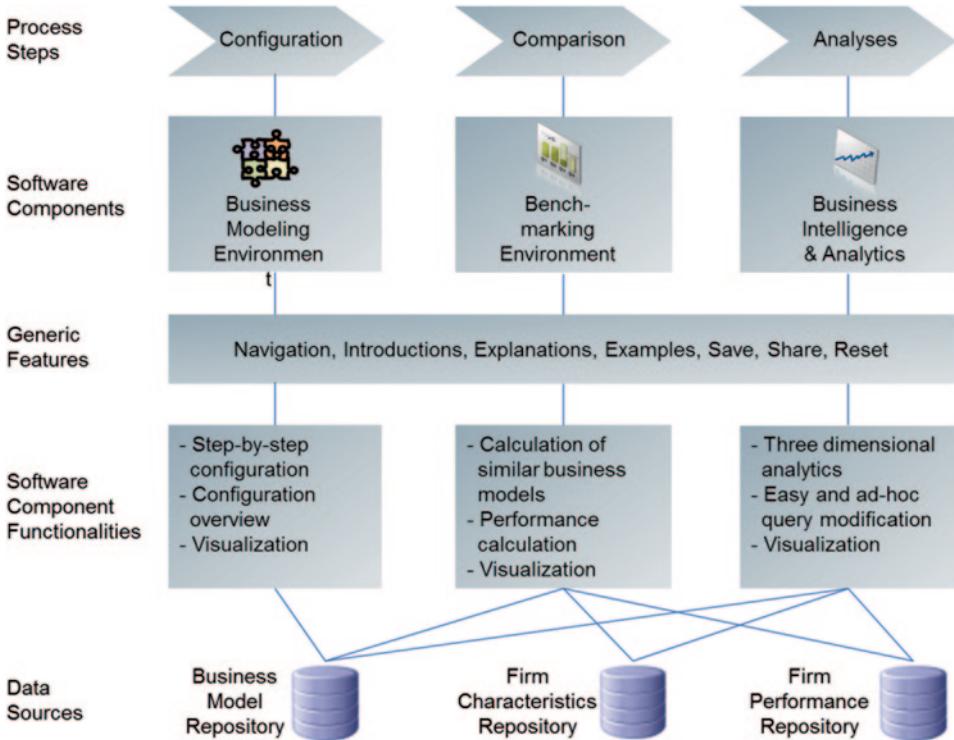


Abb. 4.13 Architecture of software business model tool. (Schief 2014, S. 174)

Das Tool bietet den Anwendern dabei mehrdimensionale Analysemöglichkeiten an: Nutzer können beispielsweise bestimmte Unternehmenscharakteristika auswählen, wie das Firmenalter (X-Achse). Daneben können ausgewählte Parameter zur Messung der Unternehmens-Performance abgebildet werden (Y-Achse). Die dritte Dimension repräsentiert bestimmte Parameterausprägungen des Geschäftsmodells (z. B. die Umsatzquelle).

Neben diesen Komponenten wurden einige weitere Features implementiert, auf die wir hier nicht weiter im Detail eingehen. Die Gesamtarchitektur des Tools ist in Abb. 4.13 dargestellt.

Das Tool steht online unter www.software-business-model.com auf einem Server der Technischen Universität Darmstadt zur Verfügung.³ Es wurde in HTML₅ unter Nutzung von CSS und JavaScript entwickelt.

Die Web-Seite ging am 1. September 2012 live. Innerhalb des ersten Jahres haben 5.464 Nutzer aus 87 Ländern die Seite besucht. Die meisten Besucher kamen dabei aus

³ Das Tool wurde in einer Kooperation zwischen dem Lehrstuhl für Wirtschaftsinformatik | Software Business & Information Management und der SAP AG im Rahmen des Software-Clusters entwickelt. Beteiligt waren insbesondere Markus Schief, Anton Pussep und Matthew Grey. Wir danken dem BmbF für die Unterstützung.

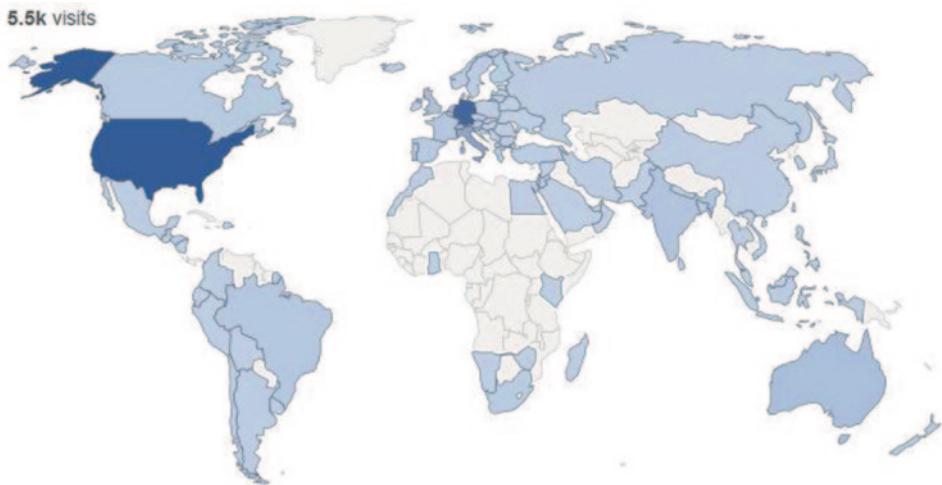


Abb. 4.14 Visitor map of software business model tool. (Piwik 2013)

den USA (1522 visits), Deutschland (1339 visits), Italien (468 visits), Australien (165 visits), und Großbritannien (155 visits).

Eine spannende Fragestellung ist, ob und inwieweit die Ausprägungen der Parameter des Geschäftsmodell-Frameworks den Erfolg der Softwarefirmen beeinflussen. Hierbei wurden – neben dem bereits erwähnten German Software Industry Survey – auch Sekundärdaten analysiert. Dabei handelt es sich zum einen um die von PricewaterhouseCoopers zur Verfügung gestellten Daten über die Global Top 100 Softwareanbieter (PWC 2010, S. 6–7). Zum anderen wurden Daten aus der Dijk Orbis database (Dijk 2013) verwendet, ergänzt um Informationen aus der Thomson Financial Worldscope Database. Das zweite Sample enthält die 120 US-amerikanischen Softwareanbieter mit den höchsten Umsätzen.

Für beide Stichproben führt Schief umfangreiche Analysen durch, um den Einfluss der Modellparameter sowohl auf die Firmen als auch auf die Markt-Performance zu untersuchen (Schief 2014). Im Ergebnis kann für die meisten Modellparameter ein nicht ganz einheitliches Bild festgehalten werden, was sicherlich auch daran liegt, dass weitere Faktoren, die nicht Bestandteil des Geschäftsmodell-Frameworks sind, einen erheblichen Einfluss auf die Firmen- und Markt-Performance haben können. Hierzu gehören etwa die Qualität des Managements sowie der Mitarbeiterinnen und Mitarbeiter, aber auch Intellectual Property und andere „Intangible Assets“.

Interessant ist jedoch, dass die Analyse beider Stichproben einen statistisch signifikanten Zusammenhang zwischen der Wertschöpfungstiefe einerseits und sowohl der Firmen- als auch Markt-Performance andererseits aufzeigt. Das heißt, dass sich eine höhere Wertschöpfungstiefe in der Softwareindustrie positiv auf die Finanz- und Markt-Performance auswirkt.

Dieses Ergebnis ist insofern überraschend, als in vielen anderen Branchen – wie etwa der Automobilindustrie – eine sinkende Wertschöpfungstiefe im Sinne einer stärkeren Auslagerung an Zulieferer häufig als Erfolgsrezept gefeiert wird. Häufig wird sogar ge-

fordert, dass sich die Softwareindustrie an diesen reiferen Branchen orientieren und ihrerseits die Wertschöpfungstiefe reduzieren sollte. Diese Forderungen haben jedoch häufig eher einen überschaubaren Anteil an gehaltvoller Information, da keine Aussagen darüber gemacht werden, ob und wenn, aus welchen Gründen Konzepte aus den so genannten reiferen Industrien auf digitale Branchen, wie die Softwareindustrie, übertragen werden können.

Woran könnte es liegen, dass eine hohe Wertschöpfungstiefe in der Softwareindustrie scheinbar vorteilhaft erscheint? Eine mögliche theoretische Begründung kann über die in Abschn. 2.4 dargestellte Transaktionskostentheorie und den Grad der Standardisierung erfolgen. Demnach haben der Grad der Spezifität bzw. der Grad der Standardisierung einen erheblichen Einfluss auf die Höhe der Transaktionskosten, die wiederum bestimmt, ob Fremdbezug oder Eigenfertigung der bessere und effizientere Koordinationsmechanismus ist. In der Softwareindustrie haben wir es relativ häufig mit spezifischen Modulen oder Funktionalitäten zu tun, während etwa in der Automobilindustrie die Komponenten einen höheren Standardisierungsgrad aufweisen und damit unspezifischer sind. Für Softwareunternehmen ist es vor diesem Hintergrund folglich wichtig, eine Auslagerung von Aktivitäten im Rahmen ihrer Wertschöpfungskette sorgfältig zu analysieren. Dies gilt sowohl für den Upstream- als auch den Downstream-Bereich. Aufgrund der Eigenschaften von Softwareprodukten und -märkten ist eine unkritische Übernahme von Konzepten aus den so genannten reiferen Industrien nicht anzuraten.

Literatur

- Abdollahi G, Leimstoll U (2011) A classification for business model types in e-commerce. In: Proceedings of the 17th American conference on information systems (AMCIS), Detroit, S 1–14
- Amit R, Zott C (2012) Creating value through business model innovation. MIT Sloan Manage Rev 53:41–49
- Arndt J-M, Kude T, Dibbern J (2008) The emergence of partnership networks in the enterprise application development industry: a global corporation perspective. In: Avison D, Kasper GM, Pernici B, Ramos I, Roode D (Hrsg) Advances in information systems research, education and practice. Springer, Boston, S 77–88
- Barnes SJ (2002) The mobile commerce value chain: analysis and future developments. Int J Inf Manage 22:91–108
- Benlian A, Hess T (2010) Chancen und Risiken des Einsatzes von SaaS—Die Sicht der Anwender. In: Benlian A, Hess T, Buxmann P (Hrsg) Software-as-a-service. Gabler, Wiesbaden, S 173–187
- Bieger T, Reinhold S (2011) Das wertbasierte Geschäftsmodell – Ein aktualisierter Strukturierungsansatz. In: Bieger T, zu Knyphausen-Aufseß D, Krys C (Hrsg) Innovative Geschäftsmodelle. Springer, Heidelberg, S 13–70
- Binner HF (1987) Anforderungsgerechte Datenermittlung für Fertigungssteuerungssysteme. Beuth, Köln
- Chakrabarti S (2002) Mining the web: discovering knowledge from hypertext data. Morgan Kaufmann, San Francisco
- Cotterman WW, Kumar K (1989) User cube: a taxonomy of end users. Commun ACM 32:1313–1320

- Dijk BV (2013) Orbis database. <http://orbis.bvdinfo.com/version-201345/home.serv?product=orbisneo>. Zugegriffen: 20. Feb. 2012
- Forward A, Lethbridge TC (2008) A taxonomy of software types to facilitate search and evidence-based software engineering. In: Proceedings of the 18th conference of the center for advanced studies on collaborative research (CASCON) ACM, S 1–13
- Gao L, Iyer B (2006) Analyzing complementarities using software stacks for software industry acquisitions. *J Manage Inf Syst* 23:119–147
- Greer D, Ruhe G (2004) Software release planning: an evolutionary and iterative approach. *Inf Softw Technol* 46:243–253
- Jansen S, Popp KM, Buxmann P (2011) The sun also sets: ending the life of a software product. In: Proceedings of the 2nd international conference on software business (ICSOB) lecture notes in business information processing, Boston, S 154–167
- Li F, Whalley J (2002) Deconstruction of the telecommunications industry: from value chains to value networks. *Telecommun Policy* 26:451–472
- Morris M, Schindehutte M, Allen J (2005) The entrepreneur's business model: toward a unified perspective. *J Bus Res* 58:726–735
- Osterwalder A, Pigneur Y (2010) Business model generation: a handbook for visionaries, game changers, and challengers. Wiley, Hoboken
- Osterwalder A, Pigneur Y, Tucci CL (2005) Clarifying business models: origins, present, and future of the concept. *Commun Assoc Inf Syst* 16:1–25
- Piwik (2013) Web analytics reports. <https://www.softwareindustrysurvey.de/mpwk/index.php>. Zugegriffen: 6. April 2013
- Porter ME (1985) Competitive advantage: creating and sustaining superior performance. Simon & Schuster, New York
- Pussep A, Schief M, Widjaja T, Buxmann P, Wolf CM (2011) The software value chain as an analytical framework for the software industry and its exemplary application for vertical integration measurement. In: Proceedings of the 17th Americas conference on information systems (AMCIS), Detroit, S 1–8
- Pussep A, Schief M, Schmidt B, Friedrichs F, Buxmann P (2012a) Topics in software industry transformation research: a topic analysis of major is conferences. In: Proceedings of the 3rd international conference on software business (ICSOB) Lecture Notes in Business Information Processing Boston, S 128–140
- Pussep A, Schief M, Weiblen T, Leimbach T, Peltonen J, Rönkkö M, Buxmann P (2013) Results of the German software industry survey 2013.
- PWC (2010) Global 100 software leaders. PricewaterhouseCoopers, S 1–63
- Rajala R, Westerlund M (2007) Business models – a new perspective on firms' assets and capabilities. *Entrep Innov* 8:115–125
- Schief M (2014) Business models in the software industry: the impact on firm and M&A performance. Springer, Wiesbaden.
- Schief M, Buxmann P (2012) Business models in the software industry. In: Proceedings of the 45th Hawaii international conference on system sciences (HICSS) IEEE, Wailea, S 3328–3337
- Scholz M (2012) Kampf der Geschäftsmodelle – Apple vs. Google. *Information Systems*. Technische Universität, Darmstadt, S. 1–90
- SIC (2013) Standard industrial classification. http://www.osha.gov/pls/imis/sic_manual.html. Zugegriffen: 6. April 2013
- Stabell CB, Fjeldstad ØD (1998) Configuring value for competitive advantage: on chains, shops, and networks. *Strategic Manage J* 19:413–437
- Stanoevska-Slabeva K, Talamanca CF, Thanos GA, Zsigri C (2007) Development of a generic value chain for the grid industry. In: Proceedings of the 4th conference on economics of grids, clouds, systems, and services (GECON) Lecture Notes in Computer Science, Rennes, S 44–57

- Veit DJ, Clemons EK, Benlian A, Buxmann P, Hess T, Kundisch D, Leimeister JM, Loos P, Spann M (2014) Business Models – an information systems research agenda. *BISE* 6:45–53
- Weise F, Lasch C, König M (2011) Von der Forschung zum Erfolg: Die Strategische Geschäftsmodellwahl bei Software, S 1–48
- Wolf CM, Benlian A, Hess T (2010) Industrialisierung von Softwareunternehmen durch Arbeitsteilung: Einzelfall oder Trend? Multikonferenz Wirtschaftsinformatik 2010.

Teil II

Spezielle Themen

5.1 Überblick

Kaum eine Branche ist internationaler als die Softwareindustrie. Diese Eigenschaft von Softwaremärkten kann hierbei auch auf die Eigenschaften des Gutes Software zurückgeführt werden. Software kann überall entwickelt und in Sekundenschnelle über das Internet verschickt werden. Im Gegensatz zu einer physischen Supply Chain gehen die entsprechenden Kosten für den Transport der Produkte oder Komponenten gegen Null. Dies eröffnet für Anbieter die Chance, Software in Projekten weltweit zu entwickeln. Damit können zum einen Lohnkosten eingespart, zum anderen aber auch weltweit hervorragend ausgebildete Mitarbeiter akquiriert werden. Die gleichen potenziellen Vorteile existieren natürlich auch für Anwendungsunternehmen, die Entwicklungsaufgaben an Softwarehäuser fremd vergeben.

Aber die Globalisierung der Softwareindustrie ist nicht nur auf den Beschaffungs- bzw. Arbeitsmärkten von Bedeutung. Vielmehr lässt sich Software auch einfach über das Internet vertreiben. Daher ist es wenig verwunderlich, dass alle größeren Anbieter weltweit organisiert sind. Die Internationalität der Märkte spiegelt sich u. a. darin wider, dass es für die Softwareindustrie kaum noch „Heimatmärkte“ gibt – im Gegensatz zu den meisten anderen Branchen, in denen Unternehmen in ihren Heimatländern häufig einen signifikant höheren Umsatz als die Konkurrenz erzielen.

Darüber hinaus findet ein Wettbewerb um Standorte statt. Dies gilt nicht nur in Asien, wo mehrere Softwareentwicklungscentren miteinander in Konkurrenz stehen, sondern auch in Europa und den USA. Die wichtigste Ressource von Softwareunternehmen sind die Mitarbeiter; das hergestellte Produkt ist digital. Damit gilt auch, dass „kein Unternehmen so leicht verlagerbar ist wie ein Softwareunternehmen“, wie Dietmar Hopp, Mitgründer der SAP AG, auch anlässlich der Debatte um die Etablierung eines Betriebsrats bei der SAP kommentierte.

In seinem mehrfach ausgezeichneten Buch „The world is flat“ beschreibt der Journalist Thomas Friedman die Internationalisierung der Märkte und identifiziert hierbei so genannte Flatteners – also Faktoren, die die Globalisierung der Welt vorantreiben (Friedman 2005). Wir wollen vor diesem Hintergrund im Weiteren auf die Softwareentwicklung offshore und die sich daraus ergebenden Chancen und Herausforderungen näher eingehen.

Zunächst geben wir einige grundlegende Definitionen und stellen unterschiedliche Varianten von Outsourcing und Offshoring vor. Darauf aufbauend betrachten wir verschiedene Organisationsformen und analysieren Fragestellungen im Zusammenhang mit der Standortwahl von IT-Service-Anbietern. Eine Diskussion möglicher Treiber und Motive für das Outsourcing und Offshoring bildet letztlich die Grundlage für die Analyse der Erfolgsfaktoren dieser Projekte. Der Schwerpunkt liegt hierbei auf der Untersuchung der Bedeutung der räumlichen Distanz zwischen Kunde und Anbieter für den Erfolg des Projekts. Die Ausführungen basieren zum einen auf einer großzahligen empirischen Untersuchung, die wir unter fast 500 CIOs durchgeführt haben. Zum anderen wurden Experteninterviews mit führenden Mitarbeitern aus zehn Softwarehäusern durchgeführt. Dabei handelt es sich um die folgenden Unternehmen:

- Accenture,
- CGI,
- Cognizant,
- Covansys,
- Gamaxcom,
- HCL Technologies,
- Infosys Technologies,
- Starsoft Development Labs,
- Tatvasoft und
- Wipro Technologies.

Abgerundet wurden die empirischen Studien durch 23 Experteninterviews unter Anwendungsunternehmen.

5.2 Formen des Outsourcings und Offshorings

Definieren wir im Folgenden zunächst einige Grundbegriffe: Unter Outsourcing wird allgemein die Vergabe von Aufgaben oder Dienstleistungen – in unserem Kontext insbesondere IT-Services – an Dienstanbieter, die nicht zum Konzernverbund des auslagernden Unternehmens gehören, verstanden (Mertens et al. 2005). Sitzt dieser Anbieter im Ausland, handelt es sich um Offshoring, andernfalls um Onshoring. Wird ein verbundenes Unternehmen im Ausland beauftragt, sprechen wir von Captive Offshoring. Die Begriffe sind in Abb. 5.1 dargestellt.

		Auftragsempfänger hat seinen Sitz im...	
		Inland	Ausland
Verlagerung von internen Aktivitäten an...	verbundenes Unternehmen	–	Captive Offshoring
	fremdes Unternehmen	Onshoring und Outsourcing	Offshoring und Outsourcing

Abb. 5.1 Outsourcing und Offshoring. (in Anlehnung an Mertens et al. 2005, S. 2)

Bei der Entscheidung, ob bestimmte Aufgaben, wie etwa die Softwareentwicklung, an Dritte auszulagern sind, handelt es sich um das klassische Make-or-Buy-Problem. Zur Lösung dieses Problems kann die Transaktionskostentheorie herangezogen werden. Hierauf sind wir bereits in Abschn. 2.4 eingegangen.

In Abb. 5.1 wurde lediglich zwischen Offshoring und Captive Offshoring unterschieden. Zwischen diesen beiden Organisationsformen gibt es jedoch eine Reihe von Abstufungen, die in Abb. 5.2 in Abhängigkeit von der Bindungsintensität dargestellt sind.

Die Gründung von Tochterunternehmen an einem Offshore-Standort kann für ein Unternehmen zahlreiche Vorteile mit sich bringen. Neben den grundsätzlich beim Offshoring verfolgten Zielen der Kosteneinsparung sowie dem Zugang zu hochqualifizierten Arbeitskräften kann das Captive Offshoring die Erschließung neuer Märkte begünstigen. Im Vergleich zu einer Vergabe an Fremdunternehmen ermöglicht das Captive Offshoring in der Regel ein hohes Maß an Kontrolle – und zwar nicht nur über Prozesse, sondern auch über das Know-how des Unternehmens (Gadatsch 2006, S. 49).

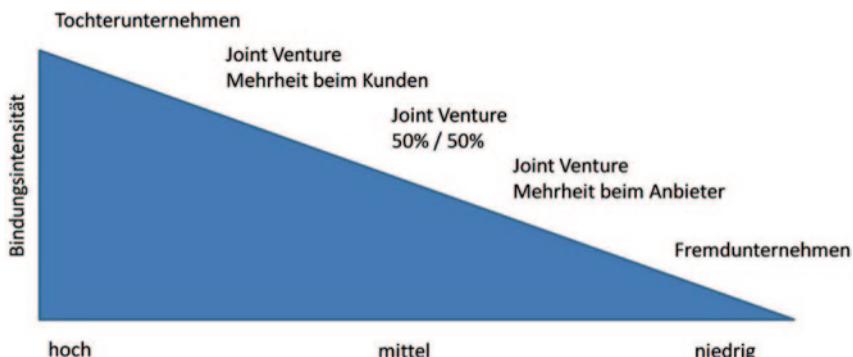


Abb. 5.2 Bindungsintensität der Organisationsformen. (in Anlehnung an Bräutigam und Grabbe 2004, S. 179)

Allerdings sind beim Captive Offshoring auch eine Reihe von Herausforderungen und Risiken zu beachten, die beim Offshoring mit Fremdunternehmen nicht auftreten. Dazu gehören vor allem die Kosten für den Aufbau des neuen Standorts (Willcocks und Lacity 2006, S. 6).

Aus diesen Gründen wählen viele Unternehmen auch die Form eines Joint Ventures und mindern die Risiken, indem sie eine Partnerschaft mit einem lokalen Partner aufbauen (Vashistha und Vashistha 2006, S. 172 f.; Morstead und Blount 2003, S. 91). Dies ist ein Weg, den bereits viele Branchen bei der Erschließung internationaler Standorte mit Erfolg gegangen sind. Das nachfolgende Beispiel der Software AG zeigt das Vorgehen und Erfolgspotenziale.

Die SAG India

Ein Beispiel für ein Joint Venture ist die Zusammenarbeit der Software AG und iGate Global Solutions, die zur Gründung eines gemeinsamen Entwicklungs- und Service Centers in Indien unter dem Namen Software AG India (SAG India) führte. Der Hauptsitz des Unternehmens ist in Pune in der Nähe von Bombay/Mumbai.

Ein wesentlicher Vorteil des Aufbaus dieses Joint Ventures bestand darin, dass die SAG India gleich in der Startphase über iGate auf einen großen Pool qualifizierter IT-Fachkräfte in Indien zugreifen und dadurch rasch operativ tätig werden konnte.

Das neue Unternehmen wird in der Rechtsform einer Private Limited Company, grob vergleichbar einer GmbH, geführt. Die Software AG hält 51 % der Anteile, während iGate GS 49 % des Einlagevermögens hält. Die Unternehmen sind jeweils durch zwei Mitglieder in der Geschäftsleitung repräsentiert.

In der Startphase hatte die Software AG lediglich zwei Mitarbeiter in Indien: einen Mitarbeiter, der sich um die rechtlichen und wirtschaftlichen Belange kümmerte (z. B. Gründung der SAG India), und einen Mitarbeiter, der die Aufgaben, die in Indien durchgeführt werden sollten, genau kannte und gemeinsam mit Mitarbeitern von iGate die Einstellungen der indischen Kollegen vornahm. Der Mitarbeiter mit dem technischen Hintergrund hatte bei den vorangegangenen Outsourcing-Aktivitäten der Software AG bereits Erfahrungen sammeln können und war daher mit der Kultur vertraut. Dies wird im Nachhinein auch als ein wesentlicher Erfolgsfaktor gesehen.

Die Zusammenarbeit mit iGate hat den Prozess des Aufbaus an einem Offshore-Standort enorm beschleunigt, da iGate bereits über Gebäude, Infrastruktur, Telefon etc. verfügte. Ein weiterer Vorteil des Joint Ventures bestand darin, dass iGate eine Vielzahl wichtiger Kontakte hatte und die Software AG beispielsweise nicht selbst herausfinden musste, welche Behörde für welche Aufgaben zuständig ist.

Eine weitere, dem Joint-Venture-Modell nahe stehende Organisationsform ist das Build-Operate-Transfer-Modell (BOT). Hierbei sucht sich der Kunde wie bei einem Joint Venture einen lokalen Offshore-Partner, der dann ein Offshore Center errichtet und in Betrieb nimmt. Erst nachdem diese Phasen erfolgreich beendet sind, geht das Offshore Center in den Besitz des Kunden über (Vashistha und Vashistha 2006, S. 92). Der Vorteil dieses Modells liegt darin, dass einige der anfänglichen Risiken zumindest teilweise vom Partnerunternehmen getragen werden, ohne dass der Kunde auf die Option eines eigenen Tochterunternehmens verzichten muss.

Als Organisationsform mit der geringsten Bindungsintensität bietet sich die Vergabe von Aufträgen an Fremdunternehmen an – oben in Abb. 5.2 ganz rechts angesiedelt. Bei dieser Form des Offshorings gibt der Kunde einen großen Teil der Kontrolle an den Offshoring-Anbieter ab, der auf vertraglicher Basis seine Offshoring-Dienstleistung vollbringt. Für den Kunden ist damit der Vorteil relativ geringer operativer Risiken verbunden. Des Weiteren ist diese Organisationsform natürlich deutlich schneller umzusetzen als der Aufbau eines Joint Ventures oder einer Tochtergesellschaft (Robinson und Kalakota 2005, S. 29).

Darüber hinaus unterscheiden wir Offshoring-Aktivitäten in Nearshoring einerseits und Farshoring andererseits. Diese begriffliche Unterscheidung zielt auf die Entfernung zwischen Kunde und Anbieter ab. Wie die Namen bereits sagen, bezeichnet Nearshoring die Verlagerung von IT-Services an einen nahe gelegenen Standort, während Farshoring entsprechend die Verlagerung von Aufgaben an einen entfernten Standort beschreibt. Hierbei spielt nicht nur die geographische Entfernung eine Rolle, sondern auch die kulturelle Distanz oder auch die Anzahl der zwischen den Ländern liegenden Zeitzonen. Auf eine Aufarbeitung der Vielzahl von neu aufkommenden Buzzwords, wie Rightshoring, Bestshoring etc., wird an dieser Stelle verzichtet.

Aus Sicht deutscher Unternehmen würde beispielsweise das Offshoring in osteuropäische Länder oder nach Irland als Nearshoring bezeichnet werden. Für amerikanische Unternehmen zählen dagegen u. a. Mexiko und Kanada zu den bevorzugten Nearshore-Standorten. Selbst wenn die räumliche Entfernung zwischen einem Kunden in Kalifornien und einem kanadischen Anbieter, etwa in Halifax an der Ostküste, erheblich ist, würde man immer noch von Nearshoring sprechen, da die kulturellen und natürlich die sprachlichen Barrieren in diesem Fall normalerweise sehr gering sein dürften. Typische Länder des Farshorings sind sowohl aus amerikanischer als auch aus europäischer Sicht beispielsweise Indien, China oder Vietnam.

Bevor wir uns den Motiven für Outsourcing und Offshoring zuwenden, sollten wir uns noch klar machen, wer die Kunden der Dienstleistung eigentlich sind. Bislang wird in der Literatur überwiegend davon ausgegangen, dass es sich um Unternehmen handelt, deren Kernkompetenz nicht die Entwicklung von Software ist. Jedoch vergeben Softwareanbieter genauso Teile der Softwareentwicklung an Offshore-Standorte. In den meisten Fällen handelt es sich hierbei um ein Captive Offshoring oder den Aufbau von Joint Ventures, wie das Beispiel der Software AG India zeigt. Dies muss jedoch nicht so sein. So gibt es auch Beispiele, in denen Softwareunternehmen die Entwicklung ganzer Module an Drittanbieter auslagern. Kunden können also sowohl Anwender als auch Softwareanbieter sein.

5.3 Motive für Outsourcing und Offshoring

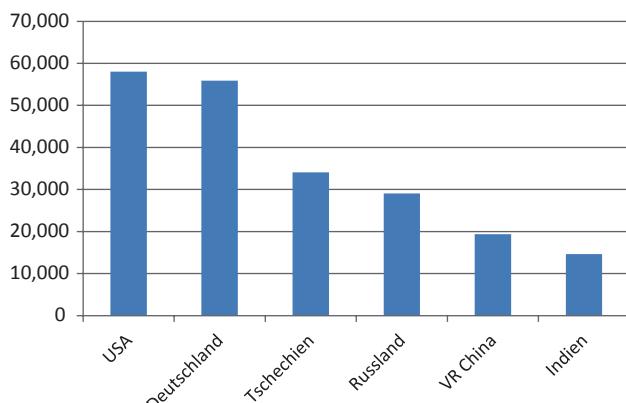
In diesem Abschnitt wollen wir auf die wesentlichen Motive für das Offshoring eingehen. Dabei handelt es sich insbesondere um (Amberg und Wiener 2006; Hirschheim et al. 2006)

- Kosteneinsparungen,
- Flexibilitätserhöhung,
- Konzentration auf Kernkompetenzen,
- Know-how-Zukauf sowie
- die Ausnutzung des „Follow-the-Sun“-Prinzips.

Kosteneinsparungen Einer der am häufigsten genannten Gründe für Outsourcing bzw. Offshoring sind die erwarteten Kosteneinsparungen, die größtenteils aufgrund der niedrigeren Lohnkosten an den Offshore-Standorten erzielt werden können. Wie hoch diese Kostenvorteile letztlich ausfallen, ist jedoch umstritten. Einerseits betragen die Lohnkosten von IT-Projektleitern an Offshore-Standorten nur einen Bruchteil des Gehalts eines in den USA oder Westeuropa Beschäftigten (siehe Abb. 5.3).

Andererseits führt Offshoring auch zu zusätzlichen Kosten, die insbesondere aus einem erhöhten Koordinationsaufwand resultieren. Es existieren mehrere Untersuchungen, etwa von Deutsche Bank Research oder dem Beratungsunternehmen neoIT, zur Schätzung von realisierbaren Kosteneinsparungen in Offshore-Projekten. Dabei sind die Ergebnisse eher als grobe Richtwerte denn als exakte Werte mit Allgemeingültigkeit zu verstehen. Bei dem Szenario eines offshore durchgeföhrten Softwareprojekts hält neoIT etwa die in Abb. 5.4 dargestellten Kosteneinsparungen für realistisch. Es zeigt sich, dass die durch deutlich niedrigere Gehälter eingesparten Kosten nur teilweise durch erhöhte Kosten aufgewogen werden, so dass sich insgesamt noch eine Kostenreduktion von 25–35 % erzielen lassen würde.

Abb. 5.3 Jahresgehälter in € für IT-Projektmanager im Vergleich (5–9 Jahre Erfahrung) (Payscale 2010, eigene Berechnungen)



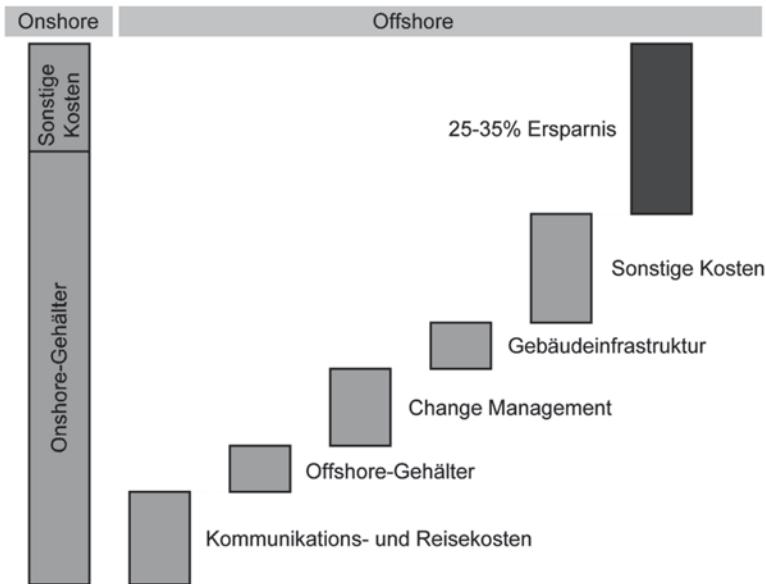


Abb. 5.4 Darstellung möglicher Kosteneinsparungen. (Kublanov et al. 2005, S. 3)

Erhöhung der Flexibilität Ein weiterer Reiz des Offshorings aus Kundensicht liegt in einer erwarteten Erhöhung der Flexibilität. Diese soll prinzipiell dadurch gewonnen werden, dass benötigte Dienstleistungen und Know-how je nach Bedarf eingekauft werden können. Damit ist auch der Vorteil einer Variabilisierung der Fixkosten verbunden.

Konzentration auf Kernkompetenzen Angesichts einer steigenden Wettbewerbsintensität besinnen sich viele Unternehmen auf ihre Kernkompetenzen. Nach dieser Logik bieten sich Bereiche, die nicht zu diesen Kompetenzen gehören, für das Outsourcing bzw. Offshoring an. Unter der Annahme, dass hoch spezialisierte Offshore-Anbieter IT-Dienstleistungen effizienter und besser erledigen können, liegt der Gedanke nahe, diese Aufgabenbereiche nach außen zu geben. Frei gewordene Kapazitäten könnten fortan für die Stärkung der Kernkompetenzen genutzt werden. Auf der anderen Seite können die Offshore-Anbieter durch eine Vielzahl von Aufträgen spürbare Skaleneffekte erzielen und ihre Dienstleistungen damit zumindest prinzipiell günstiger anbieten. So einfach und einleuchtend diese immer wieder angeführte Argumentation auch klingt, so kompliziert kann sie bei genauerer Betrachtung werden. Worin besteht etwa die Kernkompetenz eines Kreditinstituts, wenn mittlerweile auch Aufgaben wie die Kreditwürdigkeitsprüfung teilweise ausgelagert werden? Ein anderes Beispiel ist die Auslagerung von Entwicklungsprozessen in der Automobilindustrie an Zulieferer. Auch für die Softwareindustrie sind uns mehrere Fälle bekannt, in denen Softwarehäuser Teile ihrer Softwareentwicklung an Drittanbieter auslagern – nicht selten jedoch mit eher durchwachsenem Erfolg.

Know-how-Zukauf Ein weiterer häufig genannter Treiber des Offshorings ist der Arbeitsmarkt an Offshore-Lokationen. Zum einen ist dort die Zahl verfügbarer Fachkräfte deutlich höher als im Inland. Zum anderen besteht unter diesen Fachkräften eine hohe Motivation, insbesondere auch einfache und weniger anspruchsvolle Aufgaben durchzuführen (Hirschheim et al. 2006, S. 6 f.). Besonders Indien als klassischer Offshore-Standort, aber auch zum Beispiel China, verfügt über Absolventenzahlen, die um ein Vielfaches über denen in westeuropäischen Ländern liegen (NASSCOM 2007).

Darüber hinaus besitzt die IT-Industrie in Indien eine so positive Ausstrahlungskraft, dass viele hoch motivierte indische Mitarbeiter den Weg in diese Branche einschlagen (Vashistha und Vashistha 2006, S. 62).

Indien bringt aber – wie viele andere osteuropäische und asiatische Länder auch – nicht nur eine Vielzahl von Hochschulabsolventen hervor, diese sind in der Regel zudem sehr gut ausgebildet. So verfügen gerade indische IT-Fachkräfte häufig über sehr gute Qualifikationen im Bereich der mathematischen Grundlagenbildung sowie der Programmierung (Programmiersprachen, Datenbankkenntnisse etc.). Sowohl der Leiter des indischen Entwicklungszentrums von SAP als auch Manager von Daimler bestätigen, dass die Entwicklungsergebnisse und die Produktivität indischer, deutscher und amerikanischer Entwickler vergleichbar sind (Boes und Schwemmle 2005, S. 30). Untermauert werden die außerordentlich hohen Qualifikationen der Absolventen auch durch neuere Universitätsrankings.

Insofern verwundern auch die Ergebnisse einer Studie der Hackett Group nicht, wonach in 2016 etwa 920.000 Jobs in den Bereichen Finanzen, IT, Einkauf und HR von Europa und den USA nach Indien verlagert werden. Dies entspricht einem Drittel aller Jobs in diesen Bereichen (Hackett Group 2012).

Allerdings müssen wir die Euphorie an einer Stelle etwas bremsen: So berichteten uns viele Softwarefirmen von einer relativ geringen Loyalität indischer Mitarbeiter zu ihrem Arbeitgeber, und es ist dementsprechend keine Seltenheit, dass diese von der Konkurrenz sehr kurzfristig abgeworben werden. Dies führt etwa dazu, dass einige Softwareanbieter in jedem Projekt mehrere Projektmitarbeiter als „Backup“ mitlaufen lassen – für den Fall, dass einige im Laufe des Projekts zur Konkurrenz wechseln.

Ausnutzung des „Follow-the-Sun“-Prinzips Aufgrund der zwischen dem Standort des Kunden und der Offshore-Lokation liegenden Zeitverschiebung besteht grundsätzlich die Chance, die pro Tag zur Verfügung stehende Arbeitszeit auszuweiten. Im Idealfall kann das Onsite-Team vor Dienstende dem Offshore-Team noch eine Reihe von Aufgaben zuteilen, deren Ergebnisse bereits am nächsten Morgen onsite vorliegen (Hirschheim et al. 2006, S. 6; Amberg und Wiener 2006, S. 48; Thondavadi und Albert 2004, S. 18). Andererseits kann die Zeitverschiebung natürlich auch zu Schwierigkeiten bei der synchronen Kommunikation führen. Wir werden auf die hiermit verbundenen Chancen und Probleme in Abschn. 5.7.3 näher eingehen.

Abschließend wollen wir auf eine empirische Untersuchung von uns näher eingehen (Hess et al. 2007): Interessant sind die Erfahrungen der Unternehmen mit Near- und Farshoring. Tendenziell ist zu beobachten, dass erfahrene Unternehmen die Vorteile von

Offshoring höher einschätzen als Unternehmen ohne Offshoring-Erfahrung. Dazu konnten die Unternehmen in unserer Umfrage die Relevanz verschiedener Vor- und Nachteile auf einer Skala von „trifft gar nicht zu“ bis „trifft in hohem Maße zu“ bewerten. Abbildung 5.5 zeigt die Ergebnisse bezüglich des Nearshorings. Ein ähnliches Bild ergibt sich auch für das Farshoring.

Diese Erfahrungen schlagen sich offensichtlich auch auf den Blick in die Zukunft nieder: Knapp die Hälfte der Unternehmen in unserer Studie wollen in den nächsten Jahren noch mehr Aufträge nach außen geben. Die kleineren Unternehmen sind tendenziell zurückhaltender; dort planen lediglich 20% der Probanden einen Ausbau ihrer Offshoring-Aktivitäten (Hess et al. 2007).

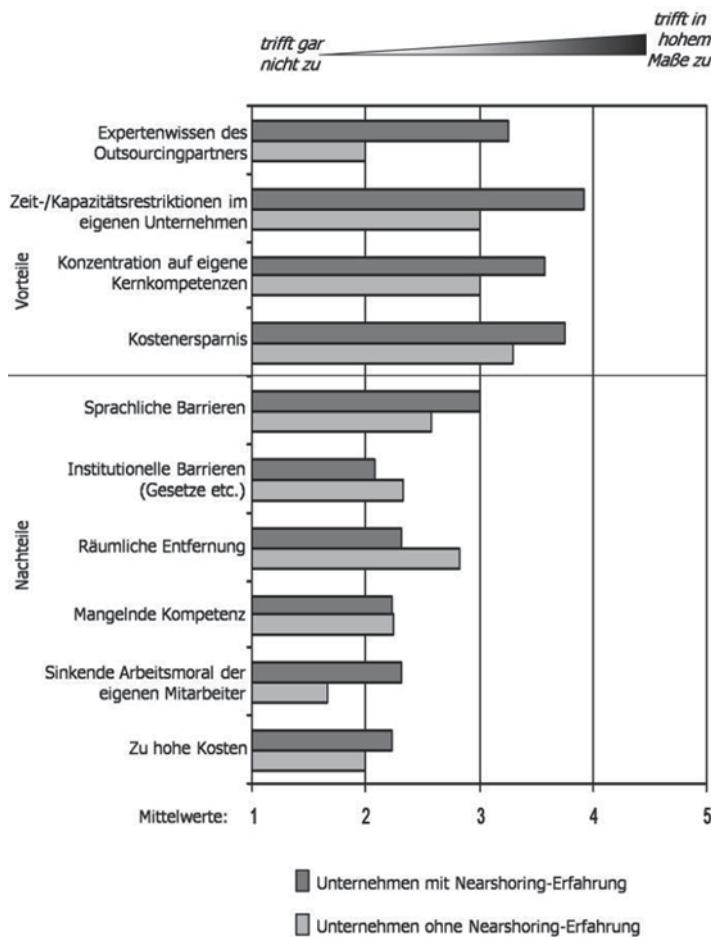


Abb. 5.5 Bewertung der Vor- und Nachteile des Nearshorings. (Hess et al. 2007, S. 5)

5.4 Abhangigkeit in IT-Outsourcing-Beziehungen

5.4.1 Grundlagen der Analyse

Neben den vielen potenziellen Vorteilen eines Outsourcings aus Kundensicht gibt es (naturlich) auch mogliche Nachteile. Dabei gelten der Verlust von Know-how sowie die Abhangigkeit von einem Anbieter als groes wahrgenommenes Risiko durch die Kunden. Diese furchten unter anderem, aus materiellen oder rechtlichen Grunden dauerhaft an den Partner gebunden und somit dessen Preis- bzw. Servicediktat ausgeliefert zu sein.

Wahrend die Forschung zu Abhangigkeiten im IT-Outsourcing noch in den Kinderschuhen steckt, reichen ihre Ursprnge in anderen Wissenschaftszweigen bis in die sechziger Jahre zurck (Emerson 1962). Schon die Arbeiten aus dieser Zeit legen nahe, das Verhaltnis zwischen Lieferant und Abnehmer aus beider Sicht zu betrachten. Nur so lassen sich die richtigen Schlusse ziehen, weil der Machtvorteil einer Vertragspartei eine asymmetrische Beziehung voraussetzt: Die uberlegene Partei hangt in geringerem Mae von der unterlegenen ab als diese von ihr. Hufig gefahrdet ein solches Machtgefalle den Geschftserfolg einer Outsourcing-Beziehung (z. B. Gulati und Sytch 2007; Lacity et al. 2009). So fuhrt eine starke Machtposition des Anbieters mitunter dazu, dass sich der Kunde mit einer vergleichsweise schwachen Leistung zufrieden geben muss. Doch auch der umgekehrte Fall kommt vor: Die Qualitt leidet, weil ein Großkunde den Preis- oder Zeitdruck uber die Maen erhoht. Unternehmen, die die Gefahr einseitiger Abhangigkeit erkennen, werden leicht Opfer opportunistischen Verhaltens ihrer Vertragspartner oder verschenken Leistungen, die sie bei ausgewogener Machtverteilung einfordern konnten (Kaiser und Buxmann 2012).

In diesem Abschnitt fokussieren wir zum einen die Faktoren der Abhangigkeit des Kunden vom Anbieter in einer IT-Outsourcing-Beziehung. Zum anderen wollen wir aber auch die Abhangigkeit der Anbieter in die Analyse einbeziehen. Dazu haben wir die Fachliteratur ausgewertet und im Rahmen von finf Fallstudien Vertreter beider Seiten befragt (Kaiser et al. 2012). Bei den finf Fallen handelt es sich um jeweils eine Outsourcing-Beziehung zwischen einem Unternehmen aus der Luftfahrindustrie mit mehr als 40.000 Mitarbeitern und je einem Outsourcing-Anbieter. Dabei gibt das Kundenunternehmen mehr als 70% des eigenen IT-Budgets fur Outsourcing aus und hat dementsprechend eine Vielzahl von Outsourcing-Anbietern. Die folgende Tabelle gibt einen Uberblick uber die finf Falle (Tab. 5.1).

Auf Kundenseite wurden insgesamt 12 Experten aus finf verschiedenen Abteilungen befragt. Zudem wurden auf Anbieterseite sieben Interviews mit Mitarbeitern gefuhrt, u. a. mit Projektmanagern und mit Key Account Managern.

Bevor wir uns die Ergebnisse ansehen, wollen wir zunachst einige Determinanten der Abhangigkeit nher untersuchen. In der Fachliteratur wird vorgeschlagen, zwischen der Relevanz der extern bezogenen Ressource und deren Verfugbarkeit zu unterscheiden (Jacobs 1974). Letztere bemisst sich an der Zahl der Alternativen. Dieses Kriterium greift jedoch zumindest dann zu kurz, wenn weitere Grunde ein Unternehmen an einen Anbieter

Tab. 5.1 Übersicht der Fälle. (Kaiser et al. 2012, S. 5)

	Fall 1	Fall 2	Fall 3	Fall 4	Fall 5
Kurzbeschreibung	Sales plattform	Self-service plattform	Core service system	CRM plattform	GUI für customer service
Anbieter- beschreibung (approx. Werte)	Weltweit, 400.000 Mitarbeiter, Umsatz 100 Mrd. \$	Europa- weit, 1.300 Mitarbeiter, Umsatz 150 Mio. €	Weltweit, 10.000 Mitarbeiter, Umsatz 2,5 Mrd. €	International, 500 Mitarbeiter, Umsatz 200 Mio. €	In-house Provider, weltweit, 3.000 Mitarbeiter, Umsatz 600 Mio. €
Vertragslaufzeit	2009–2015	2009–2011	2005–2020	2008–2020	2011–2015
Vertragstyp und -umfang	Nutzungs- abhängig, 20 Mio. €	Rahmen- vertrag, 2,9 Mio. €	Rahmen- vertrag, 400 Mio. €	Preisfixiertes Volumen, o. A.	Festpreis, 10 Mio. €
Aktuelle Phase im IS Lifecycle	Wartung	Ende der Entwicklung	Entwicklung/Wartung	Entwicklung	Wartung, Anbieter- wechsel

binden – etwa weil es transaktionsspezifisch investiert hat (siehe hierzu Abschn. 2.4 dieses Buchs) oder weil eine alternative Geschäftsbeziehung einen geringeren Nutzen erwarten lässt. Der theoretischen Verfügbarkeit einer Alternative stellen daher einige Autoren die effektive Substituierbarkeit der Ressource gegenüber. Sie berücksichtigt die Schwierigkeit, einen Anbieter durch einen anderen zu ersetzen.

Grundsätzlich eignen sich die beiden Determinanten Relevanz und Substituierbarkeit auch zur Bewertung der Abhängigkeit der Kunden beim IT-Outsourcing. Um die Besonderheiten dieser Abhängigkeit zu erfassen, bedürfen die Determinanten jedoch der weiteren Differenzierung (Kaiser et al. 2012, S. 7). Die linke Seite der nachstehenden Grafik stellt den Fall dar, dass ein einziger Anbieter bzw. Lieferant sämtliche Funktionen eines IT-Systems bereitstellt. Zudem gehen wir davon aus, dass dieses System mindestens einen Geschäftsprozess unterstützt. Handelt es sich dabei um einen weniger wichtigen Back-office-Prozess, so ist die Relevanz des Systems gering. Flankiert es hingegen einen Kernprozess wie den Vertrieb, ist sie hoch.

Die Substituierbarkeit untersuchen wir am Beispiel des Wechsels des externen Anbieters. Teils läuft eine solche Entscheidung auf einen Systemwechsel hinaus, teils kann der neue IT-Partner das System übernehmen. Weitere Alternativen wären die Beauftragung einer Tochter- oder Schwestergesellschaft, die interne Weiterentwicklung beziehungsweise der interne Betrieb (Abb. 5.6).

Die rechte Seite der Grafik bildet den häufigen Fall ab, dass mehrere Dienstleister an der Entwicklung eines IT-Systems mitwirken. Hier reduziert sich die Abhängigkeit vom einzelnen Partner auf die Relevanz und Substituierbarkeit seiner Leistung.

Vor diesem Hintergrund wollen wir im Folgenden auf die Ergebnisse der Fallstudien eingehen. Dabei erfolgt in einer dyadischen Betrachtung eine Analyse der Kunden- und

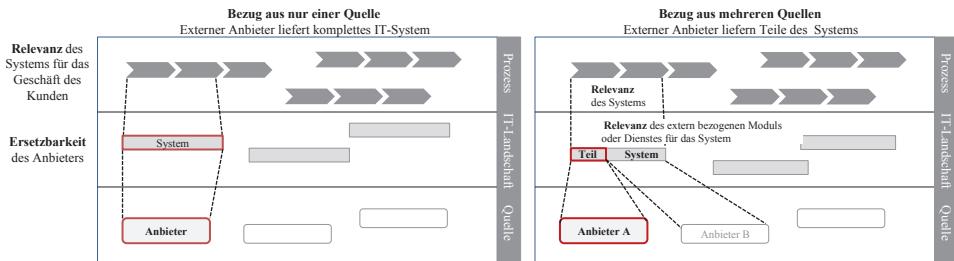


Abb. 5.6 Determinanten der Abhängigkeit des IT-Kunden. (Kaiser et al. 2012, S. 7)

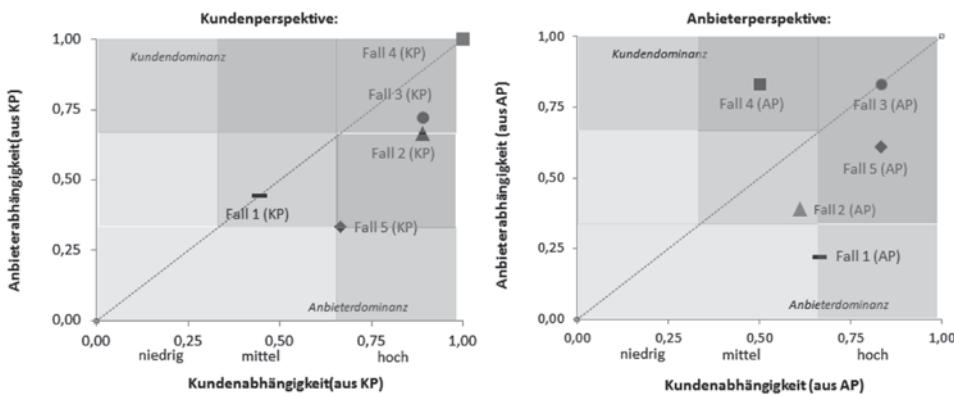


Abb. 5.7 Abhängigkeiten aus Kunden- und Anbieterperspektive. (Kaiser et al. 2012, S. 9)

der Anbieterabhängigkeit zum einen aus der Perspektive der Kunden und zum anderen auch aus der des Anbieters. Die linke so genannte Power-Map in der folgenden Abb. 5.7 zeigt die Abhängigkeiten aus Kunden-, die rechte aus Anbieterperspektive.

Auf der Abszisse ist jeweils die Kundenabhängigkeit aufgetragen, während die Ordinate die Anbieterabhängigkeit repräsentiert.

Aus Kundenperspektive variiert die Abhängigkeit von einem mittleren Wert (0,44 bei Fall 1) bis zu einem sehr hohen Wert (1 bei Fall 4). Für alle Fälle gilt, dass die wahrgenommene Abhängigkeit des Anbieters im Vergleich zur Kundenabhängigkeit als gleich hoch oder niedriger empfunden wird. Daher gibt es keinen Fall, in dem der Kunde einen Machtvorteil empfindet (Kaiser et al. 2012, S. 9).

Interessant ist es nun, die Perspektive der Kunden mit der Anbietersicht zu vergleichen. Während der Kunde etwa in den Fällen 1 und 4 eine ausbalancierte Abhängigkeit wahrnimmt, sieht der Anbieter im Fall 4 eine höhere eigene Abhängigkeit. Demgegenüber sieht der Anbieter im Fall 1 eine höhere Abhängigkeit des Kunden. In den anderen Fällen wird auch aus Anbietersicht eine höhere Abhängigkeit des Kunden wahrgenommen. Die entsprechenden Abhängigkeitswerte sind in der folgenden Tabelle dargestellt (Tab. 5.2).

Im Folgenden wollen wir auf die Relevanz sowie die Substituierbarkeit als Determinanten der Abhängigkeit näher eingehen. Zur Illustration werden Zitate aus unseren Expertengesprächen in den Text integriert.

Tab. 5.2 Abhängigkeitswerte für Kunden und Anbieter sowie für die relative und gemeinsame Abhängigkeit. (Kaiser et al. 2012, S. 9)

	Kundenperspektive (KP)				Anbieterperspektive (AP)			
	Wahrge- eigene Abh. (1)	Wahrge- Anbieter- abh. (2)	Relative Abh. (1)–(2)	Gemein- same Abh. (1)+(2)	Wahr- gen. Kunden- abh. (1)	Wahrge- eigene Abh. (2)	Relative Abh. (1)–(2)	Gemein- same Abh. (1)+(2)
Fall 1	0,44	0,44	0,00	0,88	0,67	0,22	0,45	0,89
Fall 2	0,89	0,67	0,22	1,56	0,61	0,39	0,22	1,00
Fall 3	0,89	0,72	0,17	1,61	0,83	0,83	0,00	1,66
Fall 4	1,00	1,00	0,00	2,00	0,50	0,83	-0,33	1,33
Fall 5	0,67	0,33	0,34	1,00	0,83	0,61	0,22	1,44

5.4.2 Relevanz

Als Faktoren der Relevanz einer IT-Leistung sehen wir nach Auswertung der Fachliteratur und unserer Interviews den Umfang ihrer Beschaffung sowie ihren Anteil an der Wertschöpfung des Kunden.

Relativer Umfang Der relative Umfang einer Ressource lässt sich als Anteil am Gesamtvolumen des Einkaufs beziffern (Gulati und Sytch 2007). Bezogen auf das IT-Budget heißt das: Sind mehrere Anbieter an der Entwicklung eines Systems beteiligt, so sinkt gegenüber dem Bezug aus einer Hand die Abhängigkeit des Auftraggebers vom einzelnen Partner. In den meisten der von uns untersuchten Fälle gab es entweder nur einen Lieferanten oder es trat jeweils ein Anbieter als Generalunternehmer auf. Einem der befragten Unternehmen gelang es, durch Kündigung eines Generalunternehmervertrags den Umfang der Leistungen der anderen Partei und damit die Abhängigkeit von ihr zu reduzieren (Kaiser et al. 2012, S. 11):

Unsere Abhängigkeit wurde geringer, nachdem wir den Vertrag mit dem Generalunternehmer gekündigt hatten und ein wichtiges Modul direkt von einem ehemaligen Subunternehmer bezogen.

Relativer Wertbeitrag Der relative Umfang am Einkaufsvolumen lässt sich zwar leicht feststellen, gibt die Relevanz einer extern bezogenen Ressource aber nicht vollständig wieder. Die Marketing-Literatur nennt als weiteres Maß der Relevanz einer Ressource deren Beitrag zu Umsatz und Gewinn (Heide und John 1988; Kumar et al. 1995; Geyskens et al. 1996). IT-Systeme beeinflussen diese Kennzahlen zumindest mittelbar, indem sie dem Anwender helfen, seine Kosten zu senken, die Qualität seiner Produkte zu verbessern, Geschäftsprozesse zu beschleunigen oder weitere Marktsegmente zu erschließen. Ebenfalls können IT-Systeme durch Unterstützung der Einhaltung bilanz- und aufsichtsrechtlicher Vorschriften helfen, Verluste zu vermeiden.

Dass wir neben dem Umfang einer IT-Leistung deren Wertbeitrag berücksichtigen, hat folgenden Grund: Auch wenn ihr Anteil am Einkaufsvolumen gering ist, kann sie für das Kerngeschäft unentbehrlich sein – oder umgekehrt. So dürfte bei vergleichbarem Kostenaufwand die Wartung der Vertriebsssoftware eines Unternehmens einen höheren Wertbeitrag leisten als die Pflege eines Backoffice-Systems. Ebenso können Systemmodule zweier Lieferanten trotz ähnlichem Preis in unterschiedlichem Maße zum Geschäftsergebnis beitragen (Kaiser et al 2012, S. 11).

Wenn wir die Arbeit einstellen, droht der Ausfall einer zentralen Anwendung. Insofern ist der Kunde auf uns angewiesen.

5.4.3 Substituierbarkeit

Aus der Zusammenschau unserer Studie mit früheren Beiträgen zu den Themen Wechselkosten (Jones et al. 2002; Whitten und Wakefield 2006) und Abhängigkeit haben wir sieben Faktoren abgeleitet, von denen die Substituierbarkeit eines IT-Lieferanten abhängt (Kaiser et al. 2012, S. 12–13).

Anzahl der Alternativen Je geringer die Zahl der Alternativen ist, desto schwieriger ist der bisherige Anbieter zu ersetzen und desto höher ist folglich die vom Kunden wahrgenommene Abhängigkeit (Pfeffer und Salancik 1978; Ganesan 1994; Gulati und Sytch 2007). Diese Erkenntnis belegen folgende Aussagen unserer Interviewpartner:

Unsere geringe Abhängigkeit sehe ich darin begründet, dass es am Markt genügend Anbieter gibt, die dasselbe liefern könnten wie unser derzeitiger Partner.

Momentan sind wir auf unseren Lieferanten angewiesen. [...] Außerdem gibt es nur wenige Anbieter, die ein vergleichbares CRM-System liefern könnten.

Evaluierungs- und Auswahlaufwand Doch selbst wenn der Kunde brauchbare Alternativen kennt, steht einem Anbieterwechsel manches entgegen. Zur Überwindung dieser Hürden muss der Anwender Transaktionskosten aufwenden. Soll zum Beispiel ein Großsystem ausgetauscht werden, so erfordert der Prozess von der Vorauswahl über die genaue Prüfung der Optionen bis zur Entscheidung neben Sachkunde einen erheblichen zeitlichen und materiellen Aufwand. Der Anwender muss unter anderem wissen, was das Altsystem kann, und es mit den Alternativen vergleichen. Kennt sich im Unternehmen niemand mehr mit dem System aus, wird die Wahl zum Problem. Müssen Expertenwissen oder sonstige Ressourcen erworben werden, fallen wiederum Kosten an. Hinzu kommen das Aushandeln und die Prüfung des neuen Vertrags.

Die Bewertung der am Markt erhältlichen Alternativen war extrem aufwendig. So etwas möchte man sich nicht jedes Jahr antun.

Den Aufwand zur Analyse und Spezifikation der Funktionen des Altsystems darf man bei einer Neuaußschreibung nicht unterschätzen.

Ungewissheit der Leistung anderer Anbieter Der Erfolg des IT-Outsourcings steht und fällt mit der Kompetenz und Leistung des Anbieters (Grover et al. 1996). Für den Kunden problematisch ist, dass er diese zum Zeitpunkt seiner Entscheidung nicht genau einschätzen kann. Zwar lässt sich die Wissenslücke mit viel Recherche ein Stück weit schließen. Dennoch halten wir die verbleibende Ungewissheit für eine Wechselbarriere. Dafür sprechen folgende Aussagen:

Sicher könnten andere das System für uns betreiben. Ich bezweifle aber, dass sie dies genauso gut täten.

Wir könnten diese Leistung auch anderweitig beziehen. Aber wäre das wirklich ein Gewinn?
Es ist ja keineswegs ausgemacht, dass der neue Anbieter besser arbeitet.

Irreversible Kosten Unter irreversiblen oder „versunkenen“ Kosten (*sunk costs*) versteht man den unwiederbringlichen zeitlichen und materiellen Aufwand, den der Kunde in ein Vertragsverhältnis investiert hat (Jones et al. 2002; Whitten und Wakefield 2006). In der normativen Ökonomik gelten diese Kosten als irrelevant, weil sie sich nicht mehr ändern lassen und weil man nur rational entscheidet, wer den *künftigen Nutzen* und dessen Kosten betrachtet (Arkes und Blumer 1985; Whyte 1994). Dennoch kann der Blick auf die versunkenen Kosten etwa einen IT-Entscheider dazu bewegen, an einem unproduktiven Entwicklungsprojekt festzuhalten (Keil et al. 1995). Ein Paradebeispiel solcher Kosten sind transaktionsspezifische Investitionen, die sich nicht umwidmen lassen. Beim IT-Outsourcing betrifft das unter anderem die Schulung auf das Altsystem oder dessen Entwicklung, wenn der Kunde es versäumt hat, sich das Recht zur Wartung und Weiterentwicklung zu sichern, und deshalb der Wechsel zu einem anderen Anbieter ausscheidet. Auch in unserer Studie zeichnete sich ab, dass irreversible Kosten als Wechselbarriere wirken und somit zumindest subjektiv die Abhängigkeit verstärken:

Wären wir im Besitz des Quellcodes, hätten wir uns natürlich nach einem anderen Partner umgesehen, der die Software weiterentwickelt hätte. Doch so, wie die Dinge liegen, wäre der bisherige Entwicklungsaufwand umsonst gewesen.

Entgangener Nutzen Ein weiteres Motiv, an einer Lieferbeziehung festzuhalten, können exzellente Leistungen des bisherigen Anbieters sein. Überlegungen dieser Art behandelt die Theorie des sozialen Austauschs, wenn sie die Kosten-Nutzenbilanz einer Beziehung mit der Erwartung an einen alternativen Partner vergleicht. Auch Sonderkonditionen, eine kompetente Beratung oder der technische Support fließen in die Beurteilung der Ersetzbarkeit ein und steigern das Gefühl der Abhängigkeit – es sei denn, der Kunde verspricht sich von einem alternativen Anbieter einen höheren Nutzen.

Die Zusammenarbeit mit diesem Lieferanten ist vorbildlich. Er erfüllt seine Vertragspflichten pünktlich und vollständig. Bei anderen Partnern ist das nicht der Fall.

Dieser Anbieter liefert uns einen deutlich größeren Funktionsumfang. Zudem profitieren wir von Synergien, die sich bei anderen Lieferanten nicht einstellen.

Kosten nach der Wechselentscheidung Die Entscheidung für einen anderen Anbieter zieht Folgekosten nach sich. So muss der Auftraggeber den neuen Dienstleister speziell in der Anfangsphase oft mit eigenem Personal verstärken. Hinzu kommen Ausgaben etwa für Schulung oder externe Beratung. Ebenfalls zu beachten sind die Kosten, die dem Kunden durch die Koordination und den Vollzug des Umstiegs entstehen. Darunter fällt auch der Zeitaufwand zur Anpassung an geänderte Regeln und Prozesse in der Interaktion mit dem neuen Partner (Jones et al. 2002; Whitten und Wakefield 2006). Die Folgekosten fallen in der Regel umso höher aus, je stärker die Verhandlungsposition des Lieferanten ist, weil dieser dadurch Spielraum gewinnt, die Zusammenarbeit in seinem Interesse zu gestalten.

Der Zeitaufwand, den ein Wechsel bedeuten würde, vergrößert die Abhängigkeit vom alten Partner. Schließlich müssten wir zur Migration von A nach B ein eigenes Projekt aufsetzen. Und wenn es viel zu migrieren gibt, ist die Hemmschwelle besonders hoch.

Man darf nicht unterschätzen, wie viel Arbeit die Ausschreibung und der Umstieg mit sich bringen. Ohne zusätzliche Ressourcen ginge dies zu Lasten der Innovation, weil die Pflege des Altsystems zu kurz käme.

Rüstkosten des neuen Anbieters Rüstkosten sind diejenigen materiellen und immateriellen Investitionen in die Zusammenarbeit mit dem neuen Anbieter, die ihn dazu befähigen, das IT-System vertragsgemäß weiterzuentwickeln oder zu betreiben. Dazu zählt zunächst die Vermittlung der funktionalen Anforderungen und des ihnen zugrunde liegenden Geschäftswissens. Kann oder darf der neue Partner Teile des Systems nicht weiterverwenden, so muss er sie nachentwickeln; die Kosten trägt in der Regel der Kunde. Weitere Rüstkosten entstehen durch die Einarbeitung beispielsweise in die Schnittstellen zu sonstigen Systemen des Auftraggebers. Einen Hinweis auf die Größenordnung dieser Ausgaben liefert die Dauer des Übergangs vom alten zum neuen Lieferanten. Gerade bei der Übergabe der Wartung eines IT-Systems kann sich der Wissenstransfer in die Länge ziehen:

Im Falle eines Wechsels muss sich der neue Anbieter zunächst einarbeiten. Obendrein wird dieser in den ersten Jahren weniger effizient arbeiten.

5.4.4 Spillover

Die insgesamt neun bislang beschriebenen Faktoren der Relevanz und der Substituierbarkeit fassen wir in der nachstehenden Tabelle zusammen. Bei der Auswertung unserer Interviews sind wir auf einen weiteren Faktor gestoßen, der beim Kunden das Gefühl der Abhängigkeit steigert: Ein Unternehmen steht zu einem IT-Lieferanten oft in mehreren Vertragsbeziehungen. Lässt es einen Vertrag auslaufen oder kündigt ihn, drohen Sanktionen des Partners in sonstigen Feldern der Zusammenarbeit. Das Schadenpotenzial steigt mit der Zahl der Beziehungen, in denen der Lieferant über einen Machthebel verfügt. Unerwünschte Reaktionen dieser Art wurden in etlichen Interviews als Wechselbarrieren genannt. Ein klassischer Fall ist, dass der Partner in noch laufenden Vertragsbeziehungen die Preise erhöht. Repressalien lassen sich auch außerhalb des IT-Outsourcings beobachten.

Tab. 5.3 Zehn Faktoren der Kundenabhängigkeit in IT-Outsourcing-Beziehungen. (Kaiser et al. 2012, S. 14)

Determinante	Faktor	Erklärung
Relevanz	1 relativer Umfang (+)	Anteil des IT-Systems, eines Moduls oder Dienstes etwa am Einkaufsvolumen
	2 relativer Wertbeitrag (+)	Relativer Beitrag des betreffenden Systems, Moduls oder Dienstes zur Wertschöpfung
Substituierbarkeit	3 alternative Anbieter (+)	Zahl der Anbieter eines vergleichbaren Systems, Moduls oder Dienstes
	4 Evaluierungs- und Auswahl- aufwand (-)	zeitlicher und materieller Aufwand zur Angebotsprüfung sowie zur Wahl des neuen Lieferanten
	5 Ungewissheit der Leistung anderer Anbieter (-)	Unkenntnis oder Zweifel an der Leistung alternativer Anbieter
	6 irreversible Kosten (-)	Nicht anderweitig nutzbare zeitliche und materielle Investitionen in das alte Vertragsverhältnis
	7 entgangener Nutzen (-)	Sonderkonditionen und nicht substituierbare Leistungen
	8 Kosten nach der Wechsel- entscheidung (-)	Eigener zeitlicher und finanzieller Aufwand des Kunden anlässlich des Anbieterwechsels
	9 Rüstkosten des neuen Anbieters (-)	Dem Kunden fakturierte Ausgaben des Auftragnehmers zur Wiederherstellung des früheren Entwicklungsstands
	10 Spillover (+)	Wegen einer (geplanten) Kündigung erwartete Sanktionen des Partners in anderen Feldern der Zusammenarbeit

ten, etwa wenn der frühere Lieferant zugleich ein bedeutender Abnehmer der Produkte des Kunden ist. Im Extremfall hält die befürchtete Umsatzeinbuße den Kunden selbst dann von einer Kündigung ab, wenn er mit dem Dienstleister unzufrieden ist und diesen fachlich durchaus ersetzen könnte (Tab. 5.3) (Kaiser et al. 2012, S. 14):

Der Partner liefert uns nicht nur die Systeme für das Kerngeschäft, sondern auch die Schnittstellen zu sonstigen Anwendungen. Er könnte dafür prohibitive Preise verlangen oder uns den Umstieg massiv erschweren. Daher wäre ein Wechsel nicht sinnvoll.

Der Lieferant ist bei uns auch Kunde. Er nimmt uns beachtliche Mengen ab, kaufte aber genau in jenem Jahr einen Teil davon bewusst bei der Konkurrenz.

5.5 Standortwahl von Softwareanbietern

Wie wir im vorherigen Abschnitt gesehen haben, wird von den auslagernden Unternehmen insbesondere das Ziel verfolgt, Kosteneinsparungen zu realisieren. Vor diesem Hintergrund ist die Standortwahl auch für die Anbieter von IT-Services von strategischer

Bedeutung. Zum einen besteht für sie die Chance, Leistungen durch geringere Lohnkosten in Niedriglohnländern billiger zu produzieren; zum anderen sind bei der Entscheidung über die Standorte aber auch Vorteile einer großen Kundennähe zu berücksichtigen. So lässt die SAP AG verschiedene Module in Indien entwickeln und will diesen Standort auch weiter ausbauen. Damit wird einerseits das Ziel verfolgt, die Entwicklungskosten zu senken. Andererseits bietet die Standortwahl eine größere räumliche Nähe zum Zukunftsmarkt Indien, der von der SAP AG als einer der acht Wichtigsten weltweit eingeschätzt wird. Darüber hinaus hat die SAP AG ein Netz von Fall-Back-Zentren in mehreren asiatischen Großstädten aufgebaut (Mertens et al. 2005, S. 15 f., 21).

Bislang verlagerten die Anbieter ihre Standorte überwiegend von Hochlohn- in Niedriglohnländer. In der letzten Zeit ist jedoch auch eine gegenläufige Tendenz zu beobachten. So beginnen beispielsweise indische Softwarefirmen, Nachwuchs an europäischen und amerikanischen Universitäten zu akquirieren und Niederlassungen in den Hochlohnländern zu eröffnen. Dabei nehmen diese Anbieter offenbar höhere Lohnkosten in Kauf, um näher an den potenziellen Kunden zu sein.

Grundsätzlich sind bei Standortentscheidungen insbesondere die folgenden Faktoren zu berücksichtigen (Meyer-Stamer 1999; Kearney 2004):

- harte Standortfaktoren, wie etwa die Nähe zu Bezugs- und Absatzmärkten, Verkehrsanbindung, Arbeitskräftepotenzial, Lohn- und Gehaltsniveau oder lokale Steuern,
- weiche unternehmensbezogene Standortfaktoren, beispielsweise Wirtschaftsklima, Branchenkontakte, Kooperationsmöglichkeiten, Hochschulen und Forschung sowie
- weiche personenbezogene Standortfaktoren, wie Wohnsituation, Umweltqualität, Schulen, soziale Infrastruktur, Freizeitwert.

Zur Bewertung von Standorten sind verschiedene Indizes entwickelt worden, die häufig auf so genannten Scoring-Modellen aufbauen. Ein Beispiel hierfür ist der Local Attractiveness Index der Unternehmensberatungsgesellschaft A.T. Kearney.

Um die Vorteile einer globalen Standortwahl auszunutzen, bauen Anbieter wie beispielsweise Accenture oder Cognizant weltweite IT-Delivery-Netzwerke auf. Damit ist zum einen der Vorteil verbunden, dass Onshore- und Offshore-Aktivitäten für einen Kunden entsprechend skaliert werden können. Zum anderen kann ein solches Netzwerk auch als Backup-Strategie angesehen werden. In der folgenden Abb. 5.8 ist ein Ausschnitt der weltweiten IT-Delivery-Center von Accenture dargestellt.

In diesem Zusammenhang gewinnt der Aufbau von so genannten Global Delivery Models (Prehl 2006, S. 38) an Bedeutung. Hiermit ist mehr als nur das reine Offshoring gemeint. Vielmehr wird damit ein Mix aus Onshoring, Nearshoring und Farshoring beschrieben. Vereinzelt wird diese Organisationsform auch als ein Offshore-Modell der zweiten Generation bezeichnet – bei der „Kreativität“ einiger Marketingexperten im Hinblick auf die Erfindung neuer Begriffe ist es wohl nur eine Frage der Zeit, bis von Offshoring 2.0 gesprochen wird.



Abb. 5.8 Ausschnitt aus dem weltweiten Netzwerk der Servicezentren von Accenture. (Die Weltkarte basiert auf dem Bild „BlankMap-World gray“ von Vardion, lizenziert unter Creative Commons CC-BY-SA 2.5)

Insbesondere große und international tätige Softwareanbieter sind durch breit aufgestellte Ressourcen in der Lage, die Teilaufgaben eines Projekts speziell nach ihren Anforderungen entweder onshore, nearshore oder farshore durchzuführen. Tabelle 5.4 zeigt am Beispiel von Infosys, wie eine Verteilung der im Rahmen eines Softwareentwicklungsprojekts anfallenden Aufgaben auf die verschiedenen Standorte aussehen kann. Die genaue Organisation hängt dabei natürlich von dem jeweiligen Projekt ab.

Die Betrachtung der internen Aufbau- und Ablauforganisation von Softwarehäusern zeigt, dass das Konzept, nachdem im Hochlohnland der Fachentwurf erstellt wird, die reine Programmierung und die Systemtests aber in Offshore-Lokationen erfolgen, in vielen Fällen der Vergangenheit angehört. Auch wenn dieses – fast schon klassische – Modell der Arbeitsteilung in Offshore-Projekten natürlich noch zu finden ist, so geht der Trend hinsichtlich der Organisation der Zusammenarbeit eher weg von der Hierarchie hin zu einer Kooperation auf Augenhöhe. So sind beispielsweise bei der SAP AG die Verantwortlichkeiten zwischen den verschiedenen Standorten nach Modulen aufgeteilt. Das heißt auch, dass erstmals Entwicklungsmanager in Walldorf an Vorgesetzte in Indien berichten.

Tab. 5.4 Softwareentwicklung im Global Delivery Model von Infosys. (in Anlehnung an o.V. 2006a, S. 131)

Onshore	Nearshore	Farshore
Architecture	Requirements analysis	Detailed design
Requirements	High-level design	Code development
Change management	Prototype building	Testing and integration
Implementation	Implementation support	

5.6 Outsourcing durch Softwareanwender

Eine von Lünendonk (2013) durchgeführte Befragung unter 50 Großunternehmen aus Deutschland zeigt, dass die befragten Unternehmen etwa 50 % ihres Budgets für externe Dienstleister im Sinne von Outsourcing verwenden. Für das Jahr 2014 ist eine leichte Verschiebung des Budgets hin zur Inanspruchnahme interner Ressourcen geplant. Die Vielzahl parallel geplanter und umzusetzender Projekte, deren oft komplexe Technologie sowie eine nicht ausreichende Zahl von qualifizierten internen IT-Fachberatern lässt offen, ob der interne Budgetanteil tatsächlich steigen wird. Gespräche mit IT-Verantwortlichen deuten darauf hin, dass vor allem bei internationalen IT-Projekten sowie bei Prozessveränderungsprojekten verstärkt auf externe Ressourcen zurückgegriffen wird respektive zurückgegriffen werden muss (Lünendonk 2013, S. 30).

In diesem Abschnitt wollen wir vor diesem Hintergrund Ergebnisse zum Outsourcing der Neuentwicklung, Anpassung und Wartung von Anwendungssoftware aus Sicht von Softwareanwendern vorstellen (Buxmann et al. 2010). Adressiert wurden CIOs von großen, mittleren und kleinen Unternehmen. Insgesamt wurden von ca. 5.500 kontaktierten Unternehmen 498 verwertbare Fragebögen ausgefüllt. Die Befragung wurde branchenübergreifend durchgeführt. Dabei soll zum einen der Status quo der Fremdvergabe in den genannten Bereichen dargestellt werden. Darüber hinaus zeigen die Ergebnisse, welche Anforderungen an die Software- und Servicehäuser von den Anwendern gestellt werden. Im Folgenden wollen wir uns schwerpunktmäßig mit der Neuentwicklung von Individualsoftware, der Anpassung von Standardsoftware sowie der Weiterentwicklung und Wartung von Anwendungssoftware beschäftigen.

5.6.1 Outsourcing der Neuentwicklung von Individualsoftware

Grad der Inanspruchnahme von Fremdleistungen Im Zusammenhang mit der Neuentwicklung von Individualsoftware wurde zunächst erhoben, ob die befragten Unternehmen derzeit ein größeres Individualsoftwaresystem im Einsatz haben. Etwa 47 % der Teilnehmer gaben an, ein solches einzusetzen. Dagegen verneinten ca. 53 % der Teilnehmer die Frage nach dem Einsatz eines Individualsystems und setzen demnach überwiegend standardisierte Anwendungssoftware ein (siehe Abb. 5.9).

Knapp 80 % der Unternehmen, die ein Individualsoftwaresystem einsetzen, haben für die Neuentwicklung Fremdleistungen in Anspruch genommen (siehe Abb. 5.10). Die restlichen Unternehmen haben demgegenüber keinen externen Dienstleister beauftragt. Dabei hat etwa ein Drittel der Befragten, die für die Neuentwicklung von Individualsoftware Fremdleistung in Anspruch genommen hat, die Entwicklung ausschließlich durch externe Dienstleister vornehmen lassen. Demgegenüber wurde von etwa zwei Dritteln der Unternehmen die Individualsoftware sowohl durch externe Dienstleister als auch mit Hilfe interner Mitarbeiter entwickelt (siehe Abb. 5.11).

Abb. 5.9 Anteil Individualsoftwaresysteme

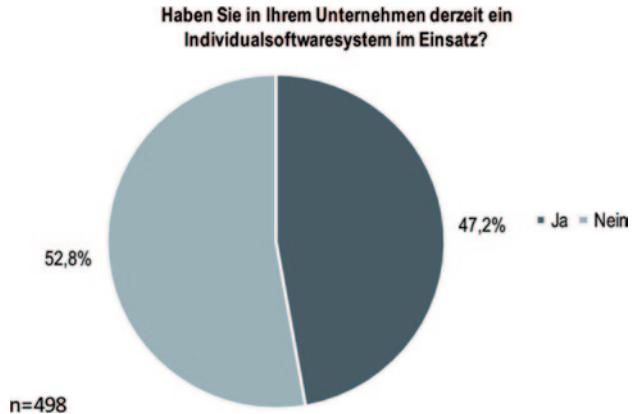


Abb. 5.10 Neuentwicklung von Individualsoftware: Anteil Fremdleistung in Anspruch genommene

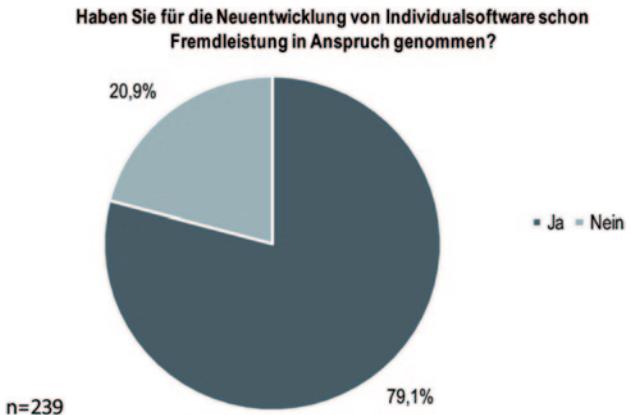
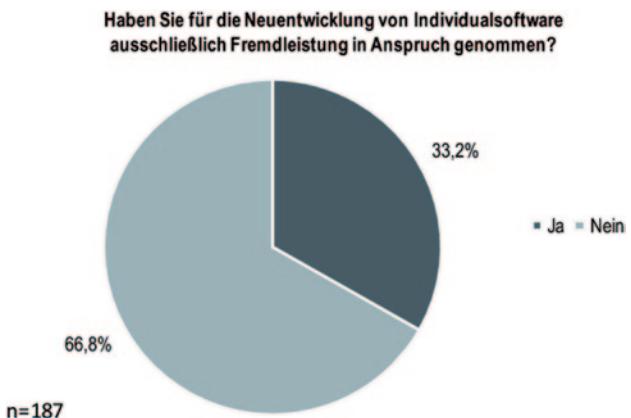


Abb. 5.11 Neuentwicklung von Individualsoftware: Anteil ausschließlich Fremdleistung in Anspruch genommen



Darüber hinaus haben wir zwischen geschäftskritischen und nicht-geschäftskritischen Prozessen unterschieden (Brandt 2010). Die Ergebnisse sind in den Abb. 5.12 und 5.13 dargestellt.

Kriterien für die Auswahl von Softwareanbietern zur Neuentwicklung von Individualsoftware Im Weiteren wurden die Studienteilnehmer befragt, welche Kriterien sie im Rahmen der Partnerwahl für die Neuentwicklung von Individualsoftware anlegen. Diese Frage haben wir lediglich an die Unternehmen gerichtet, die bereits Fremdleistung in Anspruch genommen haben. Die Unternehmen wurden gefragt, für wie wichtig sie die einzelnen Kriterien halten (1 = „sehr unwichtig“ ... 7 = „sehr wichtig“). Abbildung 5.14 zeigt ein Ranking dieser Kriterien auf Basis der Mittelwerte. Demnach ist den antwortenden Unternehmen eine langfristige Geschäftsbeziehung (Kontinuität) mit den Softwarehäusern am wichtigsten.

Ein Grund hierfür liegt darin, dass Anwender den Softwarehäusern im Falle einer langfristigen Zusammenarbeit häufig auch die Weiterentwicklung und Wartung der Softwarelösungen übertragen. Referenzen über erfolgreiche Realisierungen auf ähnlichem Arbeits-

Inwieweit stimmen Sie für die Neuentwicklung von Individualprogrammsystemen folgenden Aussagen zu?
(Arbeitsteilige Zusammenarbeit, Verantwortung bleibt im Unternehmen)

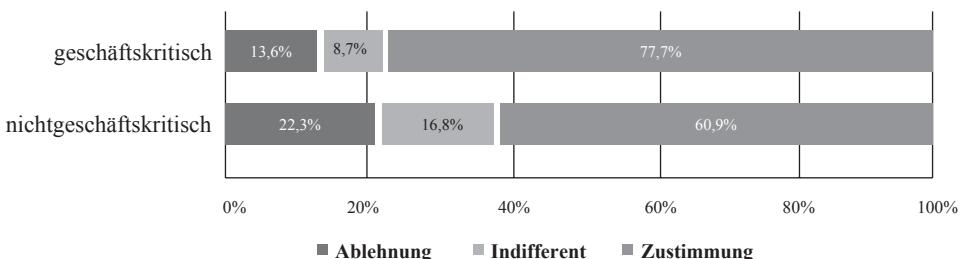


Abb. 5.12 Optionen der Inanspruchnahme von Fremdleistung. (Verantwortung bleibt im Unternehmen)tc

Inwieweit stimmen Sie für die Neuentwicklung von Individualprogrammsystemen folgenden Aussagen zu?
(Projekt an leistungsfähiges Softwarehaus vergeben, Verantwortung übertragen)

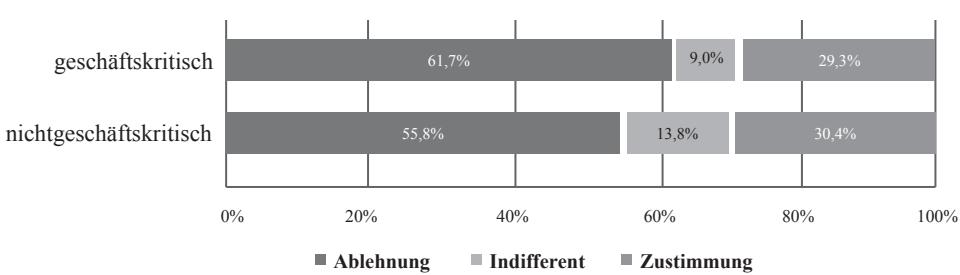


Abb. 5.13 Optionen der Inanspruchnahme von Fremdleistung. (Verantwortung wird übertragen)



Abb. 5.14 Ranking der Kriterien bei der Auswahl externer Dienstleister für die Entwicklung von Individualsoftware. (1 = sehr unwichtig, ..., 7 = sehr wichtig)

gebiet sind gemäß der Antworten der Unternehmen am zweitwichtigsten. Auf dem dritten Platz landete die Qualität der Mitarbeiter des Anbieters, die dem Unternehmen präsentiert werden. Gemäß der Angaben der Unternehmen spielt die Reputation des Softwarehauses ebenfalls eine wichtige Rolle. Der in diesem Zusammenhang berechnete Mittelwert von 5,11 zeigt zwar, dass ein gutes Preis-/Leistungsverhältnis eher wichtig ist, aber bei der Auswahl externer Dienstleister offenbar nicht als vorrangig bewertet wird. Ferner äußerten sich die Unternehmen indifferent bezüglich des Kriteriums, für die Neuentwicklung von Individualsoftware Softwarehäuser zu beauftragen, die sich „auf gleicher Augenhöhe“ befinden. Dahinter steckt die Frage, ob kleine und mittelständische Anwender tendenziell auch Anbieter ihrer Größenordnung bevorzugen.

5.6.2 Outsourcing der Anpassung von Standardsoftware

Grad der Inanspruchnahme von Fremdleistungen In diesem Abschnitt soll das Outsourcing der Anpassung von Standardsoftware näher untersucht werden. Dabei haben wir zwischen den folgenden Alternativen unterschieden (Buxmann et al. 2010):

- Customizing/Parametrisierung,
- Add-on-Programmierung und
- Veränderungen am Quelltext.

Im Rahmen eines so genannten Customizing bzw. einer Parametrisierung haben die Anwender die Möglichkeit, die Software in einem relativ geringen Umfang an die eigenen betrieblichen Anforderungen anzupassen. Hierzu sind keine umfassenden Programmier-

kenntnisse notwendig. Ein Customizing wird im Rahmen fast jeder Einführung einer betriebswirtschaftlichen Standardsoftware durchgeführt.

Häufig reicht ein Customizing der Standardsoftware jedoch nicht aus, um die individuellen Anforderungen eines Unternehmens zufriedenstellend abzubilden. Dies gilt insbesondere für geschäftskritische bzw. wettbewerbsdifferenzierende Geschäftsabläufe. Im Rahmen einer so genannten Add-on-Programmierung werden zusätzliche Funktionen entwickelt, die in die Standardsoftware integriert werden können. Häufig sind hierzu in den Standardlösungen auch entsprechende Schnittstellen vorgesehen. Hierbei wird die Standardsoftware nicht modifiziert.

Eine dritte Alternative besteht in der Modifikation des Quellcodes der Standardlösung. Voraussetzung ist dabei natürlich, dass der Softwarehersteller den Anwendern den Quelltext zur Verfügung stellt. Dabei besteht ein Vorteil des Customizing sowie der Add-on-Programmierung darin, dass die vorgenommenen unternehmensspezifischen Parametrisierungen und Ergänzungen bei einem Releasewechsel erhalten bleiben. Bei substantiellen Änderungen des Standardsystems ist dies demgegenüber häufig nicht möglich.

Etwa drei Viertel der Befragten haben für die Einführung von Standardsystemen Fremdleistung in Anspruch genommen (siehe Abb. 5.15). Ca. 70 % dieser Unternehmen führten an, nicht ausschließlich externe Dienstleister beauftragt zu haben (siehe Abb. 5.16). Im Gegensatz dazu haben knapp 30 % dieser Unternehmen für die Einführung eines Standardsystems oder Komponenten davon ausschließlich Fremdleistungen genutzt.

Die Analyse zeigt, dass Customizing und Parametrisierung die weitaus am häufigsten gewählte Anpassungsform ist. Darauf folgt die Add-on-Programmierung, während eine Veränderung des Quelltextes eher selten durchgeführt wird.

Abb. 5.15 Entwicklungsarbeiten/Anpassungen an Standardsoftware durch Fremdleistung

Haben Sie für Entwicklungsarbeiten an Ihrer Standardsoftware im Rahmen des Ersteinsatzes Fremdleistung in Anspruch genommen, oder beabsichtigen Sie dies für die Einführung weiterer Systemkomponenten?

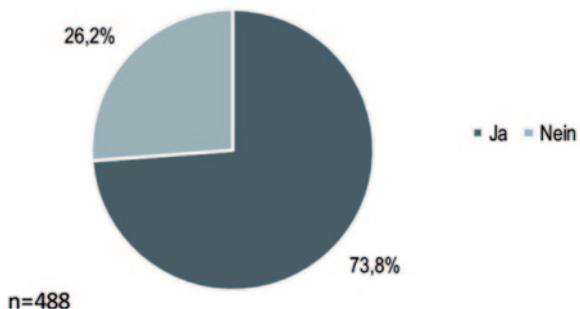
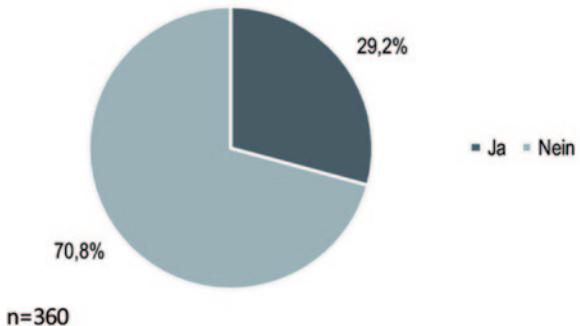


Abb. 5.16 Entwicklungsarbeiten/Anpassungen an Standardsoftware ausschließlich durch Fremdleistung

Haben Sie für Entwicklungsarbeiten an Ihrer Standardsoftware im Rahmen des Ersteinsatzes ausschließlich Fremdleistung in Anspruch genommen?



Kriterien für die Auswahl von Softwareanbietern Analog zur Evaluierung von Individualsoftwareherstellern wurden die Teilnehmer gefragt, welche Kriterien sie zur Auswahl von Standardsoftwareanbietern heranziehen (Brandt 2010). Abbildung 5.17 zeigt ein Ranking dieser Kriterien, das erneut auf einem Mittelwertvergleich der Einschätzungen der Teilnehmer basiert (1=„sehr unwichtig“, ..., 7=„sehr wichtig“). Demnach stehen die Abdeckung der betriebswirtschaftlichen Funktionalität und die Branchenerfahrung auf dem ersten Platz der Prioritätenliste der Anwender. Die Sicherheit auf eine dauerhafte Geschäftsbeziehung (Kontinuität) wurde als am zweitwichtigsten bewertet. Das vorhandene Know-how zur Unterstützung bei der Einführung von Standardsoftware wurde von den Teilnehmern, die Standardsoftware einsetzen, ebenfalls als wichtig beurteilt. Die Skalierbarkeit und die Erweiterungsmöglichkeiten des einzusetzenden Standardsoft-



Abb. 5.17 Ranking der Kriterien zur Auswahl eines Anbieters von Standardsoftware

wareproduktes sind gemäß den Antworten der Unternehmen ebenfalls wichtig. Als „eher wichtig“ kann die Bewertung der weiten Verbreitung der Standardsoftware eingeordnet werden. Für die Bewertung des Preis-/Leistungsverhältnisses ergibt sich ein analoges Bild zur Auswahl von Individualsoftware-Anbietern: Auch für die Auswahl eines Anbieters von Standardsoftware bewerten die Unternehmen das Preis-/Leistungsverhältnis nicht als vorrangig bzw. überaus wichtig relativ zu den anderen Auswahlkriterien. Der berechnete Mittelwert von 5,1 zeigt zwar, dass ein gutes Preis-/Leistungsverhältnis eher wichtig ist, die Bedeutung dieses Kriteriums im Vergleich zu den anderen genannten jedoch nicht als vorrangig eingestuft wird. Die Reputation des Softwareanbieters wurde unter den im Erhebungsinstrument angegebenen Kriterien zur Auswahl von Standardsoftwareanbietern mit einem Mittelwert von 4,86 als am wenigsten wichtig bewertet.

5.6.3 Outsourcing der Weiterentwicklung und Wartung von Anwendungssoftware

Grad der Inanspruchnahme von Fremdleistungen Auch in Bezug auf die Weiterentwicklung und Wartung von Anwendungssoftware wurde untersucht, ob die Anwender für diese Leistungen vollständig oder teilweise Fremdleistung in Anspruch genommen haben (Brandt 2010). Dabei gaben über 85 % der befragten Unternehmen an, für die Weiterentwicklung und Wartung der eingesetzten Anwendungssoftware vollständig oder teilweise externe Dienstleister beauftragt zu haben (siehe Abb. 5.18). Demgegenüber erledigt eine Minderheit diese Aufgaben ausschließlich selbst.

Der weitaus größte Teil (72,6 %) derjenigen Befragten, die für die Weiterentwicklung und Wartung der Anwendungssysteme Fremdleistung genutzt haben, gab an, nicht ausschließlich externe Dienstleister beauftragt, sondern diese vielmehr in Ergänzung zur

Abb. 5.18 Weiterentwicklung und Wartung: Anteil Fremdleistung in Anspruch genommene

Haben Sie für die Weiterentwicklung/Wartung Ihrer IT-Landschaft schon ganz oder teilweise Fremdleistung (z.B. Beratungsleistung) in Anspruch genommen?

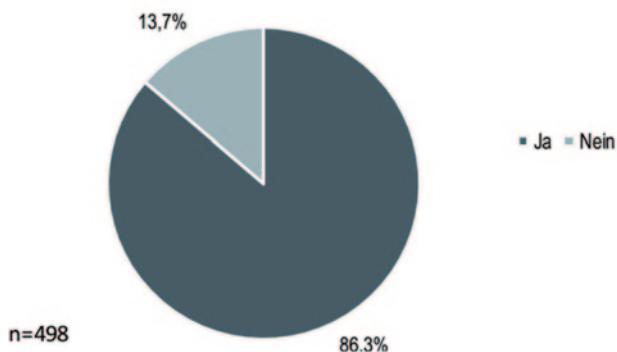


Abb. 5.19 Weiterentwicklung und Wartung: Anteil ausschließlich Fremdleistung in Anspruch genommene



internen Leistungserbringung eingesetzt zu haben (siehe Abb. 5.19). Immerhin gut ein Viertel dieser Unternehmen hat ausschließlich externe Software- und Serviceanbieter beschäftigt.

Im Weiteren wurde gefragt, ob und inwieweit die Bereitschaft des Fremdbezugs der Weiterentwicklung und Wartung von Anwendungssoftware davon abhängt, ob die Softwarelösung geschäftskritische oder weniger kritische Geschäftsprozesse unterstützt. Die Ergebnisse zeigen, dass die Bereitschaft zum Outsourcing bei weniger geschäftskritischen Prozessen deutlich höher ist als bei Kerngeschäftsprozessen (siehe Abb. 5.20).

Dabei zeigen die Ergebnisse auch, dass ein „Backsourcing“, d. h. das Zurückholen bereits ausgelagerter Prozesse im Bereich der Weiterentwicklung und Wartung von Anwendungssoftware, bislang noch relativ selten stattfindet. So geben über 80% der befragten Unternehmen an, die bereits ausgelagerte Weiterentwicklung und Wartung von Anwendungssystemen noch nicht wieder in die Verantwortung des Unternehmens „zurückgeholt“ zu haben (siehe Abb. 5.21). Lediglich knapp 3 % der Befragten haben bereits ausgelagerte Weiterentwicklung und Wartung von Anwendungssystemen wieder in die Verantwortung des Unternehmens übertragen. Allerdings stehen 15 % der Unternehmen vor der Entscheidung, die Weiterentwicklung und Wartung eines kompletten Anwendungssystems (wieder) mit eigenen Mitarbeitern durchzuführen.

Inwieweit sind Sie bereit, die Weiterentwicklung und Wartung kompletter Anwendungssysteme an Fremdleister auszulagern?

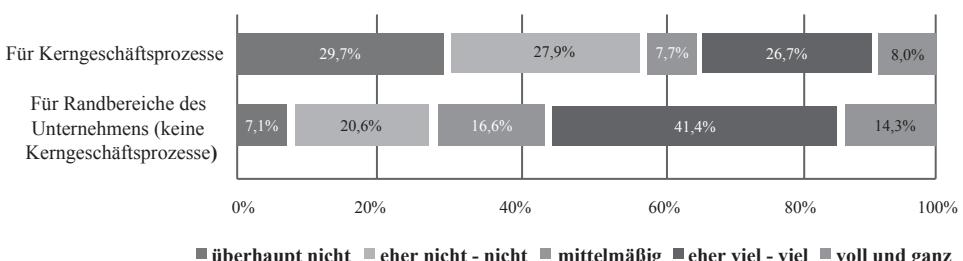
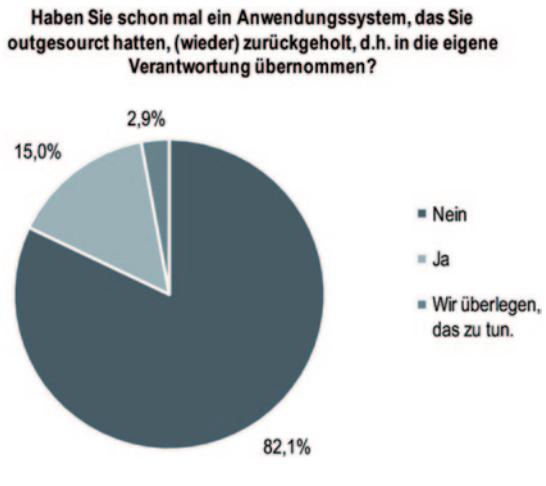


Abb. 5.20 Outsourcing von Weiterentwicklung und Wartung kompletter Anwendungssysteme tc

Abb. 5.21 Backsourcing von Weiterentwicklung und Wartung kompletter Anwendungssysteme



Kriterien für die Auswahl von Softwareanbieter Im Weiteren wurden die Studienteilnehmer befragt, welche Kriterien sie im Rahmen der Partnerwahl für die Weiterentwicklung und Wartung der Anwendungssoftware anlegen (Brandt 2010). Diese Frage wurde nur an die Unternehmen adressiert, die hierfür Fremdleistung von Software- oder Serviceanbietern in Anspruch genommen haben. Abbildung 5.22 zeigt das Ranking dieser Kriterien.

Demnach ist den antwortenden Unternehmen eine langfristige Geschäftsbeziehung (Kontinuität) mit dem Software- und Serviceanbieter am wichtigsten. Dies liegt vermut-

Inwieweit stimmen Sie folgenden Aussagen zur Wahl von IT-Dienstleistern (auch Berater) für die längerfristige Zusammenarbeit bei der Weiterentwicklung und Wartung Ihrer IT-Landschaft zu?

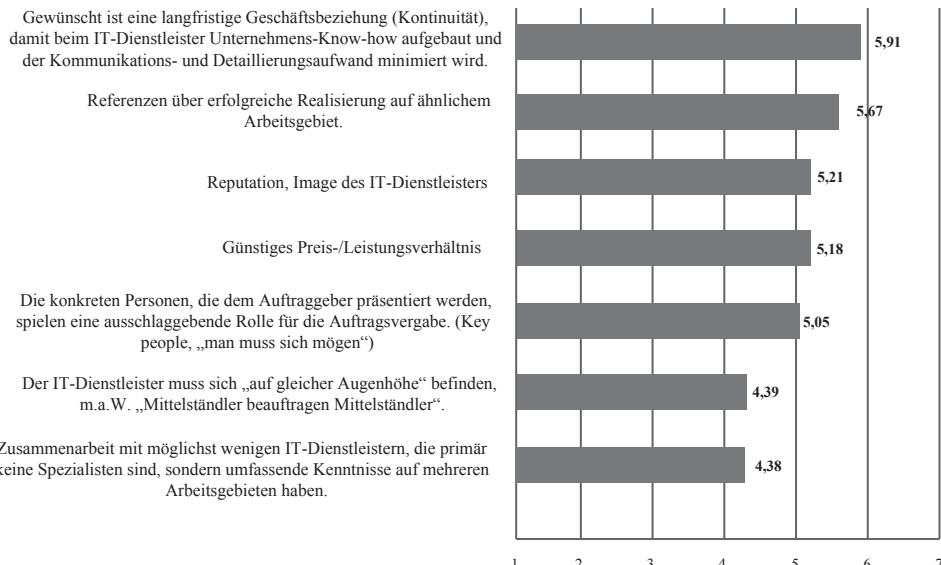


Abb. 5.22 Ranking der Kriterien zur Auswahl eines Software- oder Serviceanbieters bei der Weiterentwicklung und Wartung

lich daran, dass im Falle einer langfristig angelegten Zusammenarbeit beim IT-Dienstleister Unternehmens-Know-how aufgebaut und damit der Kommunikationsaufwand im Zusammenhang mit der Weiterentwicklung und Wartung minimiert wird. Referenzen über erfolgreiche Realisierungen auf ähnlichem Arbeitsgebiet sind gemäß den Antworten der Unternehmen am zweitwichtigsten. Die Reputation des Software- oder Serviceanbieters wurde als eher wichtig bewertet, ebenso ein günstiges Preis-/Leistungsverhältnis. Dass die Mitarbeiter, die dem Auftraggeber vom Serviceanbieter präsentiert werden, eine ausschlaggebende Rolle für die Auftragsvergabe spielen, bewerteten die Unternehmen ebenfalls als eher wichtig. Als indifferent bis eher wichtig schätzten die Teilnehmer das Kriterium, für die Weiterentwicklung und Wartung der bestehenden IT-Landschaft Software- oder Serviceanbieter zu beauftragen, die sich „auf gleicher Augenhöhe“ befinden, ein. Als ebenfalls indifferent bis eher wichtig erachteten die Studienteilnehmer, die für die Weiterentwicklung und Wartung Fremdleistung nutzten, dass die Zusammenarbeit mit möglichst wenigen IT-Dienstleistern erfolgen sollte, die primär keine Spezialisten sind, sondern umfassende Kenntnisse auf mehreren Arbeitsgebieten haben.

Ort der Leistungserbringung Aufgrund der besonderen Bedeutung, die Weiterentwicklung und Wartung im Softwarelebenszyklus spielen, haben wir die Unternehmen darüber hinaus gefragt, ob sie spezifische Vorteile der Leistungserstellung vor Ort sehen. In diesem Zusammenhang stimmten 72,8% der Unternehmen, die für die Weiterentwicklung und Wartung Fremdleistung in Anspruch genommen haben, der Aussage zu, dass Partner vor Ort extrem wichtig sind (siehe Abb. 5.23). Sogar 81,9% der Befragten gaben an, inländische Onshore-Anbieter zu bevorzugen. Darüber hinaus waren 75,9% der Unternehmen, die für die Weiterentwicklung und Wartung Fremdleistung in Anspruch genommen haben, der Meinung, dass es für das Unternehmen Zusatzaufwand bedeutet, die Aufgaben für Near- oder Farshoring-Dienstleister im Ausland zu spezifizieren und abzugrenzen.

Nach den obigen Ausführungen ergibt sich eine Tendenz zugunsten des Onshorings für die Weiterentwicklung und Wartung von Anwendungssoftware.

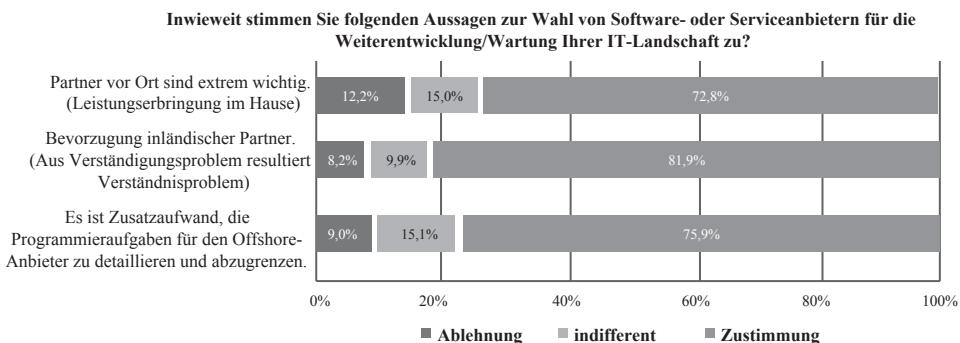


Abb. 5.23 Ort der Leistungserbringung für die Weiterentwicklung und Wartung der IT-Landschaft tc

5.6.4 Zufriedenheit der Anwender mit Onshore-, Nearshore- und Farshoreanbietern

Im folgenden Abschnitt wollen wir untersuchen, ob und inwieweit die räumliche Entfernung zwischen Anbieter und Kunden einen Einfluss auf den Erfolg von Outsourcing-Projekten hat (Buxmann et al. 2010).

In Abb. 5.24 sind die Antwortprofile zur Inanspruchnahme von Fremdleistung aus On-, Near- und Farshorelokationen für die Neuentwicklung von Individualsoftware, die Anpassung von Standardsoftware sowie die Weiterentwicklung und Wartung von Anwendungssoftware vergleichend dargestellt. Interessanterweise ergibt sich ein nahezu identisches Antwortverhalten.

Wie zufrieden sind die Anwender nun mit den Leistungen der Outsourcing-Anbieter? Um dies herauszufinden, haben wir die Unternehmen nach ihren Erfahrungen in Onshore-, Nearshore- und Farshore-Projekten befragt. Dabei interessierten wir uns insbesondere für die Zufriedenheit in Bezug auf Produkt- und Servicequalität, Kosten, Termintreue sowie Kommunikation und Koordination. Die jeweiligen Durchschnittswerte sind in der

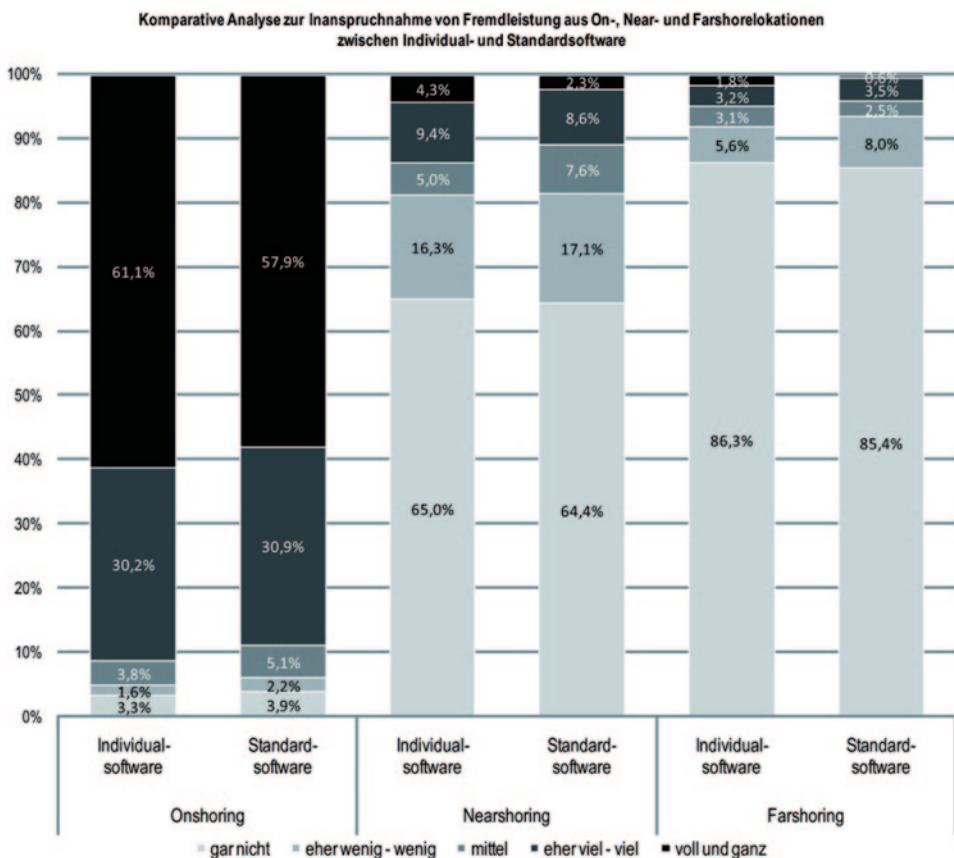


Abb. 5.24 Komparative Analyse zur Inanspruchnahme von Fremdleistung aus Onshore-, Nearshore- und Farshorelokationen zwischen Individual- und Standardsoftware

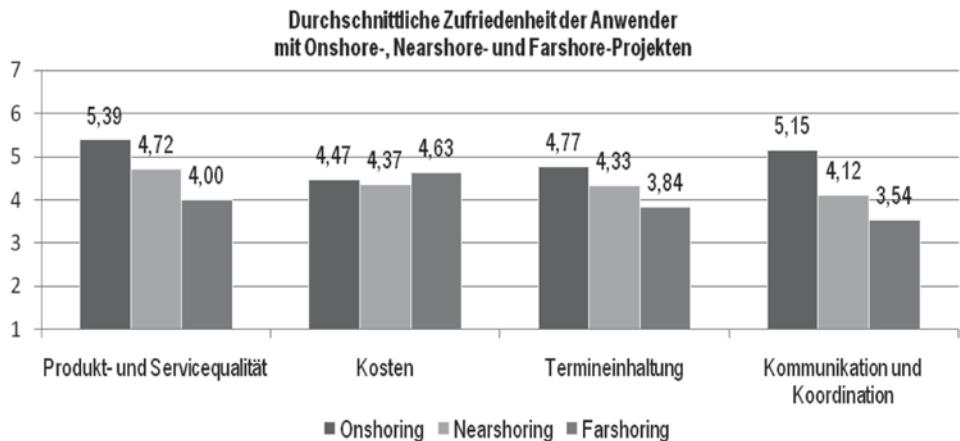


Abb. 5.25 Komparative Analyse der Zufriedenheit von Anwendern mit Fremdleistungen aus Onshore-, Nearshore- und Farshorelokationen

folgenden Abbildung dargestellt. Dabei wurden jeweils nur diejenigen Unternehmen der Stichprobe berücksichtigt, die bereits Erfahrungen mit Onshore-, Nearshore- bzw. Farshore-Projekten gesammelt haben. Im Vergleich zur gesamten Stichprobe zeigt sich, dass es sich hierbei tendenziell um die größeren Unternehmen handelt (siehe Abb. 5.25).

Die Ergebnisse zeigen, dass die Unternehmen im Hinblick auf Produkt- und Servicequalität, Termintreue sowie Kommunikation/Koordination mit den Onshore-Dienstleistern am zufriedensten waren. Am zweitbesten schnitten die Software- und Serviceanbieter in Nearshore-Ländern ab, während die Kunden mit den Farshore-Anbietern in Bezug auf diese Kriterien am unzufriedensten waren. Die Ergebnisse sind statistisch signifikant. Demgegenüber ergaben sich keine signifikanten Differenzen in Bezug auf die Zufriedenheit mit den Kosten.

Eine mögliche Erklärung hierfür ist, dass die Unternehmen die große Distanz zu den Farshoring-Anbietern für problematisch halten. Dabei geht es im Kern um die Frage der Kommunikation zwischen Anbieter und Kunden, die durch die räumliche, sprachliche und kulturelle Distanz unter Umständen erheblich erschwert wird, was in der Regel zu schlechteren Projektergebnissen bzw. längeren Projektlaufzeiten führt. Die möglichen Kommunikationsprobleme können hierbei grundsätzlich in sprachliche und kulturelle Probleme unterschieden werden, auf die wir im Folgenden eingehen.

5.7 Nearshoring versus Farshoring: Die Entfernung zum Kunden als Erfolgsfaktor?

5.7.1 Sprachliche und kulturelle Barrieren in Offshore-Projekten

Das Verständnis für den Umgang mit Menschen, die aus einem anderen Kulturreis kommen, ist einer der in Studien am häufigsten genannten Erfolgsfaktoren für Offshore-

Projekte (Buxmann et al. 2010; Gregory 2010). Auch alle von uns befragten Experten betonten die Gefahr, dass kulturelle und sprachliche Probleme zu Missverständnissen, Vertrauensverlust und zu einer sinkenden Arbeitsmotivation führen können.

Aufgrund der herausragenden Stellung Indiens als Offshore-Standort wird besonders häufig auf die kulturellen Differenzen zwischen Indern einerseits und Europäern oder Amerikanern andererseits eingegangen. In diesem Zusammenhang spielen unterschiedliche Hierarchieverständnisse sowie das Bestreben, schlechte Nachrichten zu vermeiden, eine besondere Rolle. So neigen indische Mitarbeiter häufig dazu, sich auch für kleinere Entscheidungen das Einverständnis ihres Vorgesetzten einzuholen, und äußern nicht immer ihre Meinung – insbesondere nicht in Anwesenheit der Projektleiter. Es fällt ihnen darüber hinaus oftmals schwer, „nein“ zu sagen, wenn dies eigentlich notwendig wäre. Zudem werden Probleme eher ungern zugegeben und als Gesichtsverlust angesehen. Es liegt auf der Hand, dass in diesem Fall eine Frage wie: „Können wir damit rechnen, dass der Prototyp wie geplant fertig wird?“ zu Problemen im Projekt führen kann.

Solche Unterschiede können allerdings durchaus auch an anderen Offshore-Standorten auftreten. Delmonte und McCarthy beschreiben in ihrer Offshoring-Untersuchung beispielsweise die Zurückhaltung russischer Entwickler, bei Verständnisproblemen nachzufragen. Dadurch bleibt die Kundenseite im Glauben, dass die geäußerten Wünsche verstanden worden sind, obwohl dies nicht unbedingt der Fall sein muss (Delmonte und McCarthy 2003, S. 1609).

Kulturelle Unterschiede bestehen jedoch nicht nur bei großen Entfernung. Ein Experte berichtet von seinen in den Niederlanden gesammelten Erfahrungen. Dort ist der Umgang miteinander in der Regel weniger formell als in Deutschland und mündliche Absprachen sind im Allgemeinen nicht weniger verbindlich als schriftlich festgehaltene. Die von einem deutschen Mitarbeiter geäußerte Bitte, eine soeben mündlich getroffene Vereinbarung auch schriftlich zu formulieren, wurde von niederländischer Seite somit als Misstrauen verstanden und hat dadurch wiederum Argwohn erzeugt.

Insbesondere die Experten der untersuchten Nearshore-Unternehmen legen Wert darauf, dass die kulturellen Unterschiede zu ihren deutschen Kunden vergleichsweise gering sind. Im Unterschied zu Anbietern aus Indien oder China gäbe es deutlich mehr Gemeinsamkeiten, was beispielsweise den offenen Umgang miteinander betrifft.

Für eine reibungslose Kommunikation ist es selbstverständlich auch notwendig, dass die beteiligten Personen über entsprechende Sprachkenntnisse verfügen. Doch selbst wenn dies der Fall ist, bestehen potenzielle Fehlerquellen in der unterschiedlichen Aussprache, den Bedeutungsunterschieden von Wörtern sowie den Differenzen in den Fachvokabularen, die auch auf die jeweiligen Kulturkreise zurückzuführen sind.

Dabei wird die Tatsache, dass Offshoring und insbesondere Farshoring in Deutschland bislang weniger verbreitet ist als in Großbritannien oder den USA, häufig auch auf Sprachbarrieren zurückgeführt. Hier ist Deutsch – obwohl sich dies insbesondere in Großunternehmen langsam aber sicher ändert – nach wie vor die Arbeitssprache in den meisten Unternehmen. Es ist auch nachvollziehbar, dass die Mitarbeiter deutscher Kunden (wenn es sich nicht gerade um „native speaker“ handelt) es in der Regel vorziehen, in deutscher

Sprache zu kommunizieren. Daher sollte die Interaktion mit den meisten deutschen Kunden nicht nur bei der Entwicklung, sondern auch beim First Level Support nach Möglichkeit in deutscher Sprache erfolgen. Die meisten der von uns befragten Experten messen deshalb der Sprachkompetenz eine herausragende Bedeutung bei. Auch aus diesem Grund werden von globalen Playern zunehmend Onshore- und Nearshore-Standorte eingerichtet und lokale Mitarbeiter eingestellt.

Bei einer Untersuchung des Zusammenhangs von Outsourcing-Verträgen und dem Erfolg der jeweiligen Projekte haben Willcocks und Lacity herausgefunden, dass Projekte auf der Basis detaillierter Verträge tendenziell erfolgreicher sind als solche mit weniger eng formulierten Verträgen (Willcocks und Lacity 2006, S. 20 f.). Angesichts der beim Offshoring auftretenden Sprachbarrieren ist es hier bei den Verträgen offenbar besonders wichtig, den Interpretationsspielraum möglichst gering zu halten.

Nachdem fast alle Experten die Bedeutung kultureller sowie sprachlicher Unterschiede betonen, stellt sich die Frage, wie die Unternehmen dieser Herausforderung begegnen. Eine weit verbreitete Maßnahme ist die Durchführung interkultureller Seminare, und zwar sowohl für die Mitarbeiter an den Offshore-Standorten als auch für die Onshore-Mitarbeiter. Teilweise bieten Offshore-Anbieter auch ihren Kunden entsprechende Seminare an.

Darüber hinaus stellt die Vermittlung von Wissen durch erfahrene Mitarbeiter einen wichtigen Beitrag zur kulturellen Sensibilisierung dar. Dazu werden von einigen Anbietern Mitarbeiter zwischen den Standorten ausgetauscht. Ein Experte berichtet von positiven Erfahrungen, die sein Unternehmen damit gemacht habe, immer zwei Mitarbeiter bei den Projekten aus Indien nach Deutschland zu holen. Neben vielen anderen positiven Auswirkungen habe es sich gezeigt, dass diese Mitarbeiter nach ihrer Rückkehr weitaus engagierter arbeiten würden als zuvor. Diese Beobachtung wurde auch von anderen Experten gemacht. Allerdings wird häufig darauf Wert gelegt, dass indische Mitarbeiter, die zum Kunden geschickt werden, die westliche Unternehmenskultur zumindest etwas kennen.

Ein Experte berichtet, dass er seinen Kunden immer vorschlägt, auch einen Mitarbeiter nach Indien zu schicken. Dadurch bekommen die Kunden ein besseres Gefühl der Kontrolle und bauen gleichzeitig Kompetenzen im interkulturellen Umgang auf. Viele Kunden nehmen diesen Vorschlag positiv auf.

Eine dritte Maßnahme zur Überwindung kultureller Barrieren ist das Einstellen lokaler Mitarbeiter. Ein Experte berichtet beispielsweise, dass selbst für Projekte in Österreich oder in der Schweiz nach Möglichkeit Österreicher beziehungsweise Schweizer in das Team integriert würden, da diese Feinheiten im Umgang mit den Kunden viel eher bemerken als etwa ein deutscher Mitarbeiter. Insbesondere Schnittstellenfunktionen werden von lokalen Mitarbeitern erfüllt. Nahezu alle Offshore-Anbieter beschäftigen in Projekten in Deutschland deutsche Mitarbeiter, die oftmals auch die Projektverantwortung innehaben und erste Ansprechpartner für die Kunden sind.

Im Folgenden wollen wir die Notwendigkeit persönlicher Treffen näher beleuchten und untersuchen, inwieweit diese möglicherweise zu einer verbesserten Kommunikation zwischen Anbieter und Kunde führen.

5.7.2 Die Bedeutung persönlicher Treffen für den Projekterfolg

Insbesondere in der Anfangsphase eines Projekts sind persönliche Treffen zwischen Mitarbeitern des Kunden und des Softwareanbieters von großer Bedeutung für den Projekterfolg. Dabei stimmen die meisten der von uns befragten Experten der Feststellung zu, dass manche Teile der Softwareentwicklung besser am Standort des Kunden durchgeführt werden sollten. Allerdings sind nur die großen global aufgestellten Anbieter, die auch über Entwicklungszentren in Nearshore-Ländern verfügen und mehrere Offshore-Standorte betreiben, in der Lage, die Aufteilung der Prozesse so feingranular zu betreiben. Von den interviewten Experten gaben die meisten eine Aufteilung der Phasen an, wie sie in der folgenden Abb. 5.26 dargestellt ist.

Die in der Abbildung dargestellten Abläufe sind lediglich als Tendenzaussage zu verstehen. Für den Projekterfolg ist dabei von großer Bedeutung, dass die verschiedenen Phasen nicht von völlig unterschiedlichen Projektteams durchgeführt werden. Vielmehr wird von allen befragten Experten die Notwendigkeit einer engen Integration im gesamten Projektverlauf betont.

Dabei erreichen zahlreiche Offshore-Anbieter eigenen Aussagen zu Folge Onsite-Quoten von etwa 30 %. Viele Experten legen darauf Wert, dass das Onsite-Team während des gesamten Projekts fortbesteht und nie ganz aufgelöst wird.

Die im linken Teil von Abb. 5.26 dargestellten Phasen sind sehr kommunikationsintensiv und sollten daher in der Regel am Standort des Kunden und nicht offshore durchgeführt werden. Auch moderne Kommunikationsmittel, wie beispielsweise Videokonferenzen, können persönliche Kommunikation nicht ohne Weiteres ersetzen. So ist zu beob-

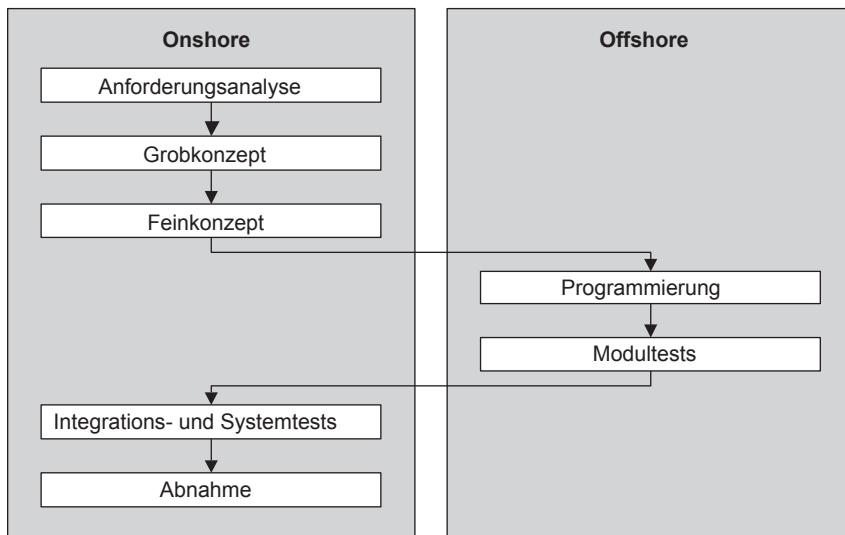


Abb. 5.26 Aufteilung der Entwicklungsphasen onshore und offshore

achten, dass Videokonferenzen bei vielen deutschen Unternehmen – im Gegensatz zu den Anbietern – immer noch etwas Besonderes sind. Beispielsweise berichtet ein Experte von einem Fall, in dem eine Abstimmung zwischen Kunden und Anbieter dringend zeitnah notwendig gewesen wäre. Dennoch wurde die Videokonferenz aufgrund organisatorischer Schwierigkeiten und dem Tatbestand, dass es kein Standardprozess war, erst für einige Tage später angesetzt, wodurch letztlich viel Zeit verloren ging.

Wenn wir jedoch davon ausgehen, dass solche Kommunikationsformen zukünftig überall zur Selbstverständlichkeit werden, können Videokonferenzen persönliche Treffen zumindest teilweise ersetzen und die Tendenz zum Offshoring verstärken.

Zwangsläufig erhöht sich mit zunehmender Entfernung auch die Reisezeit zwischen Anbieter und Kunde. Dies wird grundsätzlich dazu führen, dass die Kunden tendenziell seltener einen Farshore- als einen Nearshore-Standort aufsuchen. Ob Maßnahmen wie Online-Dokumentationen des Projektstatus aus Sicht der Kunden ausreichen, dieses Defizit zu beheben, ist sicherlich kunden- und projektabhängig. Generell gilt aber, dass die Mitarbeiter des Anbieters deutlich stärker von den Reisetätigkeiten betroffen sind als die Kunden.

Letztlich sind die kulturellen Distanzen zwischen Anbieter und Kunde von größerer Bedeutung für den Projekterfolg als die räumliche Entfernung. Ist ein Projektteam erst einmal örtlich verteilt, erschwert das die Kommunikation und Koordination und die meisten der von uns befragten Experten bestätigten die These, dass die Kommunikations- und Koordinationskosten in einem Entwicklungsprojekt nicht proportional mit der räumlichen Entfernung ansteigen.

5.7.3 Herausforderungen und Chancen der Zeitverschiebung

Eine Zeitverschiebung zwischen Kunden und Anbietern aus Offshore-Ländern führt zunächst zu möglichen Problemen bei allen Formen der synchronen Kommunikation, wie beispielsweise Telefon- oder Videokonferenzen. Je größer die Zeitverschiebung, desto kleiner ist das Zeitfenster hierfür.

Für Offshore-Anbieter lässt sich das Problem zum einen durch verlängerte Arbeitszeiten an den Offshore-Standorten und zum anderen durch eine vorausschauende Planung der Kommunikation lösen. Dabei betonen einige unserer Experten, dass insbesondere die Mitarbeiter in Indien in der Regel geringe Probleme mit längeren und ungewohnten Arbeitszeiten haben.

Neben diesen potenziellen Kommunikationsproblemen bietet die Zeitverschiebung auch die Chance, nach dem in Abschn. 5.3 angesprochenen Follow-the-Sun-Prinzip zu arbeiten. Dieses konnte beispielsweise in einigen Fällen bei amerikanischen Kunden erfolgreich umgesetzt werden. Tagsüber wurden in den USA Anforderungen gesammelt, die abends nach Indien kommuniziert und dort in der Nacht prototypisch umgesetzt wurden. Am nächsten Morgen konnten die Ergebnisse am US-Standort weiterverarbeitet werden.

Auf diese Weise lassen sich grundsätzlich Release-Zyklen sowie die Entwicklungszeiten verkürzen – auf Kosten höherer Koordinationskosten.

Dabei ist aus europäischer Sicht jedoch zu berücksichtigen, dass die Zeitverschiebung zu Indien zwischen 3,5 h im Sommer und 4,5 h im Winter pendelt, was natürlich die möglichen Vorteile des Follow-the-Sun-Prinzips etwas einschränkt. Ein Beispiel für die Ausnutzung der Zeitverschiebung sind die Abnahmetests der Software AG in Indien (wir sind auf die Gründung des Joint Ventures in Pune bereits in Abschn. 5.2 eingegangen):

Softwaretests der Software AG in Indien

Für die Entwicklung von Lösungen im SOA-Umfeld unterhält die Software AG ein Testteam in Pune, Indien. Das Team soll sich wie Kunden verhalten, die die Lösungen einsetzen möchten, und ist mit folgenden Aufgaben beauftragt:

- Installieren der Produkte,
- Koexistenz- und Integrationstests,
- Verifizieren der Interoperabilität der Produkte,
- Verifizieren der Basisfunktionen,
- Erstellung von Fehlerberichten sowie
- Nachtesten von Korrekturen.

Aufgabe des Teams ist es – neben dem Testen selbst – Testpläne zu erstellen, die Testaktivitäten zu koordinieren und zu überwachen, wöchentliche Ergebniszusammenfassungen zu liefern, Mitarbeiter vor Ort aus- bzw. weiterzubilden und die Kommunikation mit der Zentrale aufrecht zu halten.

Die Entwicklung, die weltweit organisiert ist, stellt in regelmäßigen Abständen vollständige Produktkits zur Verfügung, die von den indischen Mitarbeitern heruntergeladen und getestet werden. Die Kits werden nachts erstellt, damit die indischen Mitarbeiter morgens gleich testen können. Wenn der Arbeitstag an den Entwicklungsstandorten anfängt, liegen bereits erste Ergebnisse vor und es können Nachfragen beantwortet sowie etwaige Fehler zum Nachtesten behoben werden. Die Software AG schätzt die Einsparungen, die sich durch die Aufteilung der Prozesse auf die verschiedenen Standorte ergeben, auf etwa 50 % im Vergleich zu einer ausschließlichen Durchführung im Headquarter in Darmstadt. In dieser Schätzung sind Koordinationskosten bereits berücksichtigt worden.

Ein weiterer häufig genannter Vorteil einer Zeitverschiebung ist die Möglichkeit, Support rund um die Uhr anbieten zu können.

Auf die Frage, welche Voraussetzungen erfüllt sein müssen, um über mehrere Zeitzonen hinweg zu arbeiten, wurden von den Experten vornehmlich drei Faktoren genannt:

- sauber definierte Prozesse,
- Integration der Standorte und
- genaue Dokumentationen.

Literatur

- Amberg M, Wiener M (2006) IT-offshoring. Physika, Heidelberg
- Arkes HR, Blumer C (1985) The psychology of sunk cost. *Organ Behav Hum Decis Process* 35(1):124–140
- Boes A, Schwemmle M (2005) Bangalore statt Böblingen? VSA, Hamburg
- Brandt B (2010) Make-or-Buy bei Anwendungssystemen. Gabler, Wiesbaden
- Bräutigam P, Grabbe H (2004) Rechtliche Ausgangspunkte. In: Bräutigam P (Hrsg) IT-outsourcing. Erich Schmidt Verlag, Berlin, S 161–202
- Buxmann P, Brandt B, von Ahsen A, Hess T (2010) Outsourcing der Entwicklung und Anpassung von Anwendungssoftware: Analyse der Kundenzufriedenheit auf Basis einer empirischen Untersuchung, Darmstädter Arbeitspapier
- Delmonte AJ, McCarthy RV (2003) Offshore software development: is the benefit worth the risk?. In: Proceedings of the 2003 America's Conference on Information Systems (AMCIS 2003), Tampa (Florida), USA 4–6 Aug. 2003, pp 1607–1613
- Emerson RM (1962) Power-dependence relations. *Am Sociol Rev* 27(1):31–41
- Friedman T (2005) The world is flat: a brief history of the 21st century. Penguin, London
- Gadatsch A (2006) IT-Offshore realisieren. Vieweg, Wiesbaden
- Ganesan, S. (1994) Determinants of long-term orientation in buyer-seller relationships. *J Mark* 58(2):1–19
- Geyskens I, Steenkamp J-B, Scheer LK, Kumar N (1996) The effects of trust and interdependence on relationship commitment: a trans-Atlantic study. *Int J Res Mark* 13(4):303–317
- Gregory RW (2010) Review of the IT Offshoring literature: the role of cross-cultural differences and management practices. In: 18th European Conference on Information Systems, Manuscript ID: ECIS2010-0086.R1
- Grover V, Cheon MJ, Teng JTC (1996) The effect of service quality and partnership on the outsourcing of information systems functions. *J Manage Inf Syst* 12(4):89–116
- Gulati R, Sytch M (2007) Asymmetry and joint interorganizational relationships: effects of embeddedness on a manufacturer's performance in procurement relationships. *Adm Sci Q* 52(1):32–69
- Hackett Group (2012) <http://www.thehackettgroup.com/about/research-alerts-press-releases/2012/03272012-hackett-research-forecasts-offshoring.jsp>
- Heide JB, John G (1988) The role of dependence balancing in safeguarding transaction-specific assets in conventional channels. *J Mark* 52(1):20–35
- Hess T, Buxmann P, Mann F, Königer M (2007) Industrialisierung der Softwarebranche: Erfahrungen deutscher Anbieter. In: Management Reports des Instituts für Wirtschaftsinformatik und Neue Medien 2, München
- Hirschheim R, Heinzl A, Dibbern J (2006) Information systems outsourcing. Springer, Heidelberg
- Jacobs D (1974) Dependency and vulnerability: an exchange approach to the control of organizations. *Adm Sci Q* 19(1):45–59
- Jones MA, Mothersbaugh DL, Beatty SE (2002) Why customers stay: measuring the underlying dimensions of services switching costs and managing their differential strategic outcomes. *J Bus Res* 55(6):441–450
- Kaiser J, Buxmann P (2012) Organizational design of IT supplier relationship management: a multiple case study of five client companies. *J Inf Technol* 27(1):57–73
- Kaiser J, Widjaja T, Buxmann P (2012) Positioning clients in dyadic dependence structures of IS outsourcing relationships – conceptualization and empirical findings. In: International Conference on Information Systems (ICIS), Orlando, Florida, USA, 16–19 Dez. 2012
- Kearney AT (2004) Making offshore decisions. Offshore Location Attractiveness Index, Chicago

- Keil M, Truex DP, Mixon R (1995) The effects of sunk cost and project completion on information technology project escalation. *IEEE Trans Eng Manage* 42(4):372–381
- Kublanov EM, Satyaprasad S, Nambiyattil R (2005) Offshore & Nearshore ITO Salary Report 2004, vol 3, neoIT, San Ramon. http://www.neoit.com/pdfs/whitepapers/OIv3i05_0505_ITOSalaries2004.pdf
- Kumar N, Scheer LK., Steenkamp J-B (1995) The effects of perceived interdependence on dealer attitudes. *J Mark Res* 32(3):348–356
- Lacity MC, Khan SA, Willcocks LP (2009) A review of the IT outsourcing literature: insights for practice. *J Strateg Inf Syst* 18(3):130–146
- Lünendonk T (2013) TOP 25 der Standard-Software-Unternehmen in Deutschland 2012. http://luenendonk.de/wp-content/uploads/2013/05/LUE_Liste_u_PI_2013_Standard_Software_f160520131.pdf
- Mertens P, Große-Wilde J, Wilkens I (2005) Die (Aus-)Wanderung der Softwareproduktion – Eine Zwischenbilanz, vol 38. Institut für Informatik der Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen-Nürnberg
- Meyer-Stamer J (1999) Lokale und regionale Standortpolitik – Konzepte und Instrumente jenseits von Industriepolitik und traditioneller Wirtschaftsförderung. Institut für Entwicklung und Frieden, Gerhard-Mercator-Universität Duisburg
- Morstead S, Blount G (2003) Offshore ready: strategies to plan and profit from Offshore IT-enabled services. ISANI, Houston
- NASSCOM (2007) NASSCOM's education initiatives. Sustaining India's talent edge to fuel the next wave of IT-BPO industry growth. NASSCOM Press Information Note vom 5. Juli 2007. <http://www.nasscom.in/upload/5216/July%205%202007%20%20Education%20/initiatives-Final.pdf>
- oV (2006a) Information technology: Annual Report 2005-06. In: Government of India, Ministry of Communications & Information Technology (Hrsg) New Delhi. <http://www.mit.gov.in/annual-report2005-06.pdf>
- Payscale (2010) Salary benchmarking. www.payscale.com
- Pfeffer J, Salancik G (1978) The external control of organizations - a resource dependence perspective. Harper & Row, New York
- Prehl S (2006) Der Offshore-Trend erreicht Europa. *Computerwoche Online* 18:38–38
- Robinson M, Kalakota R (2005) Offshore outsourcing. Mivar Press, Alpharetta
- Thondavadi N, Albert G (2004) Offshore outsourcing. 1stbooks, Bloomington
- Vashistha A, Vashistha A (2006) The offshore nation. McGraw-Hill, New York
- Whitten D, Wakefield RL (2006) Measuring switching costs in IT outsourcing services. *J Strateg Inf Syst* 15(3):219–248
- Whyte G (1994) The role of asset specificity in the vertical integration decision. *J Econ Behav Org* 23(3):287–302
- Willcocks L, Lacity MC (2006) Global sourcing of business and IT services. Palgrave Macmillan, New York

6.1 Überblick

Das Aufkommen von Plattformkonzepten lässt sich in zahlreichen Industrien beobachten. Während in der Automobilbranche bereits in den neunziger Jahren Produktplattformen als neues Erfolgsrezept gefeiert und in der Folge auf zahlreiche andere Branchen übertragen wurden (vgl. z. B. Köhler 2004 zu Produktplattformen in der Medienindustrie), werden wir im Folgenden softwarebasierte Branchenplattformen näher betrachten. Aufbauend auf der Arbeit von Gawer (2009) lassen sich grundsätzlich zwei generische Plattformtypen voneinander abgrenzen: Produktplattformen ermöglichen die effiziente Herstellung von Produkten und Services durch die Wiederverwendung bereits erstellter Module (Wheelwright und Clark 1992); Branchenplattformen dienen demgegenüber in erster Linie der Attraktion komplementärer Produkte bzw. Services von Dritten aus einer Branche (Cusumano und Gawer 2002). Abbildung 6.1 zeigt weitere Merkmale der beiden Typen.

Beide Ausprägungen von Plattformen finden sich mittlerweile auch in der Softwareindustrie. Im Bereich der Software für Konsumenten gibt es bereits eine Vielzahl von Beispielen; am bekanntesten ist der von Apple betriebene AppStore für das iPhone. Branchenplattformen für Unternehmenssoftware sind beispielsweise salesforce.com AppExchange, Google Apps Marketplace oder SugarCRM SugarExchange.

6.2 Produktplattformen in der Softwareindustrie

6.2.1 Kostenstruktur plattformbasierter Softwareentwicklung

Analog zu den bekannten Beispielen aus der Automobilindustrie besteht ein wesentliches Ziel der Entwicklung und Nutzung von Produktplattformen in der Softwareindustrie in

Plattform Typ	Produktplattform	Branchenplattform
Beteiligte	Ein Unternehmen, ggf. zusammen mit Zulieferern	Mehrere Unternehmen mit kompatiblen Produkten oder Services
Zielsetzungen	<ul style="list-style-type: none"> ■ Steigerung von Effizienz und Geschwindigkeit in Entwicklung und Produktion ■ Größere Produktvielfalt bei geringeren Kosten ■ Mehr Flexibilität bei der Gestaltung neuer Produkte 	<ul style="list-style-type: none"> ■ Für Plattformbetreiber: Steigerung des Nutzens der Plattform durch komplementäre Produkte und Services ■ Für Komplementäre: Absatz(-steigerung)
Gestaltungsprinzipien	<ul style="list-style-type: none"> ■ Wiederverwendung von Komponenten ■ Beständigkeit der grundlegenden Architektur 	<ul style="list-style-type: none"> ■ Stabile Schnittstelle für Erweiterungen

Abb. 6.1 Plattform Typologie. (in Anlehnung an Gawer (2009))

der Reduktion von Entwicklungskosten und -zeit bei mindestens gleichbleibender Qualität. Software-Produktplattformen dienen insbesondere dazu, Produkte einer Produktlinie für unterschiedliche Zielgruppen, Preiskategorien und Systemumgebungen flexibel und vor allem effizient herstellen und anbieten zu können (Reussner und Hasselbring 2006).

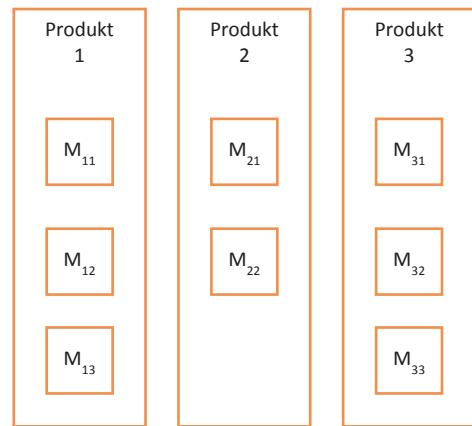
Derartige Produktplattformen sind auf die Wiederverwendung von Softwaremodulen ausgerichtet – genauso wie im zitierten Beispiel der Automobilindustrie die Wiederverwendung von Teilen im Fokus steht (Boysen und Scholl 2009). Ein Softwaremodul ist ein funktional abgrenzbares Stück Software, das über Schnittstellen – etwa auf Basis der im letzten Kapitel vorgestellten SOA-Standards – mit anderen Modulen kommuniziert, aber unabhängig von diesen ausgeführt werden kann. Module werden über eine Konfigurationssoftware, die Produktplattform, verwaltet, wobei – was entscheidend ist – ein Modul in möglichst vielen Produkten zum Einsatz kommen sollte (Baldwin und Clark 1997; Miller und Elgård 1998; Meyer und Lehnerd 1997).

Die Veränderungen der Kostenstruktur eines Softwareanbieters durch Einführung einer Produktplattform lässt sich mit Hilfe eines einfachen Kostenkalküls aufzeigen. Dabei gehen wir davon aus, dass jedes Produkt dieses Anbieters individuell, aber bereits modularisiert entwickelt wird. Folglich werden alle N Produkte, die jeweils aus M_n ($n=1 \dots N$) Modulen bestehen, unabhängig voneinander entwickelt. Abbildung 6.2 zeigt ein Beispiel mit drei Produkten, die aus jeweils zwei bzw. drei produktindividuellen entwickelten Modulen bestehen.

Dabei entstehen für ein System n die Entwicklungskosten K_n , die sich aus den Entwicklungskosten der M_n einzelnen Module K_{nm} ($m=1 \dots M_n$) zusammensetzen. Als Kosten für die Entwicklung der N Systeme eines Herstellers ergeben sich damit insgesamt:

$$K = \sum_{n=1}^N K_n = \sum_{n=1}^N \sum_{m=1}^{M_n} K_{nm}$$

Abb. 6.2 Beispiel für die Entwicklung von drei Produkten ohne Plattform



Diesem Ansatz soll nun die Kostenstruktur gegenüber gestellt werden, die sich nach Einführung einer Produktplattform bzw. einer Wiederverwendung von Komponenten ergibt. Für einen Hersteller bedeutet der Wechsel auf das Plattformkonzept zunächst, dass Kosten für den Aufbau der Produktplattform anfallen. Diese bezeichnen wir als K_p . Darüber hinaus werden Module entwickelt. Für jedes der D Module entstehen Kosten, die wir mit K_d ($d=1 \dots D$) bezeichnen. Jedes Modul wird in $1 \leq n \leq N$ Systemen eingesetzt. Für die Zusammenführung der Module zu einem System fallen ebenfalls Kosten an. Vereinfachend gehen wir dafür von einem Durchschnittskostensatz aus, den wir als Integrationskostensatz K_I bezeichnen. Für die Entwicklung der N Systeme eines Herstellers ergeben sich bei Nutzung des Plattformkonzepts folgende Kosten:

$$K = K_p + \sum_{d=1}^D K_d + N * K_I$$

Abbildung 6.3 zeigt eine Verwendungsmatrix, aus der hervorgeht, welche Module in welchen Produkten verwendet werden. In unserem Beispiel wird Modul 3 in den Produkten 1 und 3 verwendet; Modul 1 geht in alle drei Produkten ein. Die Module 2, 4 und 5 werden nur in einem Produkt verwendet.

Aus dem Vergleich der beiden Kostenfunktionen (1) und (2) lassen sich Schlussfolgerungen zu den zu erwartenden Veränderungen der Kostenstruktur nach Einführung einer Produktplattform ableiten. Zunächst wird unmittelbar klar, dass die Einführung einer Produktplattform eine nicht unerhebliche und mit Risiko behaftete Investition in die Entwicklung einer Produktplattform voraussetzt. Da zudem für jedes der N Produkte Integrationskosten K_I anfallen, deren Höhe in der Praxis vor allem von der Qualität und der vorausschauenden Gestaltung der Plattform-Schnittstellen abhängt, stellt diese Anfangsinvestition zunächst eine Hürde dar.

Trotz dieser zu erwartenden Mehrkosten führt der Einsatz von Produktplattformen jedoch besonders dann zu erheblichen Kosteneinsparungen, wenn ein hoher Wiederver-

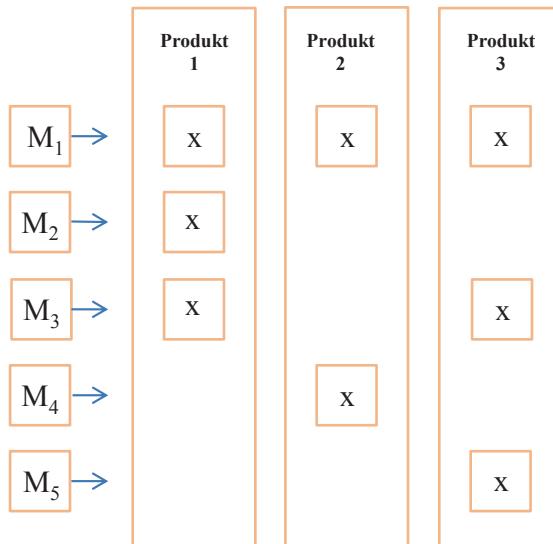


Abb. 6.3 Konfiguration von drei Produkten über eine Produktplattform

wendungsgrad erreicht werden kann, d. h., wenn sich einmal erstelle Module in möglichst vielen Produkten verwenden lassen bzw. sich die Zahl der Module für die Erstellung einer gegebenen Zahl von Produkten reduzieren lässt. In unserem Beispielfall geht es um die zweite Variante: die Zahl der Produkte ist unverändert, die Zahl der entwickelten Module konnte jedoch von acht auf fünf reduziert werden.

In der bereits mehrfach zitierten Automobilindustrie stellen die großen Hersteller nur einen kleinen Teil der Module selbst her und konzentrieren sich in erster Linie auf die Integration der Module zu einem Produkt. Die Entwicklung und Produktion der Module liegt dagegen bei hoch spezialisierten Zulieferern.

6.2.2 Add-on: Industrialisierung als Managementkonzept für die Softwareindustrie

Industrialisierung ist ein historisch gewachsenes Managementkonzept, das Ansatzpunkte zur kostengünstigen Massenproduktion liefert. Als wesentliche Ansatzpunkte sieht dieses Konzept eine verstärkte Standardisierung von Produkten und Prozessen, zunehmende Spezialisierung (d. h. mehr Arbeitsteilung) sowie Automatisierung vor (Heinen 1991, S. 10; Schweitzer 1994, S. 19). Die drei genannten Ansatzpunkte hängen zusammen, da Standardisierung die wichtigste Voraussetzung ist, um sowohl eine Spezialisierung als auch eine Automatisierung realisieren zu können. Oder anders ausgedrückt: Spezialisierung und Automatisierung setzen Standardisierung voraus. Diese alleine kann aber auch schon zu einer Stückkostenreduktion führen. Hintergrund ist, dass sich nur standardisierte Prozesse auf eine Maschine verlagern bzw. auf mehrere Aufgabenträger verteilen lassen. Genauso setzen standardisierte Prozesse standardisierte Produkte voraus.

Eine zentrale Rolle im Konzept der Industrialisierung spielen neue Technologien. Diese eröffnen neue Spielräume für Automatisierung, Spezialisierung und gegebenenfalls auch für die Standardisierung und werden daher im Kontext der Industrialisierung häufig als Treiber bezeichnet.

Ausgehend von den treibenden Technologien lassen sich zwei abgeschlossene und eine noch andauernde Phase der Industrialisierung abgrenzen (Condrau 2005). Der Ursprung findet sich in der um das Jahr 1780 beginnenden „industriellen Revolution“. Durch die Entwicklung von Dampfmaschine, Eisenbahn und Webstuhl konnte die bis dato vorherrschende handwerkliche Einzelproduktion durch eine erste Form der industriellen Massenproduktion abgelöst werden. Erfolgte die Produktion bis zu diesem Zeitpunkt vornehmlich individuell und für den Eigenbedarf, so ermöglichte die maschinelle Unterstützung nun eine eher standardisierte, zum Verkauf bestimmte Erstellung von Gütern in hoher Stückzahl. Rückblickend steht diese erste Phase der Industrialisierung somit für eine Zeit enormer Produktivitätssteigerungen und eines rasanten Wirtschaftswachstums. Eine Evolution durchlief das Konzept in der um 1840 beginnenden zweiten Phase der Industrialisierung. Neben der wissenschaftlichen Ausdifferenzierung – hier sei vor allem auf die Arbeiten von Taylor zum Scientific Management verwiesen – eröffneten die Entdeckung der Elektrizität und die Entwicklung von Elektromotoren neue Ansatzpunkte zur Industrialisierung. Unterlag der Produktionsprozess in der ersten Phase der Industrialisierung noch weitestgehend dem Prinzip der Mengenteilung (Aufteilung gleichartiger Tätigkeiten auf verschiedene Aufgabenträger), so ermöglichte die Installation von Fließbändern und eine damit einhergehende weitere Standardisierung nun das Prinzip der Aufgabenteilung nach der Art der Aufgabe. Infolge dieser zunehmenden Spezialisierung konnten hoch standardisierte materielle Güter, wie beispielsweise das Ford T-Modell, erstellt und weitere Produktivitätssteigerungen erzielt werden.

Die mit Ausgang des 20. Jahrhunderts beginnende „informationstechnische Revolution“ läutet nach heutiger Einschätzung eine dritte Phase der Industrialisierung ein. Fokussierten die ersten beiden Phasen noch die Produktion materieller Güter, so steht nun die Produktion informationsintensiver Dienstleistungen und Produkte im Mittelpunkt. Treiber dieser Entwicklung sind die bekannten Innovationsschübe bei den Informations- und Kommunikationstechnologien. Zusammenfassend zeigt Abb. 6.4 die an verschiedenen Stellen in diesem Buch bereits diskutierten Ansatzpunkte für eine Industrialisierung der Softwareentwicklung.

Abbildung 6.5 stellt die drei Phasen der Industrialisierung mit ihren wichtigsten Charakteristika überblicksartig dar. In der Literatur wird gelegentlich der Dienstleistungssektor als primäres Ziel der dritten Phase der Industrialisierung dargestellt. Wir folgen dieser Sichtweise nicht, weil sich die Industrialisierung durch moderne Informations- und Kommunikationstechnologien auch auf Informationsprodukte wie Bücher oder Software auswirkt, die sich nicht dem Dienstleistungssektor zuordnen lassen.

Dabei gilt, dass Arbeitsteilung und Automatisierung, und damit dem Konzept der Industrialisierung, klare Grenzen gesetzt sind. Diese Grenzen liegen in der Motivation der Mitarbeiter und den Kosten einer verteilten Bearbeitung bzw. in der Flexibilität der Produktionsprozesse. Ein hohes Maß an Arbeitsteilung führt zu gleichartigeren Tätigkeiten,

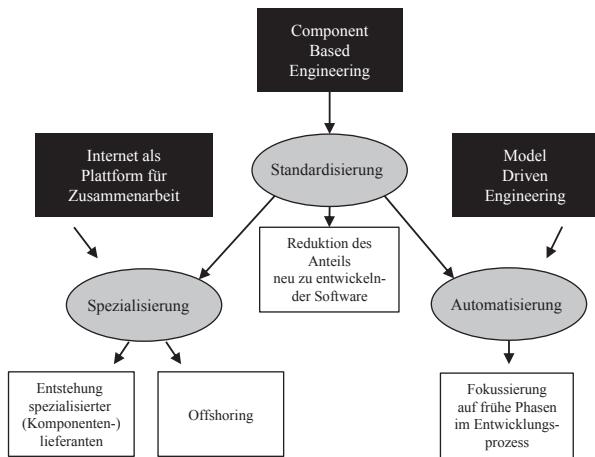


Abb. 6.4 Ansatzpunkte und erwartete Wirkungen einer Industrialisierung der Softwarebranche

	1. Phase der Industrialisierung	2. Phase der Industrialisierung	3. Phase der Industrialisierung
Treiber	Dampfmaschine, Eisenbahn, Webstuhl	Elektrizität, Elektromotoren	Informations- und Kommunikationstechnologien
Sektor	Materielle Güter	Materielle Güter	Informations-intensive Produkte und Dienstleistungen
Zeitspanne	ca. 1780-1840	ca. 1840-1960	seit Ende des 20. Jahrhunderts
Schlagwörter	Industrielle Revolution	Fordismus und Taylorismus	Informations-technische Revolution

Abb. 6.5 Drei Phasen der Industrialisierung

was bei extremer Ausprägung von den Ausführenden als stumpfsinnig und monoton empfunden wird, woraufhin die Motivation sinkt und der Stückkostenvorteil relativiert wird. Zudem setzt ein hohes Maß an Automatisierung auch einen hohen Standardisierungsgrad voraus, wodurch Veränderungen an den Prozessen, und damit auch an den Produkten, tendenziell schwieriger werden. Beide Aspekte führen dazu, dass die verarbeitende Industrie heute keinesfalls mehr auf ein Maximum an Industrialisierung setzt.

6.3 Branchenplattformen in der Softwareindustrie

6.3.1 Offenheit einer Branchenplattform

Wie bereits am Anfang dieses Kapitels angesprochen, stellen Branchenplattformen die Basis von Produkten bzw. Services dar, deren Funktionalitäten durch Dritte (so genannte Komplementäre) erweitert werden. So sind beispielsweise in Spielkonsolen nur die für Spiele notwendigen Basisfunktionalitäten, wie beispielsweise Grafikbeschleuniger, implementiert. Die Spiele selbst werden hingegen von Dritten entwickelt. Das primäre Ziel der Betreiber von Branchenplattformen liegt daher darin, die Attraktivität der eigenen Plattform durch ein möglichst breites Angebot komplementärer Produkte für eine möglichst große Gruppe von Endkunden sicherzustellen, wobei Endkunden sowohl Konsumenten als auch Unternehmen sein können.

Etwas allgemeiner ausgedrückt geht es einem Plattformbetreiber damit primär um den richtigen Umfang der Offenheit seiner Branchenplattform. Eine Plattform wird als völlig offen bezeichnet, wenn es keine Einschränkungen in Bezug auf Teilnahme, Entwicklung, Nutzung und Kommerzialisierung der Plattform gibt (Eisenmann et al. 2009). Gerade bei Plattformen kommerzieller Anbieter ist dieser Extremfall aber nur selten anzutreffen. Vielmehr geht es um die Frage nach dem aus Sicht des Plattformanbieters optimalen Umfang an Öffnung einer Plattform.

Die Schwierigkeit in der Wahl des optimalen Umfangs der Plattform-Öffnung liegt im Trade-off zwischen Offenheit und Kontrolle: Während mehr Offenheit das Mitwirken von Dritten tendenziell fördert und damit die Attraktivität der Plattform steigert, geht sie in der Regel auch mit einem Verlust von Kontrolle über die Gestaltung der Plattform einher (West 2003). Im Folgenden wird zwischen einer vertikalen und einer horizontalen Öffnung unterschieden.

6.3.1.1 Vertikale Öffnung

Unter vertikaler Öffnung wird verstanden, inwieweit komplementäre Produkte oder Services von externen Partnern bereitgestellt werden können und sollen. Dabei lassen sich drei plattformspezifische Stellschrauben unterscheiden, die wir nachfolgend vorstellen.

Exklusivität Ein wichtiger Parameter bei der Wahl des Offenheitsgrades liegt in der Vereinbarung von Exklusivrechten. Auf diese Weise soll sichergestellt werden, dass ein bestimmtes komplementäres Produkt exklusiv auf einer Plattform angeboten wird (Eisenmann und Wong 2004). Beispielsweise sind neue Versionen des beliebten Konsolenspiels Grand Theft Auto GTA immer zunächst nur auf Sonys Playstation-Plattform verfügbar. Eine zweite Form einer Exklusivitätsvereinbarung besteht in der so genannten kategorialen Exklusivität. Hier vereinbaren Plattformbetreiber und Komplementär, dass eine bestimmte Sorte von Applikationen exklusiv von diesem Komplementär angeboten werden darf. Solche Strategien, die einer Reduktion der Offenheit gleichkommen, sind dabei besonders dann sinnvoll, wenn eine Seite hohe und spezifische Investitionen vornehmen

muss und dazu nur bereit ist, wenn die jeweils andere Seite im Gegenzug den exklusiven Zugriff auf den Distributionskanal zusichert. Auch diese Form der Vereinbarung lässt sich im Markt für Konsolenspiele beobachten: Üblicherweise beschränken Plattformbetreiber den Zugang zum Markt für Konsolenspiele einerseits, um eine hohe Qualität dieser ausgewählten Spiele sicherzustellen, und andererseits, um entsprechend hohe Lizenzgebühren einnehmen zu können.

Rückwärtskompatibilität Im Rahmen der Weiterentwicklung einer Plattform werden üblicherweise auch neue bzw. andere Funktionen für Entwickler komplementärer Applikationen verfügbar. Eine wichtige Entscheidung liegt dabei darin, inwieweit eine Rückwärtskompatibilität sichergestellt wird, d. h. ob komplementäre Erweiterungen, die für eine ältere Plattformversion entwickelt wurden, auch auf neueren Versionen lauffähig sind. Aus Sicht der Plattformanbieter ist hierbei abzuwägen, ob sie die potenziell höheren Kosten und Einschränkungen in der Weiterentwicklung der Plattform akzeptieren wollen, um das vorhandene komplementäre Angebot auch auf neuen Plattformversionen verfügbar zu machen (Choi 1994). Gerade wenn Applikationen von Partnern ein zentrales Feature des Plattformprodukts darstellen, ist die Sicherstellung der Rückwärtskompatibilität eine essentielle Aufgabe. Mit dem Slogan „Es gibt für alles eine App“ hat Apple das Angebot komplementärer Applikationen als zentralen Wettbewerbsvorteil des iPhone etabliert. In der Folge musste bei der Entwicklung der vierten Generation auch die Rückwärtskompatibilität sichergestellt werden. Daher wurde die neue Displayauflösung auch so gewählt, dass sowohl in der Höhe als auch in der Breite exakt die doppelte Anzahl an Pixeln verfügbar ist. Durch diesen Schritt konnte erreicht werden, dass auch alle für die vorangegangenen iPhone-Generationen entwickelten Applikationen weiterhin uneingeschränkt lauffähig sind. Hätte Apple dies nicht berücksichtigt, wären alle komplementären Entwickler gezwungen gewesen, eine auf die neue Plattform angepasste Version bereitzustellen, was einem geringeren Grad der vertikalen Offenheit gleichgekommen wäre. Auch bei der fünften Generation setzt man auf volle Rückwärtskompatibilität. So sind alle Apps, die für das iPhone 4 entwickelt wurden, auf dem iPhone 5 lauffähig, indem auf dem größeren Display des iPhone 5 oben und unten schwarze Balken hinzugefügt wurden. Darüber hinaus akzeptiert Apple nur noch neue Apps, die auf das iPhone 5 optimiert wurden (Chip 2012).

Eingliederung von Komplementen in den Kern der Plattform Ein weiteres Mittel zur Gestaltung des Grades der Offenheit besteht darin, im Rahmen der Weiterentwicklung einer Plattform komplementäre Applikationen, die vormals von externen Partnern bereitgestellt wurden, in den Kern der Plattform zu integrieren. Ein Beispiel für eine solche Strategie ist das Betriebssystem Windows von Microsoft: Zahlreiche, in aktuellen Versionen fest integrierte Standardapplikationen, wie Browser, Mediaplayer oder System Utilities, wurden in früheren Versionen nur als optionale Komponenten von Dritten angeboten. Wie im nachfolgenden Abschnitt zum Management von Komplementären erläutert wird, kann ein solcher Schritt als vertikale Schließung der Plattform verstanden werden, da

Komplementäre das Risiko fürchten müssen, mit dem Betreiber zu konkurrieren (Yoffie und Kwak 2006). Gleichzeitig sprechen jedoch auch Gründe, wie die Reduktion der strategischen Abhängigkeit von einzelnen Anbietern oder die Möglichkeit Skaleneffekte zu realisieren, für die Wahl einer solchen Strategie.

6.3.1.2 Horizontale Öffnung

Die Öffnung einer Plattform gegenüber anderen Plattformen oder Dritten wird als horizontal bezeichnet. Dies kann mit Hilfe der folgenden Stellschrauben erreicht werden (Eisenmann et al. 2009):

Interoperabilität mit anderen Plattformen Die horizontale Öffnung einer Plattform wird insbesondere durch die Bereitstellung von offenen Schnittstellen sowie Werkzeugen erreicht (Katz und Shapiro 1985). Die Facebook Connect Schnittstelle ist ein Beispiel für eine solche Herstellung von Interoperabilität: Andere Plattformen, wie beispielsweise Yahoo, können nach Anmeldung des Nutzers auf dessen Profildaten zugreifen, gleichzeitig werden auf Facebook selbst auch die Aktivitäten des Nutzers auf anderen Plattformen angezeigt. Da die Facebook-Plattform zum Zeitpunkt der Einführung dieses „Converters“ (Ende 2008) bereits eine gewisse Reife erreicht hatte, stellte diese strategische Öffnung eine Möglichkeit dar, das Wachstum der Nutzerzahlen über die Kernzielgruppe hinaus zu fördern. Umgekehrt ist die Nutzung von Facebook Connect auch für konkurrierende Plattformen interessant, da so der Zugriff auf die Daten der bei Facebook bereits registrierten Nutzer möglich wird.

Lizenzierung weiterer Plattform-Betreiber In der Phase des Aufbaus einer Plattform werden häufig einseitige Subventionen eingesetzt, um den aus der Netzeffekttheorie bekannten „Pingueffekt“ zu vermeiden (siehe Abschn. 2.2.2.1). In dieser Phase ist es oft sinnvoll, nur einen proprietären Plattformbetreiber zu lizenzieren, da so verhindert werden kann, dass einseitige Subventionen von Trittbrettfahrern ausgenutzt werden (Eisenmann 2008). Hat die Plattform jedoch eine gewisse Reife erreicht, kann eine strategische Öffnung durch Lizenzierung weiterer Betreiber das Wachstum erheblich beschleunigen. Diese Option ist dabei besonders attraktiv, wenn die zusätzlichen Betreiber durch deren spezifisches Wissen innovative Formen der Plattform bereitstellen und damit den Kreis der potenziellen Nutzer der Plattform insgesamt vergrößern (Cusumano und Gawer 2002).

Ein Beispiel für den Erfolg einer solchen Öffnung lässt sich im Markt für Smartphone-Betriebssysteme beobachten: Das Betriebssystem Android, das von der von Google angeführten Open Handset Alliance entwickelt und bereitgestellt wird, wurde durch zahlreiche Smartphone Hersteller, wie HTC, Motorola, Samsung, Dell oder Sony Ericsson, als Betriebssystem für deren Endgeräte lizenziert. Diese Strategie, weitere Betreiber zu lizenzieren (ohne die Kontrolle über die Entwicklung der Plattform abzugeben), hat sich als erfolgreich herausgestellt, da Android in 2013 einen Marktanteil von fast 75% verzeichnen konnte (Gartner 2013b).

Aufnahme von Plattform-Sponsoren Darüber hinaus kann eine horizontale Öffnung auch durch Hinzunahme von Sponsoren erfolgen. Im Gegensatz zu Lizenznehmern, deren spezifische Erweiterungen auf dem Plattformkern aufbauen, werden Plattformsponsoren auch an der technischen Weiterentwicklung der Plattform beteiligt. Auf die Vorteile solcher Entwicklungspartnerschaften sind wir bereits in Abschn. 3.1.1.2 eingegangen. Im Gegenzug resultiert aus einer Öffnung gegenüber weiteren Sponsoren jedoch stets auch eine erhöhte Komplexität in der Koordination zwischen den Sponsoren und mehr Aufwand zur Festlegung von gemeinsamen Standards (West 2006).

Verglichen mit der Lizenzierung weiterer Betreiber ist die Erweiterung des Sponsorenkreises daher mit einem erheblichen Risiko verbunden, da im schlimmsten Fall politische Zerwürfnisse zwischen den Sponsoren die Weiterentwicklung stark verlangsamen oder gar vollständig verhindern können. Nach West (2003) sollte eine solche Öffnung insbesondere auch dann forciert werden, wenn das Geschäftsmodell des ursprünglichen Plattformbetreibers in erster Linie nicht auf die Lizenzierung der Plattform, sondern auf den Vertrieb komplementärer Produkte oder Dienstleistungen abzielt. Nicht zuletzt aus diesem Grund hat beispielsweise IBM seine Rechte an der Entwicklungsplattform Eclipse der Open-Source-Entwicklergemeinde übertragen, um damit den Absatz komplementärer Integrationsdienstleistungen zu fördern. Auf diesen Aspekt werden wir im siebten Kapitel dieses Buchs näher eingehen.

Eine weitere Ursache für eine Öffnung gegenüber neuen Sponsoren kann auch darin liegen, dass die Plattform einem erheblichen Druck von konkurrierenden Plattformen ausgesetzt ist. Als Resultat des verlorenen Browser-Kriegs hat beispielsweise Netscape mit dem Mozilla Projekt 1998 den Quellcode des damals aktuellen Netscape Communicators vollständig offengelegt und sich damit gegenüber weiteren Sponsoren geöffnet. Aus der Mozilla Foundation ist in der Folge der Firefox-Browser hervorgegangen.

6.3.2 Das Management der Komplementäre

Die intensive Analyse der eigenen Stärken sowie der Konkurrenten steht zumeist im Fokus des Managements von Softwareanbietern. Die Analyse der vermeintlich „befreundeten“ Zulieferer komplementärer Applikationen wird hingegen häufig vernachlässigt. Das Bewusstsein darüber, dass insbesondere in Branchenplattformen eine Abhängigkeit des Plattform-betreibers von der Bereitstellung komplementärer Applikationen besteht, ist zwar durchaus ausgeprägt. Gleichzeitig wird die Gemeinsamkeit der Interessen jedoch häufig überschätzt: Obgleich beide Seiten aufgrund der indirekten Netzwerkeffekte ein gemeinsames Interesse am Wachstum der Plattform haben, endet das gemeinsame Interesse beispielsweise bereits bei der Frage, in welchem Verhältnis die zusätzlichen Gewinne zwischen Betreiber und Komplementär aufgeteilt werden sollen.

Aus diesem Grund stellt das Management von Komplementären eine wichtige Aufgabe von Plattformanbietern dar. Neben einer intensiven Analyse der Geschäftsmodelle, Strategien, Zielen, Fähigkeiten und Motiven von Komplementären umfasst dies auch die Wahl ei-

nes geeigneten Paradigmas zur Gestaltung des Verhältnisses zwischen Plattformbetreiber und Komplementären. Mit den beiden Formen der Hard- und Soft-Power hat Ney (2004) zwei gegensätzliche Paradigmen herausgearbeitet, die im Folgenden kurz dargestellt werden.

Die zunächst naheliegenden Mittel zur Beeinflussung von Komplementären fallen in die Kategorie der „Hard-Power“: Durch den Aufbau einer glaubwürdigen Drohkulisse oder finanzieller Anreize, wie beispielsweise der Beteiligung am Umsatz, soll sichergestellt werden, dass sich Komplementäre im Sinne des Plattformbetreibers verhalten. Ein historisch belegtes Beispiel der Anwendung von Hard-Power lag in der Androhung von Bill Gates, die Entwicklung von Microsoft Office für Apples Mac OS einzustellen, falls sich Apple weigern sollte, Microsofts Internet Explorer weiterhin im Mac OS zu integrieren. Solche Mittel beruhen im Allgemeinen auf traditionellen Quellen der Stärke, wie beispielsweise einem hohen Marktanteil oder der exklusiven Kontrolle über einen Distributionskanal. Im Kontext von Plattformen ist die Anwendbarkeit von Hard-Power zudem durch die weitgehende Unabhängigkeit von einzelnen Komplementären bedingt. Ein Instrument der Hard-Power besteht daher auch darin, diese Abhängigkeit von Komplementären zu reduzieren, indem gezielt strategisch wichtige komplementäre Angebote durch den Plattformbetreiber selbst produziert und angeboten werden. So kann etwa in der Smartphone-Branche beobachtet werden, dass trotz des Trends, ein möglichst breites Spektrum von Apps durch Marktplätze zu koordinieren, wichtige Kernfunktionalitäten eines Smartphones, wie beispielsweise Telefonie oder Textnachrichten, nach wie vor fest in die Plattform integriert sind. Neben der Möglichkeit, Skaleneffekte zur realisieren und durch den Vertrieb dieser Komponente zusätzlichen Absatz zu generieren, kann eine solche strategische Maßnahme auch eingesetzt werden, um eine Signalwirkung zu erzeugen. Insbesondere wenn ein Plattformbetreiber jedoch auf ein möglichst breites komplementäres Angebot angewiesen ist, kann dieser Schritt kontraproduktive Wirkung haben: Das Signal, dass der Plattformbetreiber in den Markt eingreifen könnte und damit die Absatzmöglichkeiten eines Komplementärs gefährdet, kann dazu führen, dass Komplementäre von einer Zusammenarbeit Abstand nehmen.

Die Nachteile der Anwendung von Hard-Power liegen damit auf der Hand: Eingriffe in den Markt für komplementäre Angebote sind vor allem auf Dauer mit erheblichen Kosten verbunden und verhindern den Aufbau eines nachhaltigen und vertrauensvollen Verhältnisses zu den Komplementären. Zudem ist davon auszugehen, dass Komplementäre versuchen werden, sich von besonders starken Plattformbetreibern nicht zu abhängig zu machen und daher langfristig eher Konkurrenten unterstützen könnten.

Eine Alternative zur Hard-Power stellt daher die „Soft-Power“ dar: Eine im Allgemeinen billigere und langfristig erfolgreichere Methode, Komplementäre zur Zusammenarbeit zu bewegen, besteht darin, sie von gemeinsamen Zielen und Chancen zu überzeugen. Konkrete Maßnahmen hierfür liegen beispielsweise in der proaktiven Kommunikation von Marktdaten und Planungen für die Zukunft der Plattform. Auch das Verkünden einer gemeinsamen Vision, die auch die Vorteile der Komplementäre klar darstellt, kann ein Instrument der Soft-Power sein. Ein, wenn auch nicht originär aus der Softwareindustrie

stammendes, Beispiel waren die Bemühungen von Steve Jobs zur Einbindung des Angebots aller großen Musik-Labels in seine iTunes Plattform: Indem er 2003 erfolgreich eine gemeinsame Zukunftsvision für die Branche aufzeigte, konnte er alle Labels überzeugen mit ihm zu Konditionen zusammenzuarbeiten, die ein auch preislich attraktives Angebot des Apple iTunes Stores sicherstellten.

Die Nachteile dieses Vorgehens sind, dass „Soft-Power“ nur als langfristig angelegte Maßnahme erfolgreich sein kann und ein Betreiber, der ausschließlich „Soft-Power“ anwendet, leicht durch einen Konkurrenten „überrumpelt“ werden könnte.

Wie aufgezeigt wurde, können sowohl Hard- als auch Soft-Power erfolgreiche Wege im Umgang mit Komplementären sein. Zur Auswahl des geeigneten Paradigmas haben Yoffie und Kwak (2006) drei Faktoren identifiziert:

- Leistungsfähigkeit: Insbesondere die Anwendung von Hard-Power erfordert den Einsatz erheblicher (im Allgemeinen finanzieller) Ressourcen sowie eine entsprechende Positionierung im Markt. Sind solche Voraussetzungen nicht gegeben, sollte ein Plattformbetreiber eher die Anwendung von „Soft-Power“ in Erwägung ziehen. Gerade kleinere und vermeintlich schwächere Firmen können so für Komplementäre attraktive Partner darstellen, da diese dann keinen Eingriff in ihre Geschäftstätigkeit befürchten müssen.
- Vielfalt des komplementären Angebots: Ist ein Plattformbetreiber auf ein möglichst breites und vielfältiges Angebot komplementärer Produkte angewiesen, spricht das eher für die Anwendung von Instrumenten der Soft-Power, da nur so die Plattform langfristig attraktiv für Komplementäre sein kann.
- Spezifische Investitionen der Komplementäre: Wenn die Angebote der Komplementäre optimal in die Plattform eingebunden werden sollen, erfordert das zumeist spezifische und irreversible Investitionen des Partners. Sind solche Investitionen notwendig, werden potenzielle Partner darauf bedacht sein, sich soweit wie möglich gegen ein Scheitern der Beziehung zum Plattformbetreiber abzusichern. Folglich kann davon ausgegangen werden, dass ein entsprechendes Vertrauen eher durch die Instrumente der Soft-Power aufgebaut werden kann.

Für Plattformbetreiber stellt sich damit die Frage hinsichtlich Hard- oder Soft-Power nicht als „Entweder-oder-Frage“, sondern es wird nach einer geeigneten Mischform gesucht. Ein von vielen Betreibern gewählter Mittelweg liegt darin, im Sinne der Hard-Power nur strategisch besonders wichtige Komplemente selbst zu produzieren. Darüber hinaus wird nicht oder nur rudimentär, etwa im Rahmen der Durchführung von Qualitätskontrollen, in den Markt komplementärer Angebote eingegriffen, um mit den Instrumenten der Soft-Power eine möglichst stabile und offene Beziehung gegenüber den Partnern zu etablieren.

Literatur

- Baldwin CY, Clark KB (1997) Managing in an age of modularity. *Harv Bus Rev* 75:84–93
- Boysen N, Scholl A (2009) A general solution framework for component commonality problems. *BuR Bus Res* 2/1:86–106
- Chip (2012) iPhone 5 display: Diese Apps sind schon optimiert. http://www.chip.de/news/iPhone-5-Display-Diese-Apps-sind-schon-optimiert_57616163.html
- Choi J (1994) Network externality, compatibility choice, and planned obsolescence. *J Ind Econ* 42:167–182
- Condrau F (2005) Die Industrialisierung in Deutschland. Wissenschaftliche Buchgesellschaft, Darmstadt
- Cusumano M, Gawer A (2002) The elements of platform leadership. *MIT Sloan Manage Rev* 43:51–58
- Eisenmann T (2008) Managing proprietary and shared platforms. *Calif Manage Rev* 50:31–53
- Eisenmann T, Wong J (2004) Electronic arts in online gaming. *Harv Bus Sch Case* 804–140
- Eisenmann T, Parker G, Van Alstyne M (2009) Opening platforms: how, when and why? In: Gawer A (Hrsg) Platforms, markets and innovation. Edward Elgar, London
- Gartner (2013b) Gartner says Asia/Pacific led worldwide mobile phone sales to growth in first quarter of 2013. <http://www.gartner.com/newsroom/id/2482816>
- Gawer A (2009) Platform dynamics and strategies: from products to services. In: Gawer A (Hrsg) Platforms, markets and innovation, S 45–76
- Heinen E (1991) Industriebetriebslehre – Entscheidungen im Industriebetrieb. Gabler, Wiesbaden
- Katz M, Shapiro C (1985) Network externalities, competition, and compatibility. *Am Econ Rev* 75:424–440
- Köhler L (2004) Produktinnovation in der Medienindustrie - Organisationskonzepte auf Basis von Produktplattformen. Gabler, Wiesbaden
- Meyer MH, Lehnerd AP (1997) The power of product platforms: building value and cost leadership. New York
- Miller D, Elgård P (1998) Defining modules, modularity and modularization. In: Proceedings of the 13th IPS research seminar, Fuglsoe
- Ney J (2004) Power in the global information age. Routledge, London
- Reussner R, Hasselbring W (2006) Handbuch der Software-Architektur. dpunkt, Heidelberg
- Schweitzer M (1994) Industriebetriebslehre – Das Wirtschaften in Industrieunternehmungen. Vahlen, München
- West J (2003) How open is open enough? Melding proprietary and open source platform strategies. *Res Policy* 32:1259–1285
- West J (2006) The economic realities of open standards: black, white and many shades of gray. In: Greenstein S, Stango V (Hrsg) Standards and public policy. Cambridge University Press, Cambridge
- Wheelwright SC, Clark KB (1992) Creating project plans to focus product development. *Harv Bus Rev* 70:67–83
- Yoffie DB, Kwak M (2006) With friends like these: the art of managing complementors. *Harv Bus Rev* 84(9):88–98

7.1 Überblick

In der letzten Zeit hat sich mit Cloud Computing ein neues Modell zur Bereitstellung von Services etabliert. Wie bei solchen neuen Entwicklungen häufig zu beobachten, überbieben sich Experten, Anbieter und Marktforschungsgesellschaften mit einer Vielzahl neuer Definitionen. Wir wollen hier keine weitere hinzufügen, sondern greifen auf die Definition des National Institute of Standards and Technology zurück. Demnach handelt es sich bei Cloud Computing um ein Modell, „das einen komfortablen, bedarfsabhängigen und netzbasierten Zugriff auf eine gemeinsam benutzte Menge konfigurierbarer Rechenressourcen ermöglicht, die schnell, mit geringem Verwaltungsaufwand und ohne (menschliche) Interaktion mit einem Anbieter bereitgestellt und wieder freigegeben werden können“ (National Institute of Standards and Technology 2009).

Dabei basiert Cloud Computing auf dem Prinzip der Virtualisierung und Verteilung IT-basierter Serviceleistungen. Aus ökonomischer Perspektive haben die Anbieter den Vorteil, dass sie verfügbare Ressourcen besser ausnutzen und damit angebots-seitige Skaleneffekte realisieren können. Daher ist es auch nicht verwunderlich, dass insbesondere große Anbieter sich diesen Markt erschließen, wie beispielsweise Amazon, Google und Microsoft. Dabei wird das Spektrum von Cloud-Angeboten häufig unterschieden, wie in der folgenden Abbildung dargestellt wird (Abb. 7.1).

Software as a Service (SaaS) bezeichnet Softwarelösungen auf der Anwendungsebene, die grundsätzlich geeignet sind, in einer Cloud angeboten zu werden (siehe hierzu ausführlich Abschn. 7.4). Cloud-Anbieter offerieren darüber hinaus auf der Plattformebene (Platform as a Service) beispielsweise den Betrieb von Middleware-Lösungen sowie Werkzeuge zur Softwareentwicklung. Wie klassische Outsourcing-Provider auch, bieten Cloud-Anbieter Hardware- und Infrastrukturdienste, wie etwa Server- und Speicherkapazitäten sowie Rechenleistungen, an (Infrastructure as a Service).

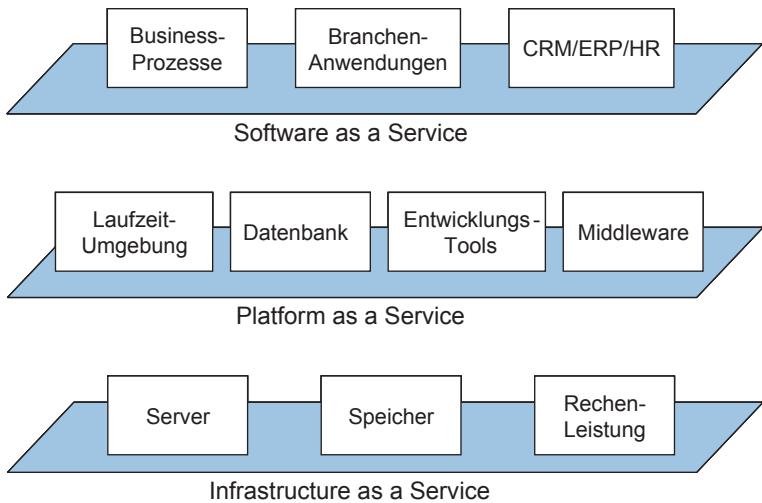


Abb. 7.1 Angebotsspektrum Cloud Computing. (Vaquero et al. 2009)

Darüber hinaus lässt sich die Form, wie Cloud-Dienste bereitgestellt werden, unterscheiden in

- Public Clouds,
- Private Clouds,
- Hybrid Clouds und
- Community Clouds.

Als Public Clouds bezeichnet man Ressourcen-Pools, die von unterschiedlichen Kunden gemeinsam genutzt werden. Die Kunden können hier ihren Bedarf meist flexibel erweitern. Als Pionier in diesem Bereich des Cloud-Markts konnten sich die Amazon Web Services (AWS) etablieren. Neben seinem Kerngeschäft, dem Internetversandhandel, hat der Amazon-Konzern mit der Gründung der AWS eine Möglichkeit geschaffen, die temporär ungenutzten Ressourcen seiner riesigen Serverparks stärker auszulasten. Schätzungen zufolge erwirtschaftete AWS mit seinen Services im Jahr 2013 3,8 Mrd. \$ Umsatz (Kalenda 2013). In Zukunft verspricht sich Amazon von AWS ein stark wachsendes Umsatz- und Gewinnpotenzial.

Eine andere Erfolgsgeschichte ist Dropbox. Das Unternehmen wurde 2007 gegründet und wächst seitdem rasant. Der Service von Dropbox besteht darin, dass Kunden Dokumente, die in der Dropbox Cloud liegen, überall dort tauschen können, wo eine Internetverbindung besteht. Dropbox wird deshalb auch häufig als Internet-Festplatte bezeichnet. Grundlage des Geschäfts ist ein Freemium-Modell, bei dem die Kunden in Abhängigkeit des genutzten Speicherplatzes zahlen. Bis zu 2 GB Speicherplatz können kostenlos genutzt werden.

Im Gegensatz zu Public Clouds sind Private Clouds geschlossene Systeme, die beispielsweise durch eine firmeneigene Infrastruktur realisiert werden. Ein anschauliches Beispiel für dieses Prinzip ist Owncloud – eine Open Source Software zur zentralen Speicherung und Bearbeitung von Daten, wie Bildern, Kalendereinträgen und Kontakten. Sie wurde als eine Alternative zu kommerziellen Cloud-Anbietern entwickelt und erlaubt dem Benutzer die vollständige Kontrolle über seine Daten, da die Software auch auf einem eigenen Server betrieben werden kann. Private Clouds sind physisch stärker begrenzt als Public Clouds und haben somit tendenziell eine geringere Flexibilität. Auf der anderen Seite bieten sie eine höhere Informationssicherheit, da u. a. bekannt ist, wo die Daten gespeichert werden. Dabei ist das Modell der Private Cloud in der praktischen Anwendung eng verwandt mit klassischen intern betriebenen Client-Server-Lösungen. In vielen Fällen wird lediglich das Label der „Cloud“ verwendet.

Bei einer Hybrid Cloud werden beide Cloud-Typen zusammen genutzt. Hier können beispielsweise bei Spitzenlasten externe Ressourcen hinzugeschaltet werden (Buxmann und Hofmann 2011, S. 826).

Zuletzt gibt es noch Community Clouds. Bei einer Community Cloud schließen sich Unternehmen oder Organisationen zusammen und bilden aus ihren Private Clouds eine Community Cloud. Diese Cloud ist nur für die Mitglieder der Community zugänglich. Community Clouds sind insbesondere sinnvoll, wenn die Mitglieder der Community gleiche Anforderungen und Aufgaben haben und die bestehende Infrastruktur gemeinsam nutzen wollen.

Bei der Analyse des Themas Cloud Computing muss man sich darüber im Klaren sein, dass es sich um nichts revolutionär Neues handelt. Die Anbieter offerieren ihren Kunden Outsourcing-Angebote, wie das viele andere Anbieter seit Jahren tun. Auch das Prinzip der Virtualisierung ist nicht wirklich neu. Dennoch gehen wir davon aus, dass der Cloud die Zukunft gehören wird, da es den Cloud-Computing-Anbietern zunehmend gelingt, ihren Kunden Lösungen anzubieten, deren Kosten signifikant unter denen des Eigenbetriebs liegen.

Im Folgenden wollen wir uns deshalb mit dem Cloud-Markt auseinandersetzen, um Wachstumsentwicklungen zu identifizieren und die Geschäftsmodelle von verschiedenen Cloud-Anbietern besser zu verstehen.

7.2 Der Cloud-Markt

Die Nachfrage nach Cloud-Angeboten in Deutschland wächst stetig. In einer Untersuchung des Bundesverbands Informationswirtschaft, Telekommunikation und neue Medien e. V. (Wallraff et al. 2013) und des Beratungsunternehmens KPMG im Jahr 2013 zeigte sich, dass zwar nach wie vor überwiegend große Unternehmen dem Thema Cloud Computing besonders aufgeschlossen gegenüberstehen, aber nun auch im Mittelstand ein Umdenken einsetzt. Abbildung 7.2 fasst den Anteil der jeweiligen Einstellungen für unterschiedliche Unternehmensgrößen zusammen.

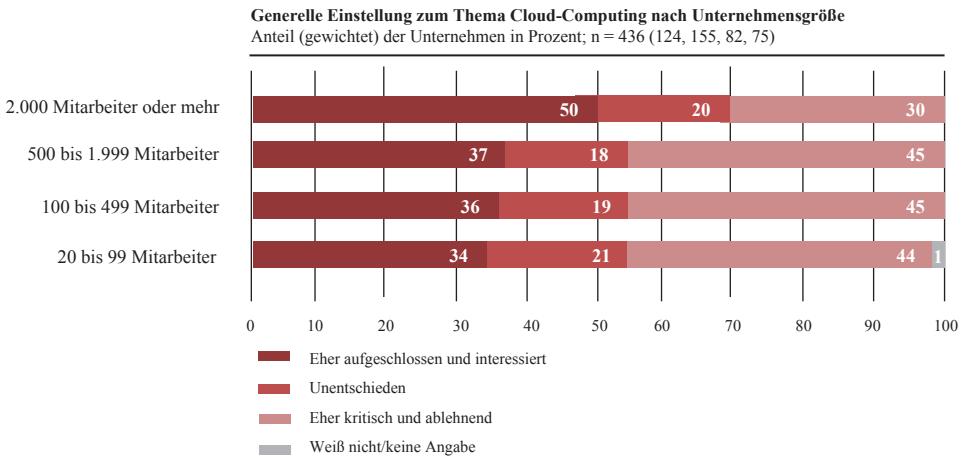


Abb. 7.2 Generelle Einstellung zum Thema Cloud-Computing nach Unternehmensgröße. (Wallraff et al. 2013, S. 8)

In Bezug auf einzelne Industrien zeigt sich, dass die Branchen der Informationstechnologie und Telekommunikation sowie des Finanzsektors eher aufgeschlossen gegenüber der Nutzung von Cloud Computing sind, während der Automobilbau, der Maschinen- und Anlagenbau sowie der Groß- und Einzelhandel ein eher geringes Interesse haben. Die Cloud-Nutzer scheinen unter dem Strich jedoch grundsätzlich zufrieden zu sein. So bewerten etwa 75 % ihre generelle Erfahrung mit der Cloud-Nutzung als positiv (Wallraff et al. 2013, S. 9).

Auch für die Zukunft scheint sich ein erfreuliches Bild für Cloud-Anbieter abzuzeichnen, wie die Prognose der Umsätze von Cloud-Anbietern bis 2016 des Statistischen Bundesamts zeigt.

Die Abb. 7.3 macht deutlich, dass bis 2016 wachsende Umsätze für Cloud-Anbieter erwartet werden, wobei diese sich von 2014 bis 2016 sogar verdoppeln sollen (Statista 2014c). Aktuell nutzt oder plant bereits die Mehrheit der deutschen Unternehmen den Einsatz von Cloud-Angeboten. Dabei spielen für Unternehmen vor allem private Cloud-Lösungen eine Rolle. Bei großen Unternehmen nutzen bereits 61 % Private-Cloud-Angebote und 20 % Public-Cloud-Computing. Für Public-Cloud-Nutzer insgesamt zeigt sich zudem, dass zwei Drittel der Nutzer zusätzlich zur Public Cloud Private-Cloud-Angebote verwenden. Die gemeinsame Nutzung von Public- und Private-Cloud-Angeboten als hybride Form scheint demnach an Bedeutung zu gewinnen (Wallraff et al. 2013, S. 10).

Betrachtet man die Marktanteile bei den Public-Cloud-Anbietern, so zeigt sich, dass Amazon Web Services mit 70 % Marktanteil die stärkste Kraft im Markt ist. Jedoch haben auch Mitwettbewerber wie Microsoft und Google die Zeichen der Zeit erkannt und wollen mit überarbeiteten Plattformen größere Marktanteile gewinnen. In diesem Zusammenhang ist auch die Übernahmestrategie der SAP interessant, die in den letzten zwei Jahren Cloud-Anbieter, wie etwa Ariba, Success Factors sowie Fieldglass übernommen hat.

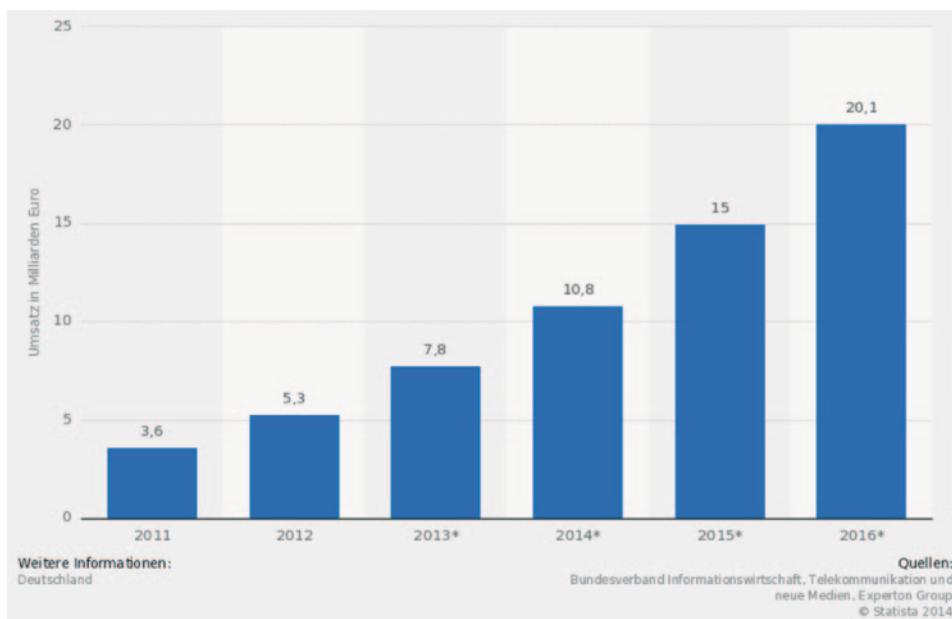


Abb. 7.3 Prognose zum Umsatz von Cloud-Anbietern in Deutschland bis 2016 in Milliarden Euro. (Statista 2014c)

Bei einer Analyse der Geschäftsmodelle zeigt sich, dass sich insbesondere auf der Software-as-a-Service-Ebene nutzungsabhängige Preismodelle nicht etabliert haben. Die Abrechnung erfolgt häufig auf der Basis der Anzahl der Nutzer. Wir gehen in Abschn. 7.4.3 näher darauf ein. Demgegenüber finden sich auf den unteren Ebenen, insbesondere im Bereich Infrastructure as a Service, häufig nutzungsabhängige Bezahlmodelle, beispielsweise auf Grundlage des genutzten Speicherplatzes. Diese Modelle werden häufig mit dem Freemium-Ansatz kombiniert (siehe hierzu Abschn. 3.3 dieses Buches).

7.3 Die Sicherheit in der Cloud – Empirische Ergebnisse aus Anwender- und Anbieterperspektive

Die Sicherheit in der Cloud ist spätestens seit Bekanntwerden der NSA/PRISM-Affäre für viele Nutzer der Technologie ein Thema. Viele Anwender haben verunsichert auf den Skandal reagiert und sind seitdem besorgt um die Sicherheit ihrer Daten. Dies gilt sowohl für den B2C- als auch den B2B-Bereich. Schätzungen gehen davon aus, dass US-amerikanische Anbieter bis 2016 insgesamt 35 Mrd. \$ Umsatzverluste hinnehmen werden müssen (Welt 2013). Vor diesem Hintergrund wollen wir uns im Folgenden mit den unterschiedlichen Sichtweisen der Anwender und Anbieter auf die Sicherheit in der Cloud auseinandersetzen. Im Fokus steht das B2B-Geschäft.

7.3.1 Anwendersicht

Cloud Computing bietet aus Anwendersicht viele attraktive Vorteile, wie das Einsparen von Kosten oder eine effiziente Nutzung von Ressourcen (siehe hierzu Abschn. 7.4.1). Die Technologie birgt jedoch auch einige Risiken, die trotz massiver Bemühungen in der Vergangenheit bereits zu schwerwiegenden Problemen bei einigen Anbietern geführt haben. So hat beispielsweise der Zusammenbruch der Amazon EC2 Cloud Services im April 2011 zu einem enormen Datenverlust für hunderte von Kunden geführt. Auch war es möglich, dass durch einen Blitzschlag im August 2011 der Microsoft Cloud-Computing-Service „Business Productivity Online Suite“ ausgefallen ist, so dass betroffene Kundenunternehmen für 48 h nicht in der Lage waren, ihre Emails, den Kalender, die Kontakte und das Dokumenten-Management-System zu nutzen. Diese Vorfälle zeigen letztendlich die grundsätzliche Verletzlichkeit zentraler Systeme.

Andererseits muss man sich darüber im Klaren sein, dass derartige Vorfälle natürlich auch beim Betrieb hauseigener Systeme und Server auftreten – und zwar ohne, dass diese dann durch klassische Medien oder soziale Netzwerke in der Öffentlichkeit bekannt gemacht werden. Vor diesem Hintergrund wollen wir uns im Folgenden mit dem wahrgenommenen IT-Sicherheitsrisiko aus Anwendersicht beschäftigen. Wir haben in diesem Kontext eine Studie durchgeführt, welche die wahrgenommenen IT-Sicherheitsrisiken von Anwendern untersucht (Ackermann et al. 2012). Im Rahmen einer ausführlichen Literaturrecherche und mehreren Einzelgesprächen wurden 31 Einzelrisiken identifiziert, die sich in sechs Dimensionen einordnen lassen (Tab. 7.1).

Auf dieser Basis wurde eine großzahlige Studie mit 356 IT-Verantwortlichen aus Anwendungsunternehmen durchgeführt. Die Zielsetzung bestand zum einen darin zu identifizieren, welche Einzelrisiken und Risikodimensionen einen Einfluss auf die Nutzung von Cloud-Angeboten haben. Zum anderen wollten wir herausfinden, inwieweit das gesamte wahrgenommene IT-Sicherheitsrisiko die Nutzung bzw. Adoption von Cloud-Lösungen beeinflusst. Bei 63 % der Befragten handelte es sich um CEOs oder CIOs und 84 % der Teilnehmer waren direkt für die Auswahl und Entscheidung des jeweiligen Applikationstyps verantwortlich. Im Rahmen unserer Studie wurden die Teilnehmer gebeten, die 31 Sicherheitsrisiken und sechs Risikodimensionen auf einer 7-stufigen Likert Skala einzurunden (1 = „überhaupt nicht riskant“ bis 7 = „überaus riskant“).

Die Auswertung der Daten für die 31 Sicherheitsrisiken hat gezeigt, dass im Durchschnitt die Entscheidungsträger im Unternehmen vor allem einen Identitätsdiebstahl, Angriffe auf die Verfügbarkeit und das Einsehen von Daten beim Anbieter als besonders riskant einschätzen. Die höchstbewerteten IT-Sicherheitsrisiken (Top 10) aus Sicht der Anwender finden sich in der folgenden Abb. 7.4.

In Bezug auf die sechs Risikodimensionen hat sich im Durchschnitt ergeben, dass Vertraulichkeit als wichtigste Dimension wahrgenommen wird, gefolgt von Verfügbarkeit, Zurechenbarkeit (Verantwortlichkeit) und Integrität. Zuletzt wurden Leistung (Performance), gefolgt von Wartbarkeit und hier vor allem auch die Unterkategorie Anpassbarkeit genannt.

Tab. 7.1 IT-Sicherheitsrisikodimensionen mit jeweiligen Ausprägungen. (Ackermann et al. 2012)

	<i>Vertraulichkeitsrisiko</i>		<i>Performanzrisiko</i>
1	Abhören der Übertragung	16	Geschwindigkeitsprobleme
2	Datenweitergabe durch den Anbieter	17	Unzureichende Anpassbarkeit
3	Einsehen von Daten beim Anbieter	18	Bewusste Minderleistung nach Vertragsabschluss
4	Einsehen von Daten auf internen System	19	Geschwindigkeitsprobleme interner Systeme
	<i>Integritätsrisiko</i>		<i>Verantwortlichkeitsrisiko</i>
5	Manipulation von übertragenen Daten	20	Identitätsdiebstahl
6	Datenmanipulation durch den Provider	21	Unzureichende Trennung von Kunden
7	Datenmanipulation während der Übertragung	22	Unzureichende Protokollierung von Aktionen
8	Datenmanipulation beim Anbieter	23	Zugriff ohne Autorisation
9	Ändern von Daten auf internen Systemen	24	Nichtzurechenbarkeit bei internen Aktionen
	<i>Verfügbarkeitsrisiko</i>		<i>Wartbarkeitsrisiko</i>
10	Einstellung des Angebots	25	Unzureichende Anpassbarkeit
11	Ungewollte Ausfallzeiten	26	Inkompatible Geschäftsprozesse
12	Angriffe auf die Verfügbarkeit	27	Fehlender technologischer Fortschritt
13	Verlust von Zugriff auf Daten	28	Unzureichender Datenimport
14	Datenverluste beim Anbieter	29	Proprietäre Technologien
15	Nichtverfügbarkeit von internen Systemen	30	Unzureichende Wartung
		31	Zeitlich ungünstige Updates

Die Auswertung der Daten hat gezeigt, dass alle Pfadkoeffizienten der Risikodimensionen statistisch signifikant sind, so dass mithilfe der Risikodimensionen das Konstrukt des wahrgenommenen IT-Sicherheitsrisikos der Software-Anwender erklärt werden kann.

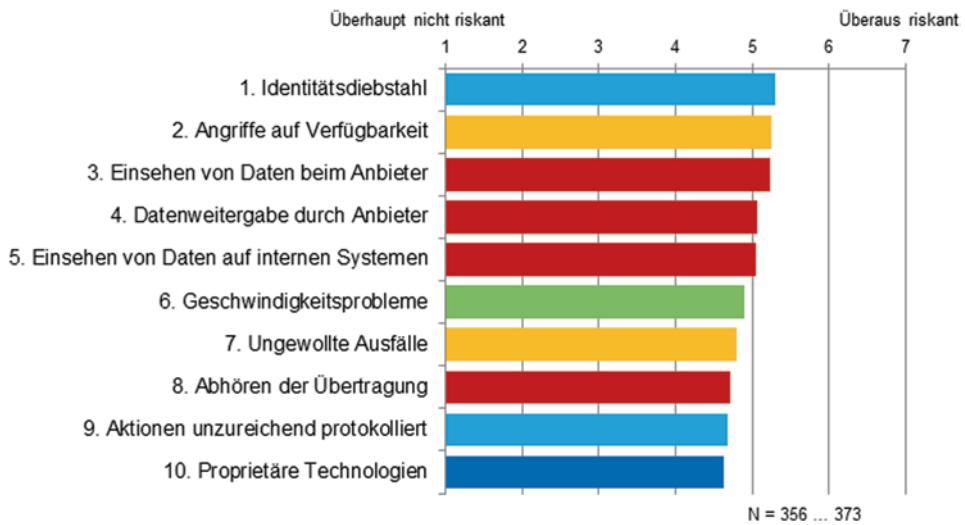


Abb. 7.4 Top 10 IT-Sicherheitsrisiken des Cloud Computing aus Anwendersicht. (in Anlehnung an Ackermann et al. 2012)

Auch der hohe Anteil der erklärten Varianz für das Konstrukt ($R^2=0,60$) belegt diese Aussage. Darüber hinaus konnten wir zeigen, dass durch den statistisch signifikanten Pfadkoeffizienten von $-0,53$ und einem $R^2=0,28$ das wahrgenommene IT-Sicherheitsrisiko die Absicht, Cloud Computing im Unternehmen verstärkt einzusetzen, beeinflusst. Abbildung 7.5 gibt eine Übersicht über die Pfadkoeffizienten und R^2 -Werte des Modells.

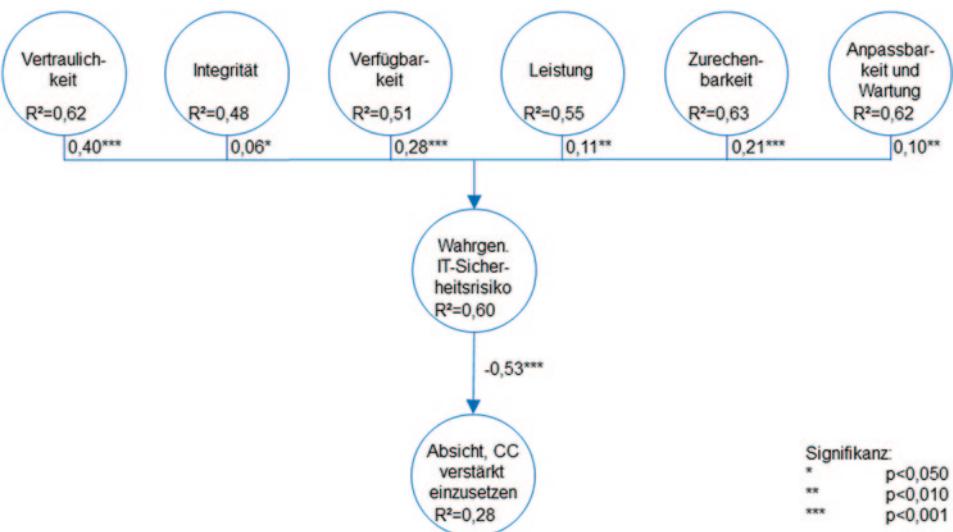


Abb. 7.5 Einfluss der wahrgenommenen IT-Sicherheitsrisiken auf die Absicht Cloud Computing verstärkt einzusetzen. (in Anlehnung an Ackermann et al. 2012, S. 10)

Wir können somit also zeigen, dass das wahrgenommene – nicht das tatsächlich existierende – IT-Sicherheitsrisiko von Cloud-Computing-Lösungen die Adoption dieser Systeme signifikant negativ beeinflusst. Diese Ergebnisse sind auch relevant für die Anbieter, deren Ziel darin bestehen sollte, Vertrauen bei den Anwendern aufzubauen. Mit dieser Anbieterperspektive wollen wir uns im Folgenden näher beschäftigen.

7.3.2 Anbietersicht

Um ein besseres Verständnis für die Wahrnehmung der IT-Sicherheit in der Cloud zu erhalten, wurde eine weitere Untersuchung mit 73 Cloud-Anbietern durchgeführt (Loske et al. 2013). Die Mehrheit der Befragten waren Führungskräfte, vor allem CIOs und CEOs. Die Teilnehmer der Studie wurden gebeten, ähnlich wie zuvor die Anwender, auf einer 7-stufigen Likert-Skala die im vorigen Abschnitt genannten 31 IT-Sicherheitsrisiken zu bewerten (1 = „überhaupt nicht riskant“ bis 7 = „überaus riskant“). Für die befragten Führungskräfte waren Identitätsdiebstahl, Angriffe auf die Verfügbarkeit sowie inkompatible Geschäftsprozesse und Software die schwerwiegendsten IT-Sicherheitsrisiken.

Die teilnehmenden Führungskräfte wurden auch gebeten anzugeben, wie sie die 31 IT-Sicherheitsrisiken in Bezug auf das eigene Unternehmen wahrnehmen und wie sie sie in Bezug auf einen durchschnittlichen Wettbewerber sehen. Das Ergebnis: Die Befragten schätzen die Risiken für das eigene Unternehmen durchweg als geringer als für den durchschnittlichen Wettbewerber ein. Eine genaue Gegenüberstellung der wahrgenommenen IT-Sicherheitsrisiken für das Unternehmen der Befragten und den durchschnittlichen Wettbewerber liefert Abb. 7.6.

Um ein noch besseres Verständnis für die Risikowahrnehmung der Anbieter zu entwickeln, haben wir deren Risikoeinschätzung mit der Anwenderperspektive verglichen. Grundlage sind die bereits identifizierten 31 IT-Sicherheitsrisiken. Die Ergebnisse sind in der folgenden Abb. 7.7 zusammengefasst.

Die Abbildung verdeutlicht, dass die Anbieter das Risiko für alle Risikodimensionen immer niedriger einschätzten als die potenziellen Anwender selbst. Dies hat gemäß der Theorie der „Perceptual Differences“ erhebliche Konsequenzen (Loske et al. 2014): Weicht das wahrgenommene IT-Sicherheitsrisiko der potenziellen Anwender stark vom wahrgenommenen Risiko der Anbieter ab, also bietet der betrachtete Anbieter aus Sicht der Anwender zu wenige Sicherheitsmaßnahmen an, so verringert sich die Absicht der potenziellen Anwender, Cloud Computing einzusetzen. Die potenziellen Anwender gehen möglicherweise davon aus, dass die für sie relevanten Risiken vom jeweiligen Anbieter nicht ausreichend berücksichtigt werden bzw. ihre Sorgen nicht ernst genug genommen werden, so dass der Einsatz von Cloud-Angeboten des betrachteten Anbieters als zu risikant erscheint. Besonders negativ wirken sich Unterschiede in der Wahrnehmung von Vertraulichkeitsrisiken und Verfügbarkeitsrisiken auf die Absicht der Anwender Cloud Computing zu nutzen aus. Darüber hinaus konnte gezeigt werden, dass Anwender die wahrgenommenen IT-Sicherheitsrisiken und das allgemeine Risiko höher bewerten, wenn

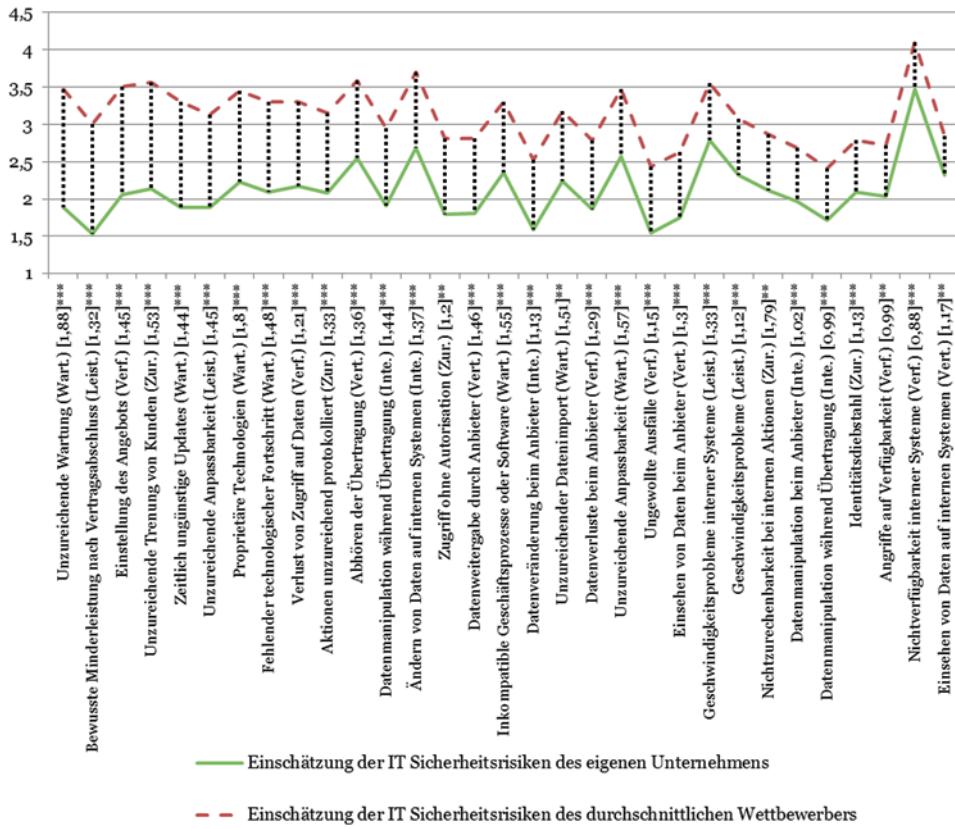


Abb. 7.6 Einschätzung der IT-Sicherheitsrisiken für den befragten Anbieter und einen durchschnittlichen Wettbewerber. (Loske et al. 2013, S. 9)

ihre Erwartungen an die Sicherheitsvorkehrungen des Anbieters nicht erfüllt werden. Insgesamt wird deutlich, dass die Unterschiede in der Wahrnehmung von Anbietern und potenziellen Anwendern sich essenziell direkt und indirekt negativ auf die Bereitschaft der Nutzer auswirkt Cloud Computing einzusetzen (Loske et al. 2014, S. 12).

Nachdem wir uns in den vorigen Abschnitten dieses Kapitels ausführlich allgemein mit Cloud Computing beschäftigt haben, wollen wir uns im Folgenden mit SaaS als Anwendungsebene des Cloud Computing auseinandersetzen, da es für unser Buch zur Software-industrie besonders relevant ist.

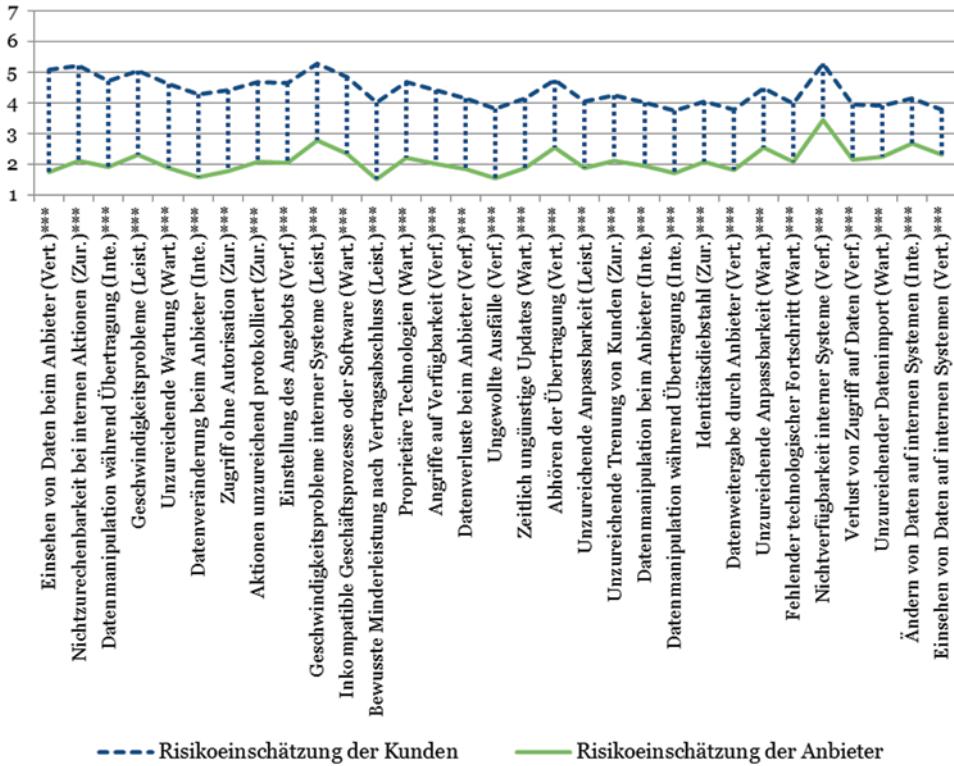


Abb. 7.7 Risikoeinschätzung für die Anwender aus Anwender- und Anbietersicht. (Loske et al. 2014)

7.4 Software as a Service: die Anwendungsebene des Cloud Computing

Der Bezug von Software als Dienstleistung, kurz „Software as a Service“ (SaaS), gilt derzeit als wichtiger Trend und steht auf der Agenda vieler IT-Führungskräfte. Bei der Nutzung von SaaS wird den Kunden eine Standardsoftwarelösung als Dienstleistung über das Internet zur Verfügung gestellt. Der SaaS-Anbieter ist für Betrieb und Wartung der mehrmandantenfähigen Software verantwortlich. Dabei erzielen die Anbieter keine Lizenzentnahmen. Vielmehr zahlen die Anwender für die gemieteten Softwarekomponenten und Serviceleistungen Nutzungsgebühren, die in der Regel monatlich, quartalsbasiert oder jährlich anfallen. Darüber hinaus sind aus Sicht der Software- und Serviceanbieter auch andere Erlösmodelle, wie z. B. Werbeeinnahmen oder eine nutzungsabhängige Abrechnung, denkbar.

Um es vorwegzunehmen: Die Idee hinter SaaS ist, genauso wie die des Cloud Computing, nicht grundlegend neu. Daher wird sie von Kritikern auch häufig als „Alter Wein in neuen Schläuchen“ bezeichnet. Bereits in den 90er-Jahren wurde ein ähnlicher Ansatz unter der Bezeichnung Application Service Providing (ASP) verfolgt (Günther et al.

2001). Bei SaaS handelt es sich letztlich um eine Erweiterung von ASP, die insbesondere aufgrund der Entwicklung und Verbreitung innovativer Internet-Technologien und -Standards Potenzial besitzt und neue Möglichkeiten für Anwender und Anbieter eröffnet. So benötigen die Anwender heute in den meisten Fällen lediglich einen Internet-Zugang und einen Webbrower, um SaaS-Lösungen nutzen zu können. Demgegenüber waren früher zum Teil hohe Investitionen und Know-how erforderlich, um entsprechende Serviceangebote von ASP-Herstellern zu nutzen. Für die Anwender bedeutet das, dass der Umstieg auf eine SaaS-Lösung in der Regel einfacher und damit auch kostengünstiger als früher sein wird. Darüber hinaus erleichtern serviceorientierte Architekturen und offene Standards, wie etwa Web-Service-Protokolle, die Integration von SaaS-Lösungen in Inhouse-Systeme sowie eine Zusammenarbeit mit anderen Services. Dass es sich bei SaaS um nichts revolutionär Neues handelt, bedeutet also nicht, dass sich dieses Konzept nicht durchsetzen wird oder für Anbieter und Anwender wenig relevant ist. Grundsätzlich ist SaaS für eine Vielzahl von Einsatzgebieten anwendbar. Besonders geeignet scheint dieses Geschäftsmodell jedoch für Funktionen und Prozesse zu sein, die sich zu einem hohen Grad standardisieren lassen. Hierzu gehören etwa Bereiche wie CRM-Software.

Vor diesem Hintergrund ist es nicht erstaunlich, dass sich mit Salesforce.com ein Anbieter von On-Demand-Geschäftsanwendungen sowie Betreiber einer Cloud-Lösung im CRM-Bereich als SaaS-Pionier auf dem Cloud-B2B-Markt mit einem Umsatz von ca. 3 Mrd. etabliert hat (Kurzlechner 2013). Die Erlöse werden, wie im Bereich SaaS üblich, nicht durch den Verkauf von Softwarelizenzen, sondern durch so genannte „Subscription Fees“ für die von den Kunden gemieteten Komponenten des CRM-Systems generiert. Dabei werden die vom Kunden gewünschten Softwarekomponenten zusammen mit der gesamten Infrastruktur, dem Support und Service über das Internet bereitgestellt.

Die CRM-Anwendung von Salesforce.com wird in fünf verschiedenen Versionen angeboten (Contact Manager, Group Edition, Professional Edition, Enterprise Edition und Unlimited Edition). Die monatliche Gebühr für die einzelnen Versionen ergibt sich aus der Anzahl der lizenzierten Benutzer und der jeweiligen monatlichen Kosten für eine Version (4 € für den Contact Manager und 270 € für die Unlimited Edition pro Nutzer und Monat). Dahinter steht die Strategie der Preisdifferenzierung, die wir in Abschn. 3.3.2.5 dieses Buches diskutiert haben.

Seit 2007 bietet Salesforce.com mit Force.com zudem eine On-Demand-Plattform für webbasierte Applikationen an. Anwender sowie Entwickler können so auf Basis der Salesforce-Infrastruktur individuelle Anwendungen entwickeln, die sie entweder selbst nutzen oder über den Marktspiel AppExchange anbieten können.

Auch die SAP hat sich ursprünglich mit Business ByDesign und etwas später durch mehrere Unternehmensübernahmen positioniert und strebt an, ebenfalls im Marktsegment Cloud Services für Unternehmenssoftware weltweit die Nummer 1 zu werden.

Andere bekannte Lösungen im Bereich Software as a Service sind etwa Google Apps oder Office 365 von Microsoft. Hier zeigt sich, dass die etablierten Player in der Software-industrie einen Wechsel einleiten und die Cloud als Zukunftschance sehen. Auch Analys-

ten strafen Softwareanbieter zunehmend ab, wenn ihnen die Investitionen in zukunftsfähige Cloud-Lösungen zu gering erscheinen.

Doch wie wird die Nutzung bzw. Bereitstellung von SaaS-Lösungen bewertet? Im Folgenden wollen wir deshalb die Chancen und Risiken aus Anwender- und Anbietersicht näher beleuchten.

7.4.1 SaaS aus Anwendersicht – Chancen und Risiken

7.4.1.1 Grundüberlegungen

Bei der Inanspruchnahme eines SaaS-Angebots handelt es sich um Outsourcing, d. h. die Auslagerung von Funktionen oder Prozessen an Dritte (Buxmann et al. 2008b). Dies bedeutet auch, dass sich einige potenzielle Vor- und Nachteile der SaaS-Nutzung grundsätzlich aus denen des Outsourcings ableiten lassen. Vor diesem Hintergrund wollen wir nun die Chancen und Risiken des Einsatzes von SaaS-Lösungen näher betrachten. Basierend auf Voruntersuchungen im traditionellen IT-Outsourcing (Earl 1996), im ASP-Markt (Kern et al. 2002) sowie in ersten Untersuchungen zur Adoption von SaaS-Anwendungen (Benlian und Hess 2009) lassen sich jeweils fünf Chancen- und Risikokategorien für den Einsatz von SaaS-Anwendungen ableiten (Benlian und Hess 2010). Tabelle 7.2 gibt eine Zusammenfassung dieser Gegenüberstellung wieder.

Im Folgenden werden wir die in der Tabelle dargestellten Chancen und Risiken der SaaS-Nutzung näher betrachten.

Als Hauptvorteil des Einsatzes von SaaS-Lösungen werden aus Anwendersicht zu meist Kosteneinsparungen und Liquiditätsvorteile angeführt, da die Softwarelösungen nicht mehr auf den Servern der anwendenden Unternehmen installiert, getestet, weiterentwickelt bzw. gewartet werden müssen und zudem die einmaligen Lizenzkosten entfallen.

Bei der Nutzung einer SaaS-Lösung fallen periodisch konstante Kosten für Betrieb, Support und die Wartung der Lösung an. Häufig wird daher auch von einem Mietmodell gesprochen. Auch bei einem Update der Software entstehen bei den meisten Anbietern keine weiteren Kosten. Darüber hinaus ist jedoch davon auszugehen, dass die Anwender zusätzlich zur Softwaremiete Implementierungskosten zu tragen haben. Diese fallen etwa im Rahmen von Projekten zur technischen und organisatorischen Integration der SaaS-Lösung an. Eine besondere Herausforderung besteht in der Integration der SaaS-Lösung in bestehende Inhouse-Systeme. Demgegenüber haben die Anwender beim klassischen Modell neben den Implementierungskosten einmalige Lizenzgebühren zu zahlen (siehe hierzu Abschn. 3.3 des Buches). Zudem sind jährliche Support- und Wartungskosten zu tragen. Darüber hinaus sind im klassischen Modell Updatekosten zu berücksichtigen, die etwa alle sieben bis zehn Jahre anfallen.

In der Regel werden die Implementierungskosten (Hardware, Software, Geschäftsprozessanwendungen, Personalkosten), einschließlich Lizenzen, für eine klassische Standardsoftwarelösung höher sein als für eine SaaS-Lösung (Altmann et al. 2007, S. 40). Dies liegt u. a. daran, dass aufgrund der Betriebssystem- und Plattformunabhängigkeit in den

Tab. 7.2 Chancen und Risiken von SaaS aus Anwendersicht. (Quelle: Benlian und Hess 2010, erweiterte Darstellung)

Chancen		Risiken	
Kategorie	Beschreibung	Kategorie	Beschreibung
Kosten- und Liquiditätsvorteile	Chance, dass der Bezug von SaaS-Anwendungen zu niedrigeren Gesamtkosten und zu einer günstigeren Liquiditätslage führt	Finanzielle Risiken	Risiko, dass SaaS-Kunden letztlich mehr für die Bereitstellung der Anwendungen bezahlen (z. B. aufgrund von Internet-Ausfällen, erhöhten Anpassungskosten oder Preissteigerungen); Risiko durch höhere Opportunitätskosten dadurch, dass SaaS-Anwendungen in der Regel weniger gut an unternehmensspezifische Anforderungen angepasst werden können
Strategische und operative Flexibilität	Chance, dass SaaS-Kunden mehr Spielraum besitzen, den SaaS-Anbieter zu wechseln (z. B. durch kürzere Kündigungsfristen und geringere Abhängigkeit)	Strategische Risiken	Risiko, dass SaaS-Kunden unternehmenskritische Ressourcen oder Kenntnisse verlieren, wenn sie die Entwicklung und den Betrieb von Anwendungen auslagern
Qualitätsverbesserungen	Chance, dass SaaS-Anbieter gezwungen sind, kontinuierlich hohe Servicequalität zu liefern, da ihre Kunden die Möglichkeit haben, kurzfristig kündigen zu können	Operative Risiken	Risiko, dass SaaS-Anbieter die vereinbarten Service Levels im Sinne der Erreichbarkeit, Performance und Interoperabilität der SaaS-Anwendungen nicht erfüllen
Zugang zu spezifischen Ressourcen	Chance, dass SaaS-Kunden von den spezifischen Ressourcen, Fertigkeiten und Technologien des SaaS-Anbieters profitieren	Sicherheits-Risiken	Risiko, dass unternehmenskritische Daten an den SaaS-Anbieter übergeben und/oder kritische Prozesse beeinträchtigt werden

Tab. 7.2 (Fortsetzung)

Chancen		Risiken	
Kategorie	Beschreibung	Kategorie	Beschreibung
Konzentration auf Kern-kompetenzen	Chance, dass es SaaS-Kunden leichter fällt, sich auf ihre Kernkompetenzen zu konzentrieren, wenn sie die Entwicklung und den Betrieb von Anwendungen auslagern	Soziale Risiken	Risiko, dass es durch die Auslagerung von Anwendungen an einen Drittanbieter zu Mitarbeiterwiderständen oder negativer Presse kommt

meisten Fällen keine oder nur geringe zusätzliche IT-Kosten anfallen. Aus diesem Grund ist tendenziell auch von einer schnelleren Verfügbarkeit von SaaS-Lösungen auszugehen.

Dass die Implementierungskosten für SaaS-Lösungen in der Regel niedriger sein werden, liegt aber auch daran, dass die Customizing-Möglichkeiten für diese Lösungen meistens geringer sind als bei klassischen Standardsoftwarelösungen (Buxmann et al. 2008b). Insofern wird der Kostenvorteil von SaaS-Lösungen durch geringere Anpassungsmöglichkeiten an die organisatorischen Anforderungen relativiert.

Eine weitere potenzielle Chance durch den Bezug von SaaS wird häufig in der größeren (strategischen und operativen) Flexibilität für Unternehmen gesehen, den SaaS-Anbieter zu wechseln, etwa wenn die gewünschten Vertragsziele nicht erfüllt werden. Durch die Installation der Hard- und Software beim Anbieter entsteht tendenziell eine geringere technische Abhängigkeit für die Anwender. Sie müssen in der Regel geringere Investitionen in die eigene IT-Infrastruktur vornehmen und können grundsätzlich den Mietvertrag bei kürzeren Kündigungsfristen auch vorzeitig wieder beenden. Sofern die auf Seiten des SaaS-Anbieters für den Anwender gespeicherten Daten in einem offenen Datenformat vorgehalten wurden, ist eine Datenmigration ebenfalls relativ einfach zu bewerkstelligen. Gerade hier zeigen jedoch einige Fälle aus der Praxis, dass diese Flexibilität seitens der Anwender häufig auch Wunschenken ist.

Im Folgenden wollen wir den potenziellen Vorteil einer höheren Flexibilität in Bezug auf die Anbieterwahl etwas genauer betrachten. Wie bereits in Abschn. 2.2.4 dargestellt, wird für Anwender der Wechsel ihrer Standardsoftwarelösung in der Regel mit hohen Switching-Costs verbunden sein. Dies gilt insbesondere für ERP-Systeme, weshalb ein Wechsel des Anbieters in der Praxis relativ selten zu beobachten ist. Doch was ist der Grund für diese hohen Switching-Costs? In erster Linie sind es nicht die Lizenzkosten für eine alternative Softwarelösung. Vielmehr gilt gerade auf dem ERP-Markt, dass diese Software auch die Geschäftsprozesse der Anwender abbildet und möglicherweise auch gestaltet hat. Ein Wechsel des Anbieters würde deshalb auch erhebliche Kosten für Organisationsänderungen nach sich ziehen. Dieser Zusammenhang besteht grundsätzlich auch bei SaaS-Lösungen. Sobald diese Lösungen in die IT-Landschaft der Anwender integriert

werden, ist ein solcher Lock-in-Effekt nicht zu vermeiden. Dabei sind der entsprechende Lock-in und die damit verbundene Abhängigkeit vom Anbieter umso höher, je mehr in diese Integration investiert wurde. Aufgrund der geringeren Customizing-Möglichkeiten sowie dem häufigen Einsatz offener Standards im Rahmen von serviceorientierten Architekturen ist im Vergleich zu klassischer Standardsoftware jedoch tendenziell ein geringerer Lock-in in SaaS-Lösungen zu erwarten.

Die Chance auf Qualitätssteigerungen wird darin gesehen, dass SaaS-Anbieter durch geringere Wechselkosten auf Seiten der Anwenderunternehmen stärker gezwungen sind, auf Wünsche und Anforderungen ihrer Kunden einzugehen. Ein weiteres Indiz für Qualitätsverbesserungen wird in der Spezialisierung von SaaS-Anbietern gesehen. Darüber hinaus kann eine Qualitätssteigerung auch dadurch erzielt werden, dass den Anwendern Updates, Patches und Erweiterungen zeitnah eingespielt werden. Zudem wird der Anbieter in die Lage versetzt, die Benutzung der Software durch die User besser nachzuvollziehen. Insbesondere Anbieter sehen hierin die Chance, die Kundenbedürfnisse besser einzuschätzen und die SaaS-Lösungen kundenfreundlicher zu gestalten. Andererseits werden nicht immer alle Anwender begeistert davon sein, dass die Möglichkeit besteht, ihre Aktivitäten zu protokollieren und auszuwerten. Auch das automatische Einspielen von Updates wird – wie viele von uns geführte Gespräche mit Anwendern gezeigt haben – nicht ausschließlich positiv bewertet. Dies gilt insbesondere, wenn das Update zu einer neuen Benutzerführung führt, die möglicherweise von den Mitarbeitern nicht problemlos angenommen wird.

Im gleichen Atemzug mit möglichen Qualitätsverbesserungen wird der Zugang zu spezifischen Ressourcen, Fertigkeiten und Technologien genannt. Aufgrund von Spezialisierungsvorteilen haben die SaaS-Anbieter in der Regel eher Mittel und Möglichkeiten, um laufend in die neueste Generation an IT-Technologien zu investieren. Ferner kann ein SaaS-Anbieter seine Mitarbeiter voll und ganz auf die Bereitstellung von SaaS-Anwendungen spezialisieren und somit Expertenwissen aufbauen, das sich die Kunden schließlich zunutze machen können.

Schließlich wird in der Diskussion um die Vorteile von SaaS auch das klassische Argument angeführt, dass die Auslagerung von Softwareentwicklung, -anpassung und -wartung an einen spezialisierten Drittanbieter die Konzentration auf das Kerngeschäft des Unternehmens ermöglicht. Es werden Ressourcen in der IT-Abteilung freigesetzt (entweder in Form von Arbeitskräften oder Investitionsmitteln), die für strategische Aufgaben verwendet werden können. Beispielsweise können Routine-Support-Aufgaben an den SaaS-Anbieter ausgelagert werden, damit sich die eigenen IT-Mitarbeiter auf die Umsetzung von strategischen IT-Projekten konzentrieren können.

Demgegenüber stehen auf der Risikoseite finanzielle, strategische, operative, Sicherheits- und soziale Risiken. Finanzielle Risiken beziehen sich dabei insbesondere auf verdeckte Kosten, die bei Vertragsschluss oft noch nicht abgeschätzt werden können und im Outsourcing-Geschäft häufig auftreten. Diese können darin begründet liegen, dass Anwenderunternehmen spezialisierte System-Integratoren beauftragen müssen, um die Software an die unternehmensindividuellen Anforderungen anzupassen bzw. die SaaS-Anwendung mit intern bereitgestellten Anwendungen zu integrieren. Versteckte (künftige) Kosten kön-

nen aber auch dadurch anfallen, dass SaaS-Anbieter ihre Subskriptionspreise erhöhen, nachdem Unternehmen Anpassungsinvestitionen und Datenmigrationen vorgenommen haben, oder Extra-Kosten für unterschiedliche Zugriffskanäle auf die SaaS-Lösung (z. B. über mobile Endgeräte) berechnen. Nicht zuletzt können erhebliche Kosten (bzw. Umsatzeinbußen) bei Systemausfällen oder Schlechtleistung (z. B. langsame Internet-Verbindung) für das Anwenderunternehmen anfallen (siehe hierzu auch Abschn. 7.3.1).

Mit strategischen Risiken ist vor allem gemeint, dass sich ein Unternehmen mit der Herausgabe von kritischen Unternehmensressourcen in eine Abhängigkeitsbeziehung begibt, welche die Handlungsfähigkeit des Unternehmens einschränken könnte. Eine solche Situation liegt etwa dann vor, wenn Unternehmen nicht mehr flexibel auf eigene Strategiewechsel reagieren können, da sie die Kenntnisse über unternehmensspezifische Anpassungsmöglichkeiten der Software verloren haben. Eine Anpassung durch den SaaS-Anbieter oder durch System-Integratoren wäre in diesem Fall zwar auch denkbar, jedoch in der Realität nicht so schnell umsetzbar, um damit Wettbewerbsvorteile zu generieren.

Schließlich wird mit sozialen Risiken die Gefahr beschrieben, dass Mitarbeiter (z. B. in IT- oder Fachabteilungen bzw. Betriebsräten) gegen den Bezug von SaaS-Anwendungen über einen Drittanbieter, also gegen eine Herausgabe von (vermeintlich) kritischen Unternehmensfunktionen, Widerstand leisten. Dies könnte nicht nur zu Rufschädigungen in der öffentlichen Diskussion, sondern auch zu einer Arbeitsverweigerungshaltung innerhalb des Unternehmens führen. Diese Risikoform ist jedoch nicht SaaS-spezifisch und kann bei vielen organisatorischen Änderungen in Unternehmen beobachtet werden.

7.4.1.2 Empirische Untersuchung zu Chancen und Risiken der SaaS-Anwender

Zur Untersuchung der Chancen und Risiken der SaaS-Nutzung aus Anwendersicht haben wir eine empirische Untersuchung durchgeführt (Benlian und Hess 2010). Hierbei wurde aus der Hoppenstedt-Firmendatenbank eine zufallsbasierte Stichprobe von 2.000 Unternehmen gezogen und per E-Mail und schriftlichem Fragebogen im Juli 2009 angeschrieben. Der Fragebogen war an EDV/IT-Leiter bzw. CIOs/CTOs gerichtet, die zur Beantwortung der Fragen auch das notwendige Vorwissen mitbringen sollten. Von den 2.000 Unternehmen antworteten 349 Unternehmen (davon 142 SaaS-Kunden und 207 Nicht-Kunden) und sendeten insgesamt 922 vollständig ausgefüllte Fragebögen zurück. Jedes Unternehmen bewertete damit durchschnittlich zwei bis drei selbst eingesetzte Anwendungstypen hinsichtlich der Chancen und Risiken eines (aktuellen bzw. potenziellen) SaaS-Einsatzes.

Um sicherzustellen, dass sich die befragten Nicht-Kunden mit SaaS-Anwendungen weitestgehend auseinandergesetzt hatten, wurden sie danach gefragt, wie sie ihre SaaS-Kenntnisse einschätzen würden. Mehr als 85 % der Nicht-Kunden gaben an, dass sie mit SaaS-Lösungen sehr gut vertraut sind. Nur 5 % der befragten Unternehmen antworteten, dass sie das Prinzip von SaaS-Anwendungen zwar verstanden, sich speziell in ihrem Unternehmen jedoch bisher nicht näher mit dem Bezug von solchen Anwendungen auseinandergesetzt hätten.

Tab. 7.3 Stichprobenzusammensetzung. (Benlian und Hess 2010)

Kategorie	Prozent	Kategorie	Prozent
Anzahl Mitarbeiter		Umsatz in Mio. €	
< 10	27,3	< 1	28,2
10–49	25,4	1–9	41,3
50–99	20,8	10–99	16,9
> 99	26,5	> 99	13,6
Nutzung von SaaS-Anwendungen (in Jahren)		Positionen der Befragten	
0 (keine SaaS-Kunden)	59,4	Geschäftsführer, CEO, CIO	24,9
> 0 (aktuelle SaaS-Kunden)	40,6	IT/EDV-Leiter	62,5
Ich bin mit SaaS seit ... Jahren vertraut		Kaufmännische Leiter	8,4
< 2	17,3	Andere und n/a	4,2
> 2	82,7		

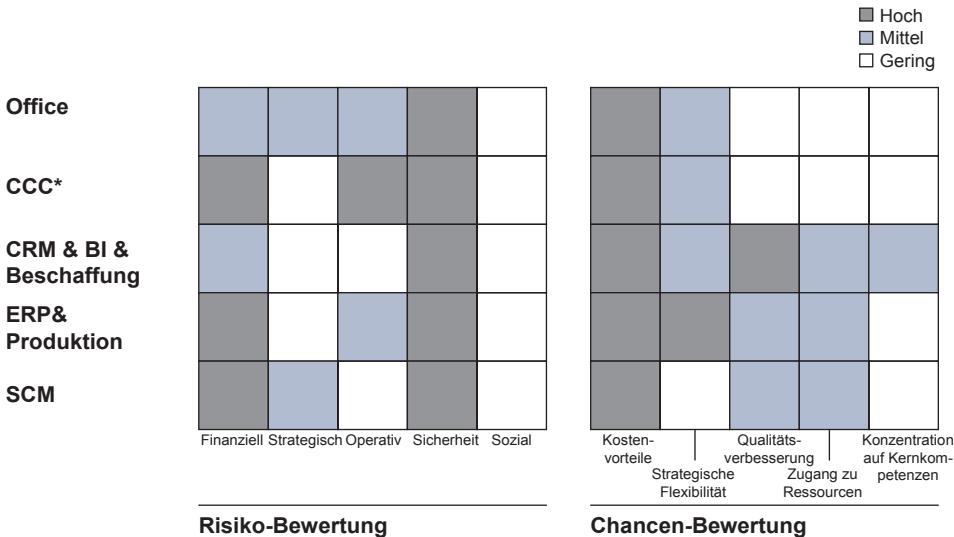
Die Stichprobe enthielt Unternehmen mit folgender Branchenzusammensetzung: Maschinenbau/Automobil, Groß- und Einzelhandel, Versicherungen/Banken, Telekommunikation/Information/Medien/Entertainment, Immobilien- und Bauwirtschaft, Logistik, Öffentlicher und Gesundheitssektor sowie Versorgungsunternehmen. Weitere Charakteristika der befragten Unternehmen werden in Tab. 7.3 dargestellt.

Zunächst wurden die teilnehmenden Unternehmen danach gefragt, ob bzw. inwieweit sie unterschiedliche SaaS-Anwendungen bereits heute einsetzen bzw. deren Einsatz in Zukunft planen. Hierbei wurde der Zeitraum von 2008 bis 2012 abgedeckt. Gemessen wurde der aktuelle bzw. geplante Einsatz von SaaS-Anwendungen über die Ausgaben für SaaS in Prozent des IT-Budgets für den jeweiligen Applikationstypen (z. B. ERP oder CRM) (Benlian und Hess 2010).

Die Ergebnisse zeigen, dass der Bezug von hochstandardisierten Anwendungssystemen, wie CRM- oder Office-Systemen, über SaaS eine größere Akzeptanz findet als der Einsatz für weniger standardisierte Anwendungssysteme. Die Ausgaben für hoch standardisierte Applikationstypen bewegen sich in einem Bereich zwischen 8–14 % des IT-Budgets für die Jahre 2008 bis 2009 und 23–35 % für 2010 bis 2012.

Weniger standardisierte Applikationstypen bewegen sich mit 0–3 % für 2008 bis 2009 und zwischen 4 und 11 % in den Jahren 2010 bis 2012 auf einem geringeren Adoptionsniveau. Aufgrund des Basiseffektes kommt es jedoch für weniger standardisierte Anwendungssysteme zu stärkeren Wachstumseffekten in den künftigen Perioden. Für ERP-Systeme konnten wir ein durchschnittliches Marktwachstum von 54 %, für SCM-Anwendungen sogar von 95 % identifizieren (Benlian und Hess 2010).

Im nächsten Schritt wurden die Unternehmen nach ihren Einschätzungen in Bezug auf Chancen und Risiken der SaaS-Nutzung gefragt. Die größte Chance sehen Unternehmen in kurz- bis mittelfristigen Kostenvorteilen. SaaS wird damit insbesondere als Kosten senkungshebel beurteilt. Daneben sehen Unternehmen noch mittlere bis große Flexibilitätsvorteile. Für CRM-, ERP- und SCM-Anwendungen ergeben sich laut der befragten



* Communication, Content, Collaboration

n=922

Abb. 7.8 Chancen-Risiken-Bewertung für unterschiedliche Applikationstypen. (Benlian und Hess 2010)

Unternehmen außerdem mittlere bis hohe Chancen durch Qualitätsverbesserungen und durch den Zugang zu spezifischen Ressourcen und Fertigkeiten. Interessanterweise wird der Einsatz von SaaS jedoch nicht als Möglichkeit gesehen, sich auf seine Kernkompetenzen fokussieren zu können.

Sicherheitsbedenken stellen gemäß den Ergebnissen der Studie die wesentlichste Risikoform über alle Applikationstypen hinweg dar. Zudem werden erhebliche finanzielle Risiken in den Applikationsmärkten CCC, ERP und SCM gesehen. Daraus lässt sich schließen, dass viele Unternehmen befürchten, dass der Einsatz von SaaS-Anwendungen versteckte Kosten mit sich bringen wird, die erst während der Vertragslaufzeit auftreten. Soziale Risiken im Sinne von Mitarbeiterwiderständen oder Rufschädigungen durch die Herausgabe kritischer Unternehmensteile wurden von Seiten der Unternehmen nicht gesehen.

Die Ergebnisse zu Chancen und Risiken der SaaS-Nutzung sind in der folgenden Abb. 7.8 dargestellt.

Vergleicht man die Chancen-Risiko-Bewertung von kleineren und mittleren Unternehmen (KMUs) mit großen Unternehmen, so zeigen sich nur geringe Unterschiede (siehe Abb. 7.9). Bei großen Unternehmen sticht allerdings insbesondere das strategische Risiko hervor, unternehmenskritisches Know-how an einen SaaS-Anbieter zu verlieren. Im Vergleich zu anderen Risiken überwiegen hingegen auf Seiten der KMUs die Bedenken, dass der Bezug von SaaS-Anwendungen über eine Internet-Schnittstelle zum Verlust von Daten bzw. zum Absturz der Internet-Verbindung führen könnte.

Auf der Chancenseite honorieren KMUs insbesondere den Vorteil, auf spezifische Ressourcen, Kompetenzen und Technologien zugreifen zu können, die sie selbst nicht be-

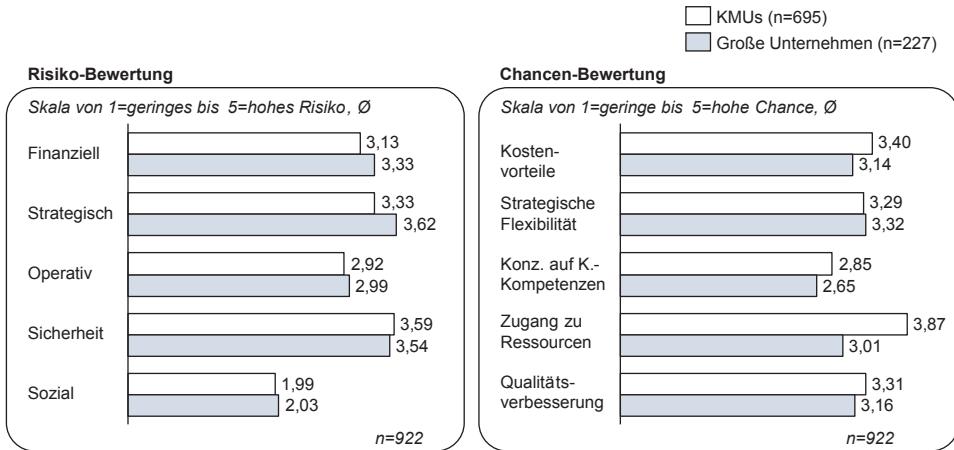


Abb. 7.9 Chancen-Risiken-Bewertung: Große Unternehmen vs. KMUs. (Benlian und Hess 2010)

sitzen. Ferner werden noch stärker als bei großen Unternehmen Kostenvorteile gesehen, die mit SaaS erzielt werden können. Große Unternehmen sehen im Gegensatz zu KMUs weniger die Kostenvorteile des Bezugs von SaaS-Anwendungen, sondern heben eher die strategische Flexibilität und potenzielle Chancen auf Qualitätsverbesserungen durch SaaS hervor. Weder KMUs noch große Unternehmen sehen große Vorteile der SaaS-Nutzung darin, dass sie sich besser auf ihre Kernkompetenzen konzentrieren können.

Interessante Unterschiede ergeben sich, wenn die Chancen- und Risiken-Einschätzungen von aktuellen SaaS-Kunden und Nicht-Kunden miteinander verglichen werden. Während Nicht-Kunden die Risiken von SaaS im Durchschnitt konsequent hoch einschätzen, sehen aktuelle SaaS-Kunden insbesondere die Vorteile durch SaaS (siehe Abb. 7.10).

Im Einzelnen überwiegen bei Nicht-Kunden insbesondere die Sicherheitsbedenken. Daneben werden finanzielle und operative Risiken als kritisch eingestuft. Aktuelle SaaS-Kunden schätzen ebenso Sicherheits- und finanzielle Risiken als kritisch ein. Risiken durch potenziellen Widerstand der Mitarbeiter gegen die Nutzung des SaaS-Modells und einhergehende Implikationen (z. B. Arbeitsplatzabbau) wurden wiederum als sehr gering eingeschätzt.

7.4.2 SaaS aus Anbietersicht – Chancen und Risiken

7.4.2.1 Grundüberlegungen

Im letzten Abschnitt wurden mögliche Vor- und Nachteile der SaaS-Nutzung für Anwender dargestellt. Auch für Anbieter lassen sich Chancen und Risiken nennen. Ähnlich wie aus Anwendersicht lassen sich mehrere Chancen- und Risikokategorien für die Bereitstellung von SaaS-Anwendungen ableiten (Tab. 7.4).

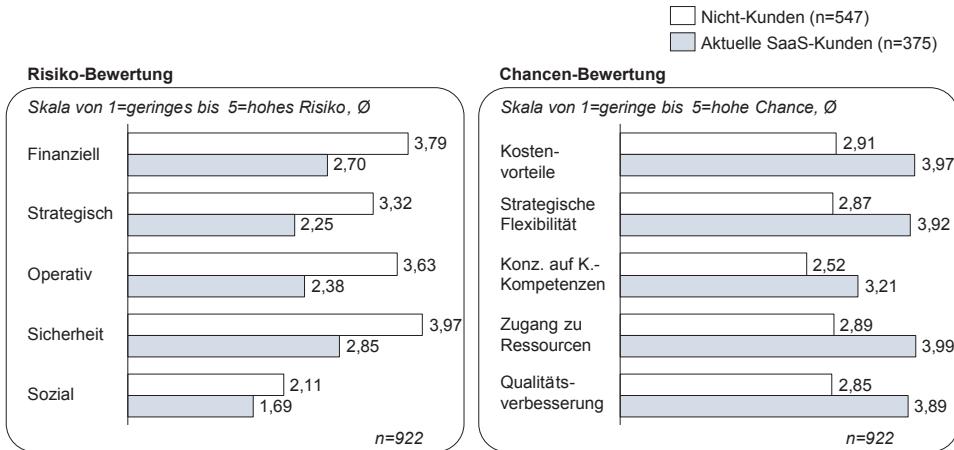


Abb. 7.10 Chancen-Risiken-Bewertung: SaaS-Kunden vs. Nicht-Kunden. (Benlian und Hess 2010)

Im Folgenden werden wir die in der Tabelle dargestellten Chancen und Risiken der SaaS-Bereitstellung näher betrachten.

Der Betreiber einer SaaS-Lösung kann, sofern er eine große Anzahl von Kunden besitzt, Skaleneffekte und somit Kostenvorteile realisieren. Dies betrifft die Bereiche IT-Infrastruktur, Hard- und Software sowie Personal. Falls es sich beim Anbieter von SaaS auch um den Hersteller der angebotenen Software handelt, sind zusätzliche Einsparpotenziale in der Entwicklung möglich. Da die Software lediglich auf seiner Plattform betrieben wird, entfallen die in der Regel aufwändigen Anpassungen hinsichtlich der Kompatibilität mit verschiedenen Betriebssystemen (Benlian und Hess 2010, S. 233).

Auch führt der Erhalt von monatlichen Gebühren im Gegensatz zu einmaligen Lizenz-erträgen zu einem vorhersehbaren und regelmäßigen Cash Flow (SIIA 2001, S. 7).

Für SaaS-Anbieter besteht darüber hinaus die Möglichkeit, neue Marktsegmente zu bedienen, die vorher für die Provider schwer zugänglich waren. So sind beispielsweise kleine und mittlere Unternehmen zunehmend an SaaS-Lösungen interessiert und können somit von Anbietern als neue Zielgruppe angesprochen werden (Stuckenbergs et al. 2011, S. 8). Eine weitere potentielle Kundengruppe wird in Unternehmen aus Entwicklungsländern gesehen. Auch entsteht durch den Verkauf von SaaS-Anwendungen die Möglichkeit, neue Vertriebskanäle zu nutzen (Marston et al. 2011, S. 178, 185). Während der Verkauf von on-premise Software häufig in persönlichen Verkaufssituationen geschieht, können SaaS-Lösungen direkt dem potenziellen Nutzer angeboten werden, ohne dass auf Kundenseite ein formalisierter Einkaufsprozess mit vielen Beteiligten stattfinden muss. Aus diesem Grund werden wachsende Erträge häufig bei bestehenden Kundenunternehmen erzielt, da sich die Nutzung von SaaS-Anwendungen über die Zeit im Unternehmen verbreitet (SIIA 2007, S. 13).

Weiterhin beinhalten SaaS-Anwendungen die Möglichkeit, die Kosten für Versions-management und Wartung zu reduzieren. Denn die Nutzung einer zentralen Applikation

Tab. 7.4 Chancen und Risiken von SaaS aus Anbietersicht

Chancen		Risiken	
Kategorie	Beschreibung	Kategorie	Beschreibung
Skaleneffekte	Chance, dass durch SaaS Applikationen Skaleneffekte entstehen und somit Kostenvorteile realisiert werden können	Kannibalisierung von bestehenden Software-Angeboten durch SaaS-Lösungen	Risiko, dass durch die Einführung von SaaS-Lösungen bestehende Software-Angebote kannibalisiert werden
Konstantere Cash Flows	Chance, dass durch den Erhalt monatlicher Gebühren für SaaS Applikationen konstantere Cash Flows entstehen	Management von komplexen Wertschöpfungsketten oder Netzwerken	Risiko, dass durch die Zusammenarbeit vieler SaaS-Anbieter in Wertschöpfungsketten oder Netzwerken sehr viele Anliegen berücksichtigt und Konflikte gelöst werden müssen
Erreichung neuer Marktsegmente	Chance, dass durch SaaS-Lösungen neue Zielgruppen angesprochen werden können, wie kleine und mittlere Unternehmen	Rückgang von initialen Umsätzen und hohe Vorabinvestitionen	Risiko, dass durch den starken Rückgang der initialen Umsätze und der hohen Vorabinvestitionen der kurzfristige Return on Investment reduziert wird
Reduzierung des Versionsmanagement und der Wartungskosten	Chance, dass durch die Nutzung einer zentralen Applikation die Verwaltung und Wartung verschiedener Versionen vereinfacht wird	Geringere Umsätze durch fehlende Professional Services	Risiko, dass durch das Angebot von SaaS-Lösungen weniger Professional Services in Anspruch genommen werden
Markteintrittsbarrieren	Chance, dass durch den frühen Markteintritt und die Erzielung von Economies of Scale Markteintrittsbarrieren für Wettbewerber aufgebaut werden können	Geringe Möglichkeit zur Individualisierung	Risiko, dass beispielsweise durch Probleme bei Änderungen an der zentralen Software alle Kunden gleichzeitig betroffen sind

Tab. 7.4 (Fortsetzung)

Chancen		Risiken	
Kategorie	Beschreibung	Kategorie	Beschreibung
Neue Arten von Anwendungen	Chance, dass durch die Unabhängigkeit von den IT-Ressourcen der Kunden leichter neue Lösungen entwickelt werden können, wie für beispielsweise mobile Endgeräte	Geringe Kundenabhängigkeit	Risiko, dass durch die Zahlung monatlicher Gebühren im Gegensatz zu einmaligen Lizenzkosten höhere Kundenfluktuation entsteht
SaaS als Marketingelement	Chance, dass SaaS Applikationen beispielsweise in die Präsentation von Lösungen beim Kunden eingebunden werden	Performanz- und Skalierungsrisiken	Risiko, dass das Nutzungsverhalten einzelner Kunden sich auf andere Kunden auswirkt und die Absicherung vor Auslastungsspitzen zu hohen Kosten führt
		Sicherheitsrisiken	Risiko, dass die Absicherung vor Sicherheitsrisiken hohe Kosten verursacht
		Starke Konkurrenz	Risiko, dass der hohe Wettbewerbsdruck eine „Winner-takes-it-all“ Situation schafft

vereinfacht die Verwaltung verschiedener Kundenversionen und -releases (Bhardwaj 2010, S. 41). Entwickler können sich somit auf die Optimierung und Verbesserung von aktuellen Versionen konzentrieren, anstatt Probleme mit überholten Versionen zu beheben (Stuckenberg et al. 2011, S. 10). Für alle Kunden ergibt sich dadurch, dass sie sofort von gelösten Problemen oder Fehlerbehebungen profitieren (Saeed und Jaffar-Ur-Rehman 2005, S. 304).

Für erfolgreiche Software-Anbieter besteht zudem die Chance, Markteintrittsbarrieren für Wettbewerber aufzubauen. So kann das frühe Erzielen von Marktanteilen zu Netzwerk- und Lock-in-Effekten führen (Cherry Tree & Co 2000, S. 7 und 16). Zu bewältigende Risiken (wie Sicherheits- und Zuverlässigkeitssprobleme) erhöhen u. U. ebenfalls die Markteintrittsbarrieren für potenzielle neue Wettbewerber (Garrison et al. 2012, S. 66).

Bei SaaS-Anwendungen ergibt sich das Potenzial, völlig neue Arten von Anwendungen einzuführen. So sind solche Applikationen unabhängig von den IT-Ressourcen auf der Kundenseite, wodurch die Erstellung von Lösungen für beispielsweise mobile Endgeräte enorm erleichtert wird. Darüber hinaus gibt es inzwischen immer mehr Plattformen und

Marktplätze, über die beispielsweise Apps vertrieben werden können, so dass ggf. zusätzliche Einnahmen entstehen (Stuckenbergs et al. 2011, S. 8).

Zuletzt entsteht durch eine gelungene SaaS-Umsetzung die Möglichkeit, die Marketingstrategie des anbietenden Unternehmens zu bereichern. So wird das Präsentieren von Lösungen und das Ausprobieren dieser durch potenzielle Kunden durch den einfachen Zugang – es wird nur ein Internetzugang und ein Web Browser benötigt – stark erleichtert (Stuckenbergs et al. 2011, S. 9).

Neben den Chancen existieren für SaaS-Anbieter natürlich auch Risiken. So besteht für Softwareunternehmen, die ihr Leistungsportfolio um SaaS-Anwendungen erweitern wollen, das Risiko, dass SaaS-Anwendungen andere Angebote kannibalisieren. Alle angebotenen Leistungen eines Unternehmens sollten demnach ausreichend voneinander abgegrenzt werden, damit Kunden, die zuvor on-premise Software bezogen haben, diese nicht durch eine SaaS-Lösung ersetzen. Vor allem beim Multi-Channel-Management der unterschiedlichen Leistungen eines Anbieters ist das Kannibalismusrisiko zu berücksichtigen (Stuckenbergs et al. 2011, S. 8).

SaaS-Anwendungen können grundsätzlich von mehreren Anbietern gemeinsam bereitgestellt werden. Viele kleine oder mittlere Anbieter sind beispielsweise nicht in der Lage, eine eigene Infrastruktur zur Verfügung zu stellen, sie sind auf spezialisierte Partner angewiesen. Das Management dieser komplexen Wertschöpfungsketten oder Netzwerke ist in der Regel schwierig (Stuckenbergs et al. 2011, S. 8). So müssen die Anforderungen aller beteiligten Parteien frühzeitig berücksichtigt werden und (potenzielle) Konflikte gelöst sowie die Zusammenarbeit sichergestellt werden (SIIA 2001, S. 7).

Der konstante Cash Flow beim Vertrieb von SaaS-Lösungen wurde bereits als Chance genannt, er kann aber auch nachteilig sein. So ist mit einem starken Rückgang des initialen Umsatzes bei der Bereitstellung von SaaS-Leistungen zu rechnen. Im Rahmen von bestehenden on-premise Softwareverträgen ist der Software-Anbieter in der Lage, signifikante initiale Umsätze für Lizizenzen, Implementierung und Customizing zu generieren. Die regelmäßigen monatlichen Gebühren für SaaS-Leistungen sind vergleichsweise geringer, ersetzen aber nun diese Umsätze. Dies ist insofern besonders problematisch, da SaaS-Anwendungen hohe Vorabinvestitionen (für Infrastruktur oder Entwicklung) benötigen (Anding 2010, S. 52). Im täglichen Geschäft sind dann die monatlichen Gebühren häufig die einzige Ertragsquelle. Im Gegensatz dazu werden bei on-premise Software typischerweise Professional Services angeboten, die über die Hälfte der Umsätze ausmachen. Bei SaaS-Modellen sind solche Services häufig nur noch in einem geringen Umfang notwendig, so dass sich diese Umsätze ebenfalls drastisch reduzieren (SIIA 2001, S. 22).

Bei SaaS-Anwendungen besteht nur eine begrenzte Möglichkeit zur Individualisierung. Weiterhin wird zum Teil in der geringen Kundenabhängigkeit ein Risiko gesehen werden. So müssen Kunden vor Nutzungsbeginn einer SaaS-Lösung keine hohen Investitionen durch beispielsweise einmalige Lizenzkosten tragen, stattdessen zahlen sie einen monatlichen Betrag. Dies kann dazu führen, dass Kunden schneller den Anbieter wechseln als bei klassischen Softwaremodellen (Anding 2010, S. 52).

SaaS-Anwendungen werden häufig mit Performanz- und Skalierungsrisiken in Verbindung gebracht. Da eine geteilte Infrastruktur vorliegt, kann das Nutzungsverhalten eines Kunden andere Kunden beeinflussen. SaaS-Anbieter können hierfür Maßnahmen ergreifen, um auch bei Auslastungsspitzen die Qualität der Anwendung aufrechtzuerhalten und die vorhandene Leistung zwischen den Kunden aufzuteilen (Durkee 2010, S. 66). Dieses Vorgehen führt jedoch zu höheren Kosten und somit tendenziell auch zu erhöhten Preisen. Weiterhin sind Netzwerke ein kritischer Faktor im Zusammenhang mit der Bereitstellung von Services, jedoch ist ihre Performanz und Zuverlässigkeit häufig nicht durch den SaaS-Anbieter beeinflussbar. Um sich gegen Zuverlässigkeitsrisiken absichern zu können, sind beispielsweise eine redundante Stromversorgung und eine redundante Netzanbindung notwendig, wodurch hohe Kosten für den Anbieter entstehen (Walsh 2003, S. 105).

Sicherheitsaspekte sind für die Kunden von SaaS-Anbietern ebenfalls sehr wichtig. Anbieter müssen diese Bedenken wahrnehmen und in Lösungen investieren, welche die Privacy und Security Anliegen ihrer Kunden berücksichtigen. Eine genaue Darstellung der Sicherheitsaspekte im Cloud Computing aus Anwender- und Anbietersicht findet sich in Abschn. 7.3.

Zuletzt wird in der Literatur der SaaS-Markt häufig als wettbewerbsintensiv beschrieben, wobei der Preis als wichtiges Entscheidungskriterium gesehen wird (Durkee 2010, S. 63 f.).

7.4.2.2 Empirische Untersuchung zu Chancen und Risiken der SaaS-Anbieter

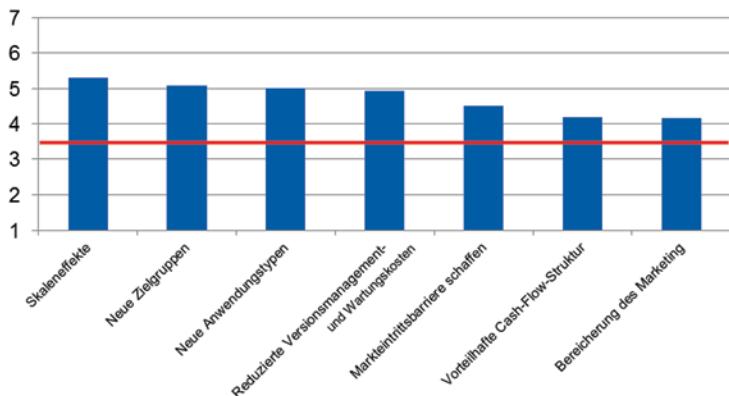
Vor dem im letzten Abschnitt dargestellten Hintergrund wollen wir nun untersuchen, welche Chancen und Risiken bei der Bereitstellung solcher Lösungen für SaaS-Anbieter bestehen. Im Rahmen einer Studie haben wir im Februar 2013 insgesamt 147 Software-Anbieter angeschrieben. 87 Anbieter haben an der Befragung teilgenommen und insgesamt 45 vollständig ausgefüllte Fragebögen konnten zur Analyse verwendet werden (Harnisch und Buxmann 2013, S. 6). Bei den Teilnehmern unserer Studie handelte es sich vor allem um Entscheidungsträger im Unternehmen, wie beispielsweise CIOs und CEOs. Fast zwei Drittel der Befragten hatte zudem mehr als zehn Jahre Berufserfahrung auf ihrem jeweiligen Gebiet. Auch zeigte sich, dass mehr als 75 % der befragten Unternehmen angaben, in Zukunft SaaS-Lösungen anbieten zu wollen. Weitere Charakteristika der befragten Unternehmen können der folgenden Tabelle entnommen werden (Tab. 7.5).

Der Fragebogen enthielt insgesamt fast 100 Fragen, wobei ein Großteil der Fragen als 7-stufige Likert Skala konzipiert wurde (1 = „Ich stimme überhaupt nicht zu“ bis 7 = „Ich stimme absolut zu“).

Die Befragung der Entscheidungsträger hat ergeben, dass in Bezug auf die Chancen bei der Bereitstellung von SaaS-Anwendungen vor allem Skaleneffekte als besonders wichtig eingestuft wurden. Hier wurde ein Mittelwert von 5,29 auf der Likert Skala ermittelt. Auch die Erreichung neuer Marktsegmente (Mittelwert=5,08) und die Möglichkeit, neue Anwendungstypen entwickeln zu können (Mittelwert=5), waren für die Teilnehmer der Studie wichtige Chancen. Die am niedrigsten bewertete Chance war die Möglichkeit, mit

Tab. 7.5 Stichprobenzusammensetzung (Harnisch und Buxmann 2013)

Unternehmensgröße		Positionen der Befragten	
Klein	80 %	CEO, CIO	64.4 %
Mittel	13.3 %	IS/IT Projektleiter	15.6 %
Groß	6.7 %	Andere	20 %
<i>Angebotene Applikationsarten</i>			<i>Berufserfahrung</i>
On-Premise	73.3 %	<5	15.6 %
SaaS	60 %	5–10	20 %
Planung neuer SaaS-Angebote	77.8 %	>10	64.4 %

**Abb. 7.11** Einschätzung der Chancen aus Anbietersicht. (Harnisch und Buxmann 2013)

SaaS-Lösungen das Marketing zu bereichern, jedoch zeigt ein Mittelwert von 4,16, dass diesem Vorteil auch Relevanz beigemessen wird. Abbildung 7.11 fasst die Ergebnisse der abgefragten Chancen zusammen.

Für die Einschätzung der Risiken gaben die Befragten der quantitativen Studie an, dass aus ihrer Sicht vor allem die hohen Vorabinvestitionen (Mittelwert=4,68) besonders kritisch sind. Ähnlich problematisch werden das Management von komplexen Wertschöpfungsketten oder Netzwerken (Mittelwert=4,62) und die Performanz- und Skalierungsprobleme (Mittelwert=4,58) gesehen. Als geringstes Risiko sahen die Entscheidungsträger die schlechtere Cash-Flow-Struktur (Mittelwert=3,84). Wie bereits bei der Bewertung der Chancen zeigte sich auch für die Risiken, dass die Mittelwerte nah beieinander liegen und somit kein Risiko als völlig irrelevant angesehen wird. Weitere Bewertungen der abgefragten Risiken finden sich in der folgenden Abb. 7.12. Das Sicherheitsrisiko war nicht Teil der Befragung, da wir hierzu bereits eine umfassende Studie durchgeführt haben, die in Abschn. 7.3.2 dargestellt wurde.

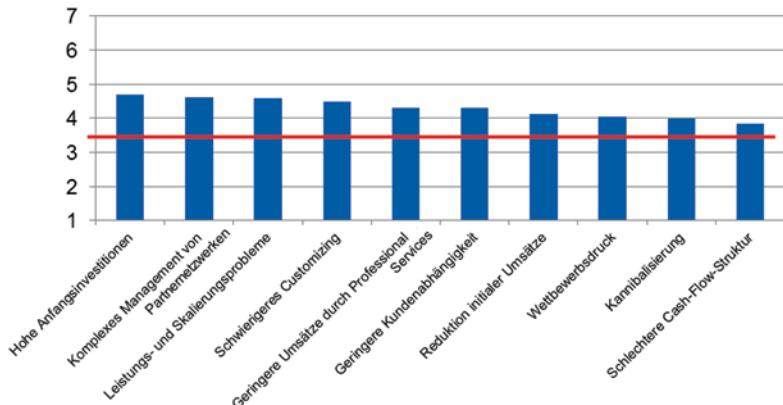


Abb. 7.12 Einschätzung der Risiken aus Anbietersicht. (Harnisch und Buxmann 2013)

7.4.3 Empirische Untersuchung der Preisstrategien und Geschäftsmodelle für SaaS-Anbieter

7.4.3.1 Datengrundlage und Methodik

Ziel unserer empirischen Studie war die Erhebung des Status quo derzeitiger SaaS-Angebote in Bezug auf ihre aktuell verwendeten Preismodelle (Lehmann et al. 2010). Im Rahmen einer *Inhaltsanalyse* wurden die Webseiten von SaaS-Anbietern hinsichtlich ihrer Preismodelle für SaaS-Produkte untersucht. Dabei wurden neben Preislisten und sonstigen Preisangaben auch Dokumente über die Allgemeinen Geschäftsbedingungen (AGB) ausgewertet.

Als Ausgangspunkt für die Suche nach SaaS-Anbietern diente das „SaaS-Forum“ (<http://www.saas-forum.net>). Dieses Informationsportal listet SaaS-Anbieter auf, die ihre Produkte für den deutschsprachigen Markt anbieten. Mehr als 250 SaaS-Anwendungen werden auf dieser Seite aufgeführt.

Zielgruppe der Studie waren Anbieter mit SaaS-Produkten für Geschäftskunden, sodass 114 SaaS-Produkte von insgesamt 80 Anbietern analysiert werden konnten. Die Repräsentativität der betrachteten SaaS-Anbieter für den deutschsprachigen Markt kann nicht garantiert werden. Da wir jedoch alle Anbieter aus der SaaS-Forumsliste für den B2B-Bereich berücksichtigt haben und das „SaaS-Forum“ nach Kenntnis der Autoren derzeit die umfassendste Auflistung von SaaS-Anbietern für den deutschsprachigen Markt ist, gehen wir davon aus, dass die untersuchte Stichprobe als für den Markt charakteristisch angesehen werden kann. Die Liste wird regelmäßig durch den Betreiber der Seite sowie durch eigenständige Registrierung seitens der SaaS-Anbieter aktualisiert und erweitert.

Hinsichtlich der Unternehmensgröße besteht die Stichprobe zu 53 % aus SaaS-Anbietern mit bis zu 50 Mitarbeitern. Rund 8 % der Anbieter haben zwischen 51 und 100 Mitarbeiter und 20 % verfügen über mehr als 100 Mitarbeiter. Zu den größten SaaS-Anbietern, die in der Studie untersucht wurden, zählen Microsoft, Salesforce.com und Google.

Nach der Produktart wurden die SaaS-Anwendungen in die Kategorien „Datenmanagement“ (z. B. Content-Management-Systeme), „Geschäftsanwendung“ (z. B. ERP, CRM, Finanzanwendungen, eCommerce- und eProcurementlösungen), „Kommunikation & Projektmanagement“ (z. B. Web Conferencing, Projektmanagementlösungen) sowie „Weitere“ (alle übrigen Produktarten) eingeteilt. Unter den betrachteten 114 SaaS-Produkten konnten 59 Produkte den Geschäftsanwendungen, 28 dem Bereich Kommunikation und Projektmanagement, 10 dem Datenmanagement und die verbleibenden 17 Produkte der Kategorie „Weitere“ zugeordnet werden.

Im Folgenden gehen wir auf ausgewählte Ergebnisse der Studie in Bezug auf die Preismodelle der Anbieter ein.

7.4.3.2 Ergebnisse

Die Analyse der jeweiligen Webseiten der Anbieter einschließlich der AGBs ergab ein gemischtes Bild über die Transparenz der Preismodelle.

Bei etwa einem Viertel der untersuchten SaaS-Produkte (26%) waren der Webseite des Anbieters keine Preisinformationen zu entnehmen. Unvollständige Informationen über das Preismodell der jeweiligen Lösung wurden bei ca. einem weiteren Viertel (26%) festgestellt. Beispielsweise gab es in diesen Fällen lediglich Aussagen zur Struktur des Zahlungsstroms. Bei knapp der Hälfte der betrachteten Anwendungen (48%) war das Preismodell einsehbar.

Die Ergebnisse sind im Weiteren nach der Systematik in Abschn. 3.3 gegliedert. Zunächst wird die Struktur des Zahlungsstroms betrachtet, bevor auf die Preisdifferenzierung und die Bemessungsgrundlage eingegangen wird. Die Untersuchung konzentriert sich im Folgenden auf die 84 SaaS-Produkte der Stichprobe, über die entweder teilweise oder vollständig Preisinformationen auf der Website des Anbieters vorliegen.

Struktur des Zahlungsstroms Die Struktur des Zahlungsstroms unterscheidet in Einmalzahlungen mit einem unbegrenzten Nutzungsrecht für den Kunden und regelmäßig wiederkehrende Zahlungen. Weiterhin ist auch eine hybride Form aus einmaligen und regelmäßigen Zahlungen denkbar. Die Analyse des Zahlungsstroms ergab folgendes Bild (Abb. 7.13):

Die Ergebnisse zeigen, dass kein Produkt mit einer ausschließlichen Einmalzahlung angeboten wird. Allerdings werden in einigen Fällen Einmalzahlungen, wie z. B. Einrichtungsgebühren, mit regelmäßigen Zahlungen verbunden.

In Bezug auf die Häufigkeit wird die monatliche Zahlung am häufigsten eingesetzt. Mit deutlichem Abstand folgt die jährliche Zahlung. Hinweise zur Mindestlaufzeit wurden bei 35 Produkten gefunden – mit 19 Nennungen ist eine zwölfmonatige Mindestdauer am häufigsten vertreten.

Preisdifferenzierung Im Folgenden betrachten wir die Preisdifferenzierung zweiten Grades. Sie ist zum einen in der Praxis weit verbreitet, zum anderen ist sie aufgrund der Selbstselektionsmöglichkeit der Kunden ein Teil des an den Kunden zu kommunizierenden Preismodells und daher auch häufig auf den Seiten der SaaS-Anbieter dargestellt.

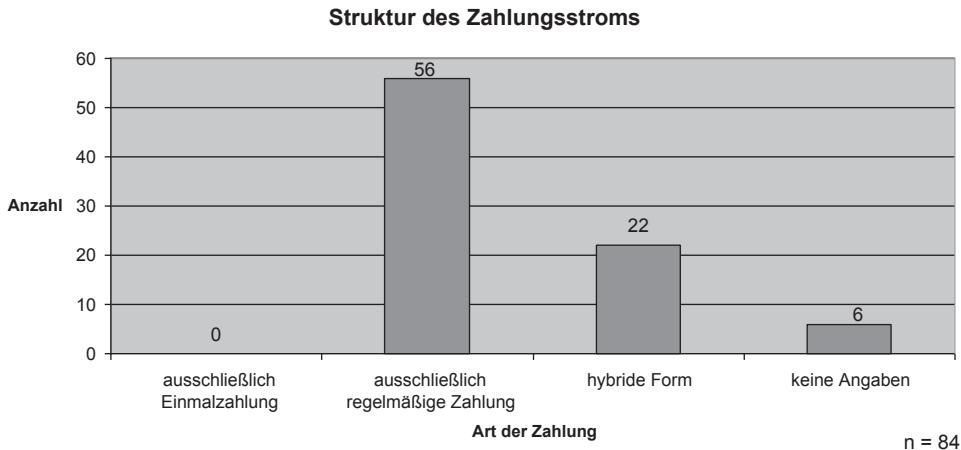


Abb. 7.13 Struktur des Zahlungsstroms in Bezug auf Einmal- und/oder regelmäßige Zahlung. (Lehmann et al. 2010)

Sowohl regionen- als auch personenbezogene Preisdifferenzierung wird von Anbietern nicht oder nur in sehr begrenztem Umfang über die Webseite kommuniziert, da hierbei keine Wahlmöglichkeit der Kunden vorliegt und die Kenntnis dieser Form der Preisdifferenzierung bei Kunden möglicherweise eher zu Verärgerungen führen würde.

Abbildung 7.14 zeigt die Ausprägungen der Preisdifferenzierung zweiten Grades in unserer Stichprobe. Durch die Möglichkeit der mehrdimensionalen Preisdifferenzierung (Skiera und Spänn 2002, S. 279), d. h. die gleichzeitige Differenzierung nach unterschiedlichen Dimensionen, sind Mehrfachzuordnungen zulässig.

Besonders ausgeprägt ist die mengenbezogene Preisdifferenzierung mit 63 von 75 betrachteten Produkten. So sinkt z. B. der Preis je Nutzer mit steigender Anzahl der Anwender. Ebenfalls verbreitet ist die leistungsbezogene Preisdifferenzierung, bei der Kunden beispielsweise unter Produktvarianten mit unterschiedlichem Funktionsumfang zu unterschiedlichen Preisen wählen können. Vergleichsweise selten konnten wir eine zeitbezogene Preisdifferenzierung feststellen, bei der die Preise z. B. abhängig von der Tages- oder Jahreszeit variieren.

Im Weiteren betrachten wir den Aspekt der leistungsbezogenen Preisdifferenzierung näher, der auch als Versioning bezeichnet wird (siehe hierzu Abschn. 3.3). Laut Informationen der Webseiten der SaaS-Anbieter gibt es bei knapp der Hälfte der betrachteten SaaS-Lösungen mehrere Versionen, unter denen die Kunden wählen können. Unter den 37 SaaS-Lösungen mit Angaben zur Anzahl der Produktversionen existieren 15 Anwendungen mit genau drei Versionen. Möglicher Hintergrund ist die Umsetzung des bereits in Abschn. 3.3.2.5 beschriebenen Prinzips der Extremeness Aversion.

Bemessungsgrundlage Bei den Bemessungsgrundlagen wird zwischen nutzungsabhängigen und nutzungsunabhängigen Einheiten unterschieden. Nutzungsabhängige Bemessungsgrundlagen orientieren sich an der tatsächlichen Nutzung der Software durch den

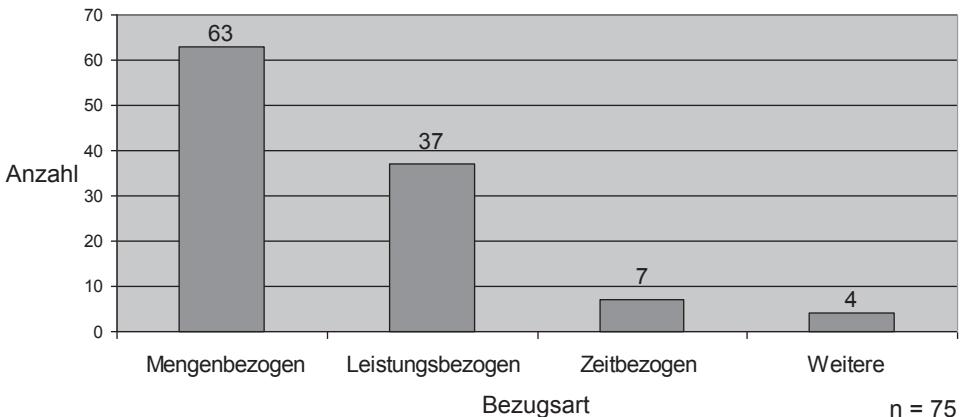


Abb. 7.14 Preisdifferenzierung zweiten Grades – Mehrfachzuordnungen zulässig. (Lehmann et al. 2010)

Kunden. Die Bemessungsgrundlage wird für jede Preiskomponente festgelegt. Daher bestimmt ein Anbieter zunächst die Anzahl der Preiskomponenten. Die Untersuchung der Preiskomponenten ergab, dass 23 % der SaaS-Lösungen lediglich eine Preiskomponente verwenden. Bei 35 % der Produkte enthält das Preismodell zwei Preiskomponenten. Weitere 32 % der Produkte verfügen über drei bis sechs Preiskomponenten.

Die Analyse aktuell verwendeter Bemessungsgrundlagen zeigt, dass die Mehrheit der Preismodelle auf nutzungsunabhängigen Einheiten basiert (siehe Abb. 7.15).

Unter den nutzungsunabhängigen Bemessungsgrundlagen dominieren mit 78 % die User-basierten Preismodelle. Unter ihnen sind 20 % an eine Mindestanzahl an Nutzern gebunden. Neben der klassischen Einheit User konnte in wenigen Fällen eine Bepreisung nach Concurrent User festgestellt werden. 13 % der Produkte mit nutzungsunabhängiger Bemessungsgrundlage basieren auf einer unternehmensweiten Lizenz.

Preismodelle mit ausschließlich nutzungsbasischen Bemessungsgrundlagen konnten mit 6 von 84 SaaS-Produkten relativ selten beobachtet werden. Häufiger ist hier eine Kombination aus nutzungsbasischen und nutzungsunabhängigen Bemessungsgrundlagen (29 Produkte). Bei den nutzungsbasischen Einheiten wird insbesondere auf Transaktionsbasis bepreist. Hierzu zählen u. a. Preise je Buchung, Gehaltsabrechnung, durchgeführte Prüfung, Bestellung oder Zollanmeldung. Ebenso wird auch der Speicherbedarf als nutzungsbasische Bemessungsgrundlage herangezogen. Dies trat allerdings in den von uns untersuchten Fällen nur als Größe auf, die nach dem Überschreiten einer bestimmten Inklusivmenge berechnet wird, nicht jedoch ab der ersten Einheit.

Weiterhin ist auffällig, dass nutzungsunabhängige Bemessungsgrundlagen fast ausschließlich obligatorisch für den Kunden sind, während nutzungsbasische Einheiten in vielen Fällen optional vom Kunden gewählt werden können (siehe Abb. 7.16).

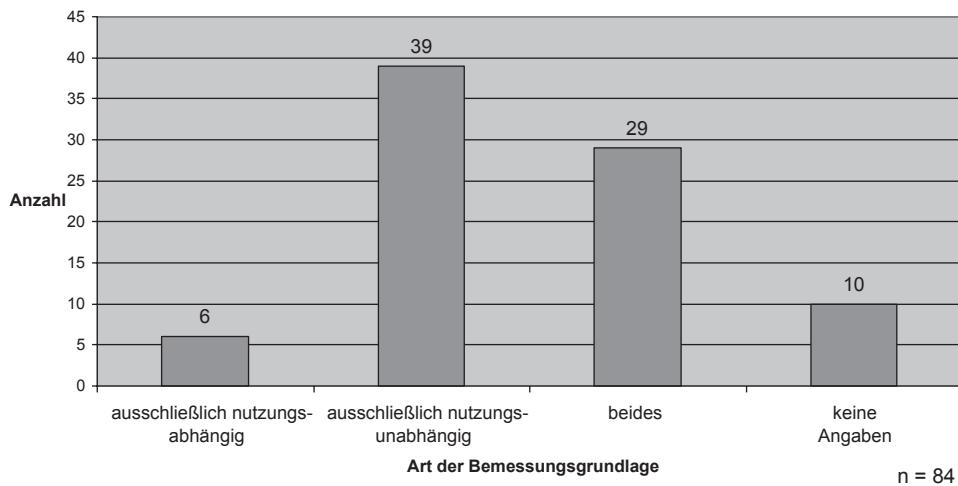


Abb. 7.15 Nutzungsabhängige und nutzungsunabhängige Bemessungsgrundlagen. (Lehmann et al. 2010)

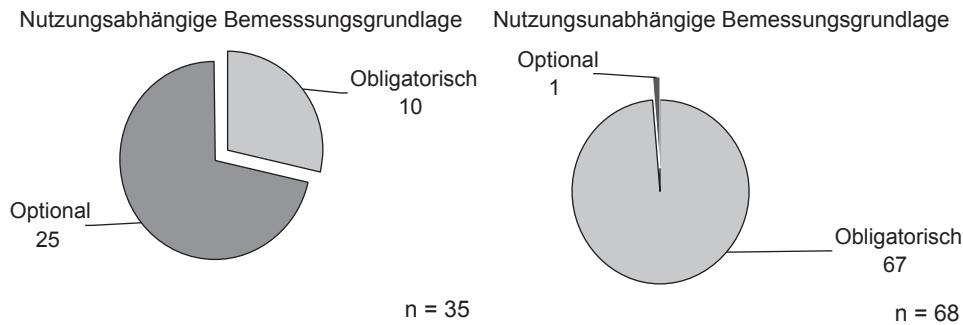


Abb. 7.16 Optionale und obligatorische Bemessungsgrundlagen. (Lehmann et al. 2010)

Insgesamt zeigt sich, dass der Einsatz von nutzungsabhängigen Bemessungsgrundlagen sich bisher nicht durchgesetzt hat, sondern weiterhin vorwiegend User-basierte Modelle verbreitet sind. Im folgenden Abschnitt gehen wir auf dieses Thema näher ein und untersuchen im Rahmen einer Fallstudie die Zahlungsbereitschaften für nutzungsabhängige und -unabhängige Bemessungsgrundlagen.

7.4.4 Fallstudie zum Vergleich nutzungsabhängiger und nutzungsunabhängiger Preismodelle

Im Folgenden wollen wir die Fragestellung einer nutzungsabhängigen bzw. nutzungsunabhängigen Bepreisung von SaaS-Lösungen anhand eines Beispiels näher erläutern.

Vor diesem Hintergrund haben wir mit einem Anbieter statistischer Anwendungssoftware für den B2B-Bereich eine Fallstudie durchgeführt. Dieser Softwareanbieter plant eine Applikation, die er derzeit als On-Premise-Variante bereitstellt, in Zukunft als SaaS-Lösung anzubieten. In diesem Zusammenhang stellt sich die Frage, welches Preismodell nach der Umstellung gewählt werden soll.

7.4.4.1 Datengrundlage und Methodik

Um eine belastbare Datenbasis aufzubauen, wurde eine telefonische Befragung unter den Kunden des Softwareanbieters durchgeführt. In diesem Zusammenhang wurde das *Price Sensitivity Meter (PSM)* von van Westendorp (van Westendorp 1976) eingesetzt, um die Zahlungsbereitschaften der Kunden für eine SaaS-Lösung im nutzungsabhängigen sowie nutzungsunabhängigen Fall zu vergleichen. Aufgrund des geringen Umfangs der Stichprobe und der Möglichkeit, Gründe für die Entscheidungen zu erfahren, haben wir uns gegen alternative Verfahren, wie beispielsweise die Conjoint-Analyse, entschieden.

Bei der van Westendorp-Methode handelt es sich um eine Form der direkten Kundenbefragung, bei der den Befragten vier Fragen zu einem vorab definierten Produkt gestellt werden. Diese Fragen betreffen die Einschätzung des Probanden bezüglich der vier Preispunkte: „günstig“, „teuer“, „zu teuer“ sowie „zu billig“, wenn er nach einer Preisvorstellung für das Produkt gefragt wird. Die Antworten der einzelnen Teilnehmer werden anschließend aggregiert und als Kurven in einem Diagramm visualisiert. Durch die Schnittpunkte der vier resultierenden Kurven wird eine Spanne beschrieben, innerhalb derer der Preis für das angebotene Produkt liegen sollte.¹ Damit zielt das PSM nicht darauf ab, eine konkrete Preis-Absatz-Funktion zu ermitteln. Vielmehr liegt die Zielsetzung in der Ermittlung einer „akzeptablen Preisspanne“ (Lock 1998, S. 507) für ein innovatives Produkt, für das noch keine Preisvorstellung existiert.

Die Stichprobe bestand aus 28 Kunden dieses Softwareanbieters, die das vorgestellte SaaS-Produkt bereits als On-Premise-Variante nutzen. Daher kennen die Kunden sowohl die Funktionalität der Anwendung als auch die im aktuellen Lizenzierungsmodell anfallenden Kosten.

Hinsichtlich der Unternehmensgröße besteht die Stichprobe zu rund 38 % aus kleinen und mittleren Unternehmen mit einem Jahresumsatz von weniger als 500 Mio. € und zu 62 % aus großen Unternehmen mit mehr als 500 Mio. € Jahresumsatz. Die Unternehmen entstammen vorwiegend der Automobil- und Zuliefererindustrie im deutschsprachigen Raum. Im Folgenden gehen wir auf die Ergebnisse der durchgeführten Befragung ein.

¹ Für die Bedeutung der einzelnen Schnittpunkte siehe u. a. Simon 2008, S. 175.

7.4.4.2 Ergebnisse

Die van Westendorp-Methode wurde im Rahmen der Befragung zweimal eingesetzt. Im ersten Fall wurde für das Preismodell der beschriebenen SaaS-Anwendung die nutzungsunabhängige Bemessungsgrundlage des Concurrent User gewählt. Mit zwölf von 28 Befragten machten rund 43 % der Probanden Angaben zu allen vier abgefragten Preispunkten. Die Ergebnisse dieser Untersuchung werden in Abb. 7.17 illustriert. Das PSM liefert – wie oben beschrieben – eine Preisspanne, innerhalb derer der Preis für die Anwendung festgelegt werden sollte. Die Empfehlung liegt zwischen dem „Point of Marginal Cheapness“ (PMC) und dem „Point of Marginal Expensiveness“ (PME), also zwischen 5,46 und 24,10 € pro Monat und Concurrent User.

Die parallel durchgeführte Erhebung der Zahlungsbereitschaften im nutzungsabhängigen Fall lieferte eine deutlich geringere Antwortquote. Da die Software u. a. die Anfertigung von statistischen Berichten ermöglicht, wurde als nutzungsabhängige Bemessungsgrundlage die Transaktion „angefertigter Bericht“ gewählt. Lediglich 14 % (vier der 28 Teilnehmer) nannten konkrete Werte für alle vier abgefragten Preispunkte.

Dennoch lassen sich aus den Ergebnissen erste Erkenntnisse hinsichtlich der geringen Bedeutung nutzungsabhängiger Preismodelle ableiten. Ergänzend zu den konkreten Preispunkten konnten durch die telefonische Befragung Hintergründe für die fehlende Einschätzungsfähigkeit der Teilnehmer in die Auswertung einbezogen werden: Der Mehrheit der Befragten ist die tatsächliche Nutzungsintensität ihrer eingesetzten Software nicht bekannt. Da die Intensität jedoch im Falle einer nutzungsabhängigen Bemessungsgrundlage die Kosten bestimmt, ist es diesen Kunden nicht möglich, eine Einschätzung der Zahlungsbereitschaft anzugeben.

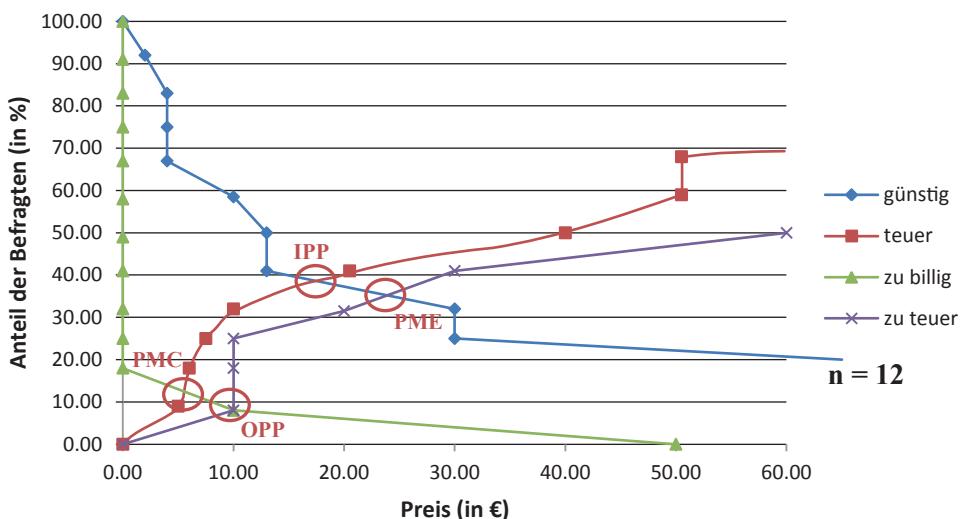


Abb. 7.17 Ergebnisse des PSM im nutzungsunabhängigen Fall. (Lehmann et al. 2010)

Als weiteres Problem stellte sich die Auswahl der konkreten Bemessungsgrundlage – wie hier in Form des Berichts – heraus. Aufgrund der Vielfältigkeit der Software und dem daraus resultierenden heterogenen Nutzungsverhalten der Anwender, bestanden unterschiedliche Auffassungen über eine geeignete nutzungsabhängige Bemessungsgrundlage.

Auch unter den Antwortenden wurden unerwartete Ergebnisse erzielt. So bestand zwischen den Einschätzungen zweier Befragter zum „günstig“-Preispunkt ein Unterschied von einem Faktor größer 1000, was auf eine sehr heterogene Zahlungsbereitschaft der Kunden hinweist. Aus Anbieterperspektive ist diese ausgeprägte Heterogenität der Zahlungsbereitschaften besonders ungünstig: Um einerseits viele Kunden zu erreichen und andererseits deren Zahlungsbereitschaft bestmöglich abzuschöpfen, ist eine homogene Struktur der Zahlungsbereitschaften mit einer geringen Varianz erforderlich.

Generell bietet sich die Preisdifferenzierung zum Abschöpfen unterschiedlicher Zahlungsbereitschaften an. Dies dürfte jedoch in diesen Dimensionen schwierig umzusetzen sein. Weiterhin kommt hinzu, dass die Preisgestaltung für SaaS-Produkte vermutlich aufgrund des Vertriebs über Internet tendenziell transparenter sein wird als beispielsweise bei On-Premise-Software. Dies erschwert insbesondere eine Preisdifferenzierung dritten Grades.

Im Ergebnis kann die häufig verbreitete Aussage, dass sich SaaS hervorragend für eine nutzungsabhängige Bepreisung eignet, durch unsere Untersuchung nicht bestätigt werden. Bei den Bemessungsgrundlagen kommen überwiegend nutzungsunabhängige Einheiten zum Einsatz. Nutzungsabhängige Bemessungsgrundlagen hingegen, wie Preise je durchgeführte Transaktion, sind für Kunden meist optional.

Allerdings ist zu beachten, dass die Erhebung der Zahlungsbereitschaften unter potentiellen SaaS-Kunden eines spezifischen Softwareprodukts durchgeführt wurde. Von Interesse sind vergleichbare Erhebungen unter SaaS-Produkten aus anderen Bereichen, wie beispielsweise einer ERP- oder CRM-Software sowie die allgemeine Analyse des Zusammenhangs zwischen der Verteilung von Zahlungsbereitschaften und der Ausgestaltung des Preismodells.

Literatur

- Ackermann T, Widjaja, T, Benlian, A, Buxmann, P (2012) Perceived IT security risks of cloud computing: conceptualization and scale development. In: Proceedings of the 33rd International Conference on Information Systems, Orlando, USA
- Altmann J, Ion M, Bany M, Ashraf A (2007) Taxonomy of grid business models. In: Veit DJ, Altmann J (Hrsg) Grid economics and business models, 4th International Workshop, GECON 2007, Berlin, Heidelberg, S 29–43
- Anding M (2010) SaaS: a love-hate relationship for enterprise software vendors. In: Benlian A (Hrsg) Software-as-a-Service, 1st Aufl. Gabler, Wiesbaden, pp 43–56
- Benlian A, Hess T (2009) Welche Treiber lassen SaaS auch in Großunternehmen zum Erfolg werden? Eine empirische Analyse der SaaS-Adoption auf Basis der Transaktionskostentheorie. In: Proceedings der 9. internationalen Tagung Wirtschaftsinformatik, Wien

- Benlian A, Hess T (2010) Chancen und Risiken des Einsatzes von SaaS – Die Sicht der Anwender. In: Benlian A, Hess T, Buxmann P (Hrsg) Software-as-a-Service. Gabler, Wiesbaden, S 173–187
- Bhardwaj S (2010) An approach for investigating perspective of cloud Software-as-a-Service (SaaS). *Int J Comput Appl* 10:40–43
- Buxmann P, Hofmann A (2011) Cloud computing. *Das Wirtschaftsstudium* 40:824–828
- Buxmann P, Hess T, Lehmann S (2008b) Software as a service. *Wirtschaftsinformatik* 50:500–503
- Cherry Tree & Co (2000) Framing the IT services industry: „2nd Generation Asps“. Cherry Tree & Co, Edina
- Durkee D (2010) Why cloud computing will never be free. *Commun ACM* 53:62–69
- Earl MJ (1996) The risks of outsourcing IT. *Sloan Manage Rev* 37:26–32
- Garrison G, Kim S, Wakefield RL (2012) Success factors for deploying cloud computing. *Commun ACM* 55:62–69
- Günther O, Tamm G, Hansen L, Meseg T (2001) Application service providers: Angebot, Nachfrage und langfristige Perspektiven. *Wirtschaftsinformatik* 43:555–568
- Harnisch S, Buxmann P (2013) The providers' perspective on Software-as-a-Service: what motivates software vendors to offer SaaS? Initial empirical insights. Arbeitspapier, TU Darmstadt
- Kalenda F (2013) Analyst: Amazon Web Services macht 2013 etwa 3,8 Milliarden Dollar Umsatz. <http://www.zdnet.de/88138507/analyst-amazon-web-services-macht-2013-38-mrd-dollar-umsatz/>
- Kern T, Willcocks LP, Lacity MC (2002) Application service provision: risk assessment and mitigation. *MIS Q Exec* 1:113–126
- Kurzlechner W (2013) SAP schwächtelt bei CRM. <http://www.cio.de/knowledgecenter/crm/2914935>
- Lehmann S, Draisbach T, Koll C, Buxmann P, Diefenbach H (2010) Preisgestaltung für Software-as-a-Service. Ergebnisse einer empirischen Analyse mit Fokus auf nutzungsabhängige Preismodelle. In: Proceedings zur Teilkonferenz „Software-Industrie“ der Multikonferenz Wirtschaftsinformatik (MKWI) 2010, pp 505–516
- Lock D (1998) The Gower handbook of management. Gower Publishing, Hampshire
- Loske A, Widjaja T, Buxmann P (2013) Cloud computing providers' unrealistic optimism regarding IT security risks: a threat to users? In: International Conference on Information Systems (ICIS), Milan, Italy
- Loske A, Widjaja T, Benlian A, Buxmann P (2014) Perceived IT security risks in cloud adoption: the role of Perceptual Incongruence between Users and Providers. In: Proceedings of the 22nd European Conference on Information Systems, Tel Aviv, Israel
- Marston S, Li Z, Bandyopadhyay S, Zhang J, Ghalsasi A (2011) Cloud computing – the business perspective. *Decis Support Syst* 51:176–189
- National Institute of Standards and Technology (2009) NIST definition of cloud computing. <http://csrc.nist.gov/groups/SNS/cloud-computing/>
- Saeed M, Jaffar-Ur-Rehman M (2005) Enhancement of software engineering by shifting from software product to software service. In: First International Conference on Information and Communication Technologies, S 302–308
- SIIA (2001) Software as a service: strategic backgrounder. Software & Information Industry Association (SIIA)
- SIIA (2007) Channels for the new SaaS industry. SIIA
- Skiera B, Spann M (2002) Preisdifferenzierung im Internet. In: Schögel M, Tomczak T, Belz C (Hrsg) Roadmap to E-Business – Wie Unternehmen das Internet erfolgreich nutzen. St. Gallen, Texis, S 270–284
- Statista (2014c) Prognostizierter Umsatz im B2B- und B2C-Segment von Cloud Computing. <http://de.statista.com/statistik/daten/studie/180537/umfrage/prognostizierter-umsatz-im-b2b-und-b2c-segment-von-cloud-computing/>

- Stuckenbergs, Fiebt E, Loser T (2011) The impact of Software-as-a-Service on business models of leading software vendors: experiences from three exploratory case studies. In: PACIS 2011 Proceedings, S 1–16
- Vaquero LM, Rodero-Merino L, Caceres J, Lindner M (2009) A break in the clouds: towards a cloud definition. SIGCOMM Comput Commun Rev ACM 39:50–55
- Wallraff B, Weber M, Dufft N (2013) Cloud Monitor 2013. http://www.bitkom.org/files/documents/BITKOM_KPMG_PAC_Studie_Cloud_Monitor_2013.pdf
- Walsh K (2003) Analyzing the application ASP concept: technologies, economies, and strategies. Commun ACM 46:103–107
- Welt (2013) NSA Skandal kostet USA bis zu 35 Milliarden Dollar. <http://www.welt.de/wirtschaft/article122307814/NSA-Skandal-kostet-USA-bis-zu-35-Milliarden-Dollar.html>
- van Westendorp PH (1976) NSS-Price Sensitivity Meter: a new approach to study consumer perception of prices. Venice ESOMAR Congress, Amsterdam, S 139–167

8.1 Überblick

Softwareanbieter im engeren Sinne erstellen Software mit dem Ziel, Lizenz- und gegebenenfalls auch Dienstleistungserlöse zu generieren. Die Motivationslage bei Open-Source-Projekten sieht demgegenüber meistens anders aus: Eine internationale Gemeinde von Softwareentwicklern schließt sich zusammen, um gemeinsam ein Problem zu lösen und ihr Wissen zu teilen. Dabei investieren viele Entwickler ihre Zeit, ohne dass sie – zumindest gilt dies in den meisten Fällen – dafür bezahlt werden. Das bedeutet jedoch nicht, dass Open Source Software (OSS) ökonomisch nicht relevant ist. Im Gegenteil: OSS stellt sowohl die Softwareindustrie als auch die Anwender vor grundlegende Fragen, denen wir in diesem Kapitel nachgehen möchten.

Wir starten mit einer kurzen Einführung in die Charakteristika von OSS und die Entstehung der Open-Source-Bewegung. Es folgt eine Untersuchung des Entwicklungsprozesses im Rahmen von Open-Source-Projekten, der sich grundsätzlich vom Entwicklungsprozess in einem traditionellen Softwareunternehmen unterscheidet. Insbesondere wollen wir hier auch auf die Motivation der Entwickler zur Teilnahme an Open-Source-Projekten eingehen. Darauf aufbauend wird die Einführung von OSS aus Anwenderperspektive betrachtet. Im nächsten Schritt geben wir einen Überblick über Strategien kommerzieller Softwareanbieter „gegen und mit“ Open Source. Wir schließen mit Überlegungen und ersten empirischen Ergebnissen zum Einsatz von Open Source Software für betriebswirtschaftliche Anwendungen.

8.2 Charakteristika von Open Source Software

Frei verfügbare Software gibt es schon recht lange. Zum einen haben private Nutzer häufig auf Freeware zurückgegriffen, so z. B. bei bestimmten Datenbanksystemen oder Spielen. Dabei haben sich die Distributionswege über die Jahre verändert: In den ersten Jahren des Personal Computers wurde Freeware über Disketten ausgetauscht, dann über CDs und inzwischen fast ausschließlich über das Internet. Zum anderen war es seit jeher üblich, dass Programmierer ihre Quelltexte und Programme untereinander tauschten, um voneinander zu lernen und sich gegenseitig zu unterstützen.

Als in den siebziger Jahren einige Firmen dazu übergingen, nur die kompilierte Software zu vertreiben und den Quelltext unter Verschluss zu halten, entwickelte sich Widerstand gegen dieses Vorgehen. Als Vorreiter ist hier Richard Stallman zu nennen. Er begann seine akademische Laufbahn 1971 am Labor für Künstliche Intelligenz beim MIT. Folgendes Statement von Stallman gibt einen Eindruck von der zu dieser Zeit dort herrschenden Kultur (Grassmuck 2004, S. 219):

Ich hatte in den 70er Jahren das Glück, Teil einer Gemeinschaft zu sein, in der die Menschen Software miteinander teilten. Wir entwickelten Software und wann immer jemand ein interessantes Programm geschrieben hatte, wurde es weitergegeben. So arbeitete einer nach dem anderen, um die Software zu verbessern und weiterzuentwickeln. Man konnte in dieser Gemeinschaft immer eine zumindest passive Mitarbeit eines jeden erwarten. Sie mochten zwar nicht bereit sein, ihre Arbeit zu unterbrechen, um stundenlang etwas für dich zu tun, aber das, was sie bereits erledigt hatten, konntest du gerne benutzen.

Damit wird auch klar, dass freie Software keineswegs erst in den neunziger Jahren als Phänomen in Erscheinung getreten ist. Vielmehr wurde das Teilen von Softwarecode und von Wissen in vielen Fällen bereits früher als eine Selbstverständlichkeit betrachtet.

Die Unzufriedenheit mit der Funktionalität eines Druckertreibers war schließlich ein wichtiger Ausgangspunkt der Entstehung der Open-Source-Bewegung (Grassmuck 2004, S. 222) – auch wenn der Begriff Open Source damals noch nicht verwendet wurde: So gab es bei den damals am MIT verwendeten Xerox-Netzwerkdruckern keine Funktion, die den Druckerstatus direkt am Arbeitsplatz anzeigen konnte. Stallman wollte eine Funktion schreiben, die diese Statusabfrage ermöglicht, und in den Quellcode des Druckertreibers integrieren. Der zuständige Mitarbeiter bei Xerox verweigerte jedoch die Herausgabe des Quellcodes, da er sich zur Nichtweitergabe verpflichtet hatte.

Vor diesem Hintergrund entwickelte er zum einen den Treiber selbst. Zum anderen gründete Stallman das Projekt GNU. GNU ist die rekursive Abkürzung für „GNU is not Unix“. Um die kommerzielle Verwertung seiner Arbeit zu verhindern, kündigte er seinen Job am MIT. Zur Sicherung seines Lebensunterhalts und zur Fortführung des GNU-Projekts gründete Stallman die Free Software Foundation, die für die Distribution von GNU-Software auf Trägermedien mit Handbüchern – nicht aber für die Software selbst – Gebühren erhob, Spenden einsammelte und Entwickler einstellte. Anfang der neunziger

Jahre wurden die Komponenten des GNU-Projekts mit dem Linux-Kernel zusammengefügt, und es entstand ein komplettes System namens GNU/Linux.

Im Rahmen des GNU-Projekts entstanden aber nicht nur freie Systemkomponenten, sondern es wurde zudem mit der GNU General Public License (GPL) eine spezielle Softwarelizenz entwickelt, welche die Freie-Software- sowie die Open-Source-Bewegung maßgeblich beeinflusst hat. Die GPL gewährt hierbei den Nutzern etwa den freien Zugang zum Quellcode, das Recht, die Software zu kopieren und weiterzugeben, die Freiheit, das Programm zu ändern sowie die Möglichkeit, das veränderte Programm – allerdings unter denselben Bedingungen – zu verbreiten.

Ökonomisch ausgedrückt verhindert die letztgenannte Lizenzbedingung eine zukünftige Änderung der Eigentumsrechte (Property Rights) an einer Software. Anstelle eines Verzichts auf die urheberrechtlichen Ansprüche der Softwareentwickler, wie dies etwa bei Public Domain Software der Fall ist, wird hierbei das so genannte Copyleft-Prinzip genutzt, um die Freiheit der Software auf Dauer zu gewährleisten. Dieses Prinzip besagt, dass auch eine weiterentwickelte Version der Software unter die gleiche Lizenz gestellt werden muss.

Neben der GPL gibt es eine Vielzahl weiterer Open-Source-Lizenzen, wie die weniger restriktive Library/Lesser General Public License (LGPL, zunächst speziell für Bibliotheken entwickelt), die Berkeley Software Distribution Style License (BSD-Style-License) und die Mozilla Public License (MPL). In Tab. 8.1 sind ausgewählte Eigenschaften dieser Lizenzen im Überblick dargestellt.

Der Begriff Open Source entstand erst im Jahr 1998 mit der Gründung der Open Source Initiative (OSI). Bis dahin hatte sich die Bezeichnung „Free Software“ etabliert. Es war allerdings mehr als eine rein sprachliche Änderung. Neben Eric S. Raymond, der insbesondere durch seinen Aufsatz „The Cathedral and the Bazaar“ (Raymond 1999) bekannt wurde, in dem er ein zentral gesteuertes Softwareprojekt mit dem Bau einer Kathedrale und die dezentrale Organisation eines Projekts der Linux-Gemeinde mit einem Bazar verglich, waren auch Vertreter von Softwareunternehmen an einer Umorientierung der Freien-Software-Bewegung interessiert. Ein Anlass zur Gründung war die Ankündigung von Netscape, den Quellcode seines Browsers zu veröffentlichen.

Im Kern ging es den Gründern der OSI darum, der Freien-Software-Bewegung eine neue Richtung zu geben. So bestand ein wesentliches Ziel darin, die Zusammenarbeit mit

Tab. 8.1 Ausgewählte Eigenschaften spezieller Open-Source-Lizenzen. (in Anlehnung an Perens 1999, S. 186)

Lizenztyp	GPL	LGPL	MPL	BSD-License
Kann mit proprietärer Software verbunden und ohne OS-Software-Lizenz redistribuiert werden	Nein	Ja	Ja	Ja
Modifikationen am OS-lizenzierten Quellcode können im Distributionsfall proprietär bleiben	Nein	Nein	Nein	Ja

Softwareunternehmen zu erleichtern. Um „das Konzept der Freien Software an Leute zu verkaufen, die Krawatten tragen“ (Perens 1999, S. 173), wurde „Free Software“ in Open Source umbenannt. Volker Grassmuck kommentiert, dass einige Entwickler wohl befürchteten, dass das Wort „free“ zu Missverständnissen führen und als kommunistisches „four letter word“ interpretiert werden könnte (Grassmuck 2004, S. 230).

Auf Basis dieser Überlegungen fand jedoch nicht nur eine Begriffsänderung statt, sondern die Gruppe formulierte eine Definition von „Open Source“, die auf Bruce Perens, den ehemaligen Projektleiter von Debian GNU/Linux, zurückgeht. Die Definition umfasst mehrere Kriterien, die erfüllt sein müssen, damit eine Software als Open Source bezeichnet werden kann. Diese sind in dem folgenden Kasten abgebildet.

Open-Source-Definition der OSI

Quelloffen („Open Source“) bedeutet nicht nur freien Zugang zum Quellcode. Bei quelloffener Software müssen die Lizenzbestimmungen in Bezug auf die Weitergabe der Software folgenden Kriterien entsprechen:

1. Freie Weitergabe

Die Lizenz darf niemanden in seinem Recht einschränken, die Software als Teil eines Software-Paketes, das Programme unterschiedlichen Ursprungs enthält, zu verschenken oder zu verkaufen. Die Lizenz darf für den Fall eines solchen Verkaufs keine Lizenz- oder sonstigen Gebühren festsetzen.

2. Quellcode

Das Programm muss den Quellcode beinhalten. Die Weitergabe muss sowohl für den Quellcode als auch für die kompilierte Form zulässig sein. Wenn das Programm in irgendeiner Form ohne Quellcode weitergegeben wird, so muss es eine allgemein bekannte Möglichkeit geben, den Quellcode zum Selbstkostenpreis zu bekommen, vorzugsweise als gebührenfreien Download aus dem Internet. Der Quellcode soll die Form eines Programms sein, die ein Programmierer vorzugsweise bearbeitet. Absichtlich unverständlich geschriebener Quellcode ist daher nicht zulässig. Zwischenformen des Codes, so wie sie etwa ein Präprozessor oder ein Konverter („Translator“) erzeugt, sind unzulässig.

3. Abgeleitete Software

Die Lizenz muss Veränderungen und Derivate zulassen. Außerdem muss sie es zulassen, dass die solcherart entstandenen Programme unter denselben Lizenzbestimmungen weitervertrieben werden können wie die Ausgangssoftware.

4. Unversehrtheit des Quellcodes des Autors

Die Lizenz darf die Möglichkeit, den Quellcode in veränderter Form weiterzugeben, nur dann einschränken, wenn sie vorsieht, dass zusammen mit dem Quellcode so genannte „Patch files“ weitergegeben werden dürfen, die den Programmcode bei der Kompilierung verändern. Die Lizenz muss die Weitergabe von Software, die aus verändertem Quellcode entstanden ist, ausdrücklich erlauben. Die Lizenz kann

verlangen, dass die abgeleiteten Programme einen anderen Namen oder eine andere Versionsnummer als die Ausgangssoftware tragen.

5. Keine Diskriminierung von Personen oder Gruppen

Die Lizenz darf niemanden benachteiligen.

6. Keine Einschränkungen bezüglich des Einsatzfeldes

Die Lizenz darf niemanden daran hindern, das Programm in einem bestimmten Bereich einzusetzen. Beispielsweise darf sie den Einsatz des Programms in einem Geschäft oder in der Genforschung nicht ausschließen.

7. Weitergabe der Lizenz

Die Rechte an einem Programm müssen auf alle Personen übergehen, die diese Software erhalten, ohne dass für diese die Notwendigkeit bestünde, eine eigene, zusätzliche Lizenz zu erwerben.

8. Die Lizenz darf nicht auf ein bestimmtes Produktpaket beschränkt sein

Die Rechte an dem Programm dürfen nicht davon abhängig sein, ob das Programm Teil eines bestimmten Software-Paketes ist. Wenn das Programm aus dem Paket herausgenommen und im Rahmen der zu diesem Programm gehörenden Lizenz benutzt oder weitergegeben wird, so sollen alle Personen, die dieses Programm dann erhalten, alle Rechte daran haben, die auch in Verbindung mit dem ursprünglichen Software-Paket gewährt wurden.

9. Die Lizenz darf die Weitergabe zusammen mit anderer Software nicht einschränken

Die Lizenz darf keine Einschränkungen enthalten bezüglich anderer Software, die zusammen mit der lizenzierten Software weitergegeben wird. So darf die Lizenz z. B. nicht verlangen, dass alle anderen Programme, die auf dem gleichen Medium weitergegeben werden, auch quelloffen sein müssen.

© 1999–2014 Frank Ronneburg – Dieser Inhalt ist unter einem Creative Commons Namensnennung – Nicht-kommerziell – Keine Bearbeitung Lizenzvertrag lizenziert (<http://creativecommons.org/licenses/by-nc-nd/3.0/de/>)

Bei dieser Definition handelt es sich also um keine Lizenz, sondern um einen Standard, an dem Lizenzen gemessen werden. Die OSI übernimmt damit faktisch die Rolle einer Zertifizierungsinstanz. Mittlerweile wurden mehr als 60 Lizenzen zertifiziert, darunter u. a. die GNU GPL, GNU LGPL, Mozilla Public License sowie die New BSD License. Eine aktuelle Liste findet sich unter <http://www.opensource.org/licenses>.

Einige dieser Lizenzen machen es Softwareanbietern leichter, OSS zu privatisieren bzw. zu kommerzialisieren. Aus diesem Grund ist die Open-Source-Definition der Open Source Initiative auch Gegenstand vieler kontroverser Debatten. Beispielsweise schließt Punkt 1 der oben dargestellten Kriterienliste nicht aus, dass der Open Source Code im Rahmen eines kommerziell vertriebenen Softwarepaketes genutzt wird. So erlaubte es beispielswei-

se die BSD-Lizenz Microsoft, Open Source Code in Windows zu integrieren. Hätte dieser Code unter der GPL gestanden, hätte ihn Microsoft nicht verwenden dürfen oder Windows wäre heute freie Software (Grassmuck 2004, S. 299).

Zudem schließt die GPL, wie oben dargestellt, eine Privatisierung von Codeänderungen aus, während dies bei Lizenzen wie Apache oder BSD durchaus erlaubt ist (Grassmuck 2004, S. 301). Das bedeutet, dass eine modifizierte OSS sogar ohne Offenlegung des Quellcodes verkauft werden darf.

Es ist unmittelbar einsichtig, dass somit viele Softwareunternehmen von der neuen durch die OSI vorgegebenen Richtung profitieren können. Zudem kann die GPL durchaus ein Hindernis (auch für nicht-kommerzielle) Softwareprojekte darstellen, wenn etwa Open Source Code mit kommerziellen Produkten integriert werden soll. So wurde die bereits zuvor angesprochene LGPL mit dem Ziel entwickelt, das Teilen von Code attraktiver zu machen, indem sich Bibliotheken einfacher einbinden lassen. Würde man etwa eine Bibliothek, die unter der GPL-Lizenz steht, in ein Softwareprogramm einbinden, so würde dies dazu führen, dass die gesamte Software unter der GPL stehen müsste. Da dies aber häufig von den Programmierern nicht gewünscht wird und um den Entwicklern Anreize zu geben, freie Bibliotheken zu benutzen, lockert die LGPL diese Auflagen (Grassmuck 2004, S. 290).

Die freieren Softwarelizenzen, die eine Privatisierung und Kommerzialisierung von Open Source Code vereinfachen, sind ein zweischneidiges Schwert. Einerseits könnte man argumentieren, dass eine solche Kommerzialisierung den Open-Source-Entwicklern per se keinen finanziellen Schaden zufügt und vielen Softwareprojekten nutzt. Andererseits wird dies sicherlich von vielen Teilnehmern an Open-Source-Projekten als Unge rechtigkeit und Trittbrettfahrermentalität der Unternehmen angesehen und könnte demzufolge die Motivation zur Teilnahme an diesen Projekten beeinträchtigen. Wir werden auf diese Anreizmechanismen zurückkommen, zunächst aber auf die Entwicklungsprinzipien in Open-Source-Projekten eingehen.

8.3 Open-Source-Projekte: Prinzipien und Motivation der Softwareentwickler

Der Entstehungsprozess von OSS unterscheidet sich grundlegend von der in kommerziell orientierten Unternehmen betriebenen Softwareentwicklung. Wir gehen nachfolgend auf die wesentlichen Unterschiede sowie deren Auswirkungen auf den Projektlauf und die Motivation der Beteiligten ein.

8.3.1 Ablauf und Organisation von Open-Source-Projekten

Open-Source-Projekte entstehen in der Regel dadurch, dass jemand ein Problem lösen möchte. Wir haben im letzten Abschnitt als Beispiel die Unzufriedenheit Stallmans mit

dem Xerox-Druckertreiber als Ausgangspunkt des GNU-Projekts beschrieben. Ein anderes prominentes Beispiel ist die Entwicklung des Betriebssystems LINUX. So wollte Linus Torvalds ein Unix-Betriebssystem auf seinem 386er PC laufen lassen. Als er kein adäquates fand, begann er selbst, ein solches System zu entwickeln und stellte seinen Quellcode im Internet zur Verfügung. Wie wir wissen, fand das Projekt reges Interesse und eine Vielzahl von Softwareentwicklern machte sich an die Weiterentwicklung. Selbst Torvalds war von diesem Verlauf überrascht und betonte immer wieder, dass er von einem solchen Erfolg sich niemals hätte träumen lassen.

OSS entsteht also evolutionär und verteilt. Wie bereits dargestellt, bezeichnet Raymond (Raymond 1999) diesen Entwicklungsprozess recht plastisch als Basar-Modell und stellt diesem das so genannte Kathedralen-Modell gegenüber, das er als Synonym für die klassische Softwareentwicklung sieht. Sicherlich sind die Unterschiede zwischen beiden Ansätzen sehr groß, wenn als Referenzpunkt für die kommerzielle Softwareentwicklung die ersten Modelle aus dem Software Engineering herangezogen werden. Allerdings lassen sich in den Konzepten der evolutionären sowie der agilen Softwareentwicklung durchaus Elemente der Ideen der Open-Source-Bewegung wiederfinden (Sharma et al. 2002).

Nach der Gründung ist es für den Erfolg des Projekts entscheidend, dass es frühzeitig gelingt, eine Gemeinschaft um diese Software herum aufzubauen. Dabei ist es stets förderlich, wenn bereits erste lauf- und testfähige Module vorliegen.

Von zentraler Bedeutung für die Durchführung des Projekts sind die Entscheidungsstruktur und Zusammensetzung des Entwicklungsteams. Dabei gibt es in größeren Open-Source-Projekten in der Regel ein Kernteam, das meistens aus Entwicklern besteht, die entweder am längsten daran arbeiten oder sehr viel Code beigetragen haben. Bei der Entwicklung des Apache Web Servers bestand dieses Kernteam aus 22 Programmierern aus sechs Ländern (Grassmuck 2004, S. 237).

Bei kleinen Open-Source-Projekten übernimmt meist der Gründer die Rolle eines so genannten „Maintainers“. Dieser ist für die Koordination des Projekts und insbesondere für die Qualitätssicherung zuständig. Bei größeren Projekten findet sich häufig eine zweistufige Struktur, in der verdiente Entwickler als Maintainer die Entwicklung eines Moduls koordinieren und zudem an den Grundsatzentscheidungen für das Gesamtprojekt mitwirken. Häufig, aber keinesfalls immer, hat der Gründer in diesem Kernteam eine herausgehobene Rolle. Beispielsweise hatte sich im Linux-Projekt eine Gruppe von fünf bis sechs Entwicklern herausgebildet, die eintreffenden Code testeten und selektierten und diese überprüften Quelltexte an Torvalds weitergaben, der schließlich auf dieser Basis die endgültigen Entscheidungen traf (Dietrich 1999). Im Apache-Projekt wird im Kernteam demgegenüber sogar nach demokratischen Prinzipien entschieden. So wird etwa in Mailinglisten darüber abgestimmt, ob bestimmte Module eingebunden werden sollen oder nicht.

Maintainer und Softwareentwickler, die viel Code zum Projekt beitragen, werden darüber hinaus durch eine Vielzahl weiterer Personen unterstützt, die die Software testen sowie Dokumentationen und Übersetzungen schreiben. Dabei ist es in vielen Fällen schwerer, solche unterstützenden Projektmitarbeiter als qualifizierte Softwareentwickler zu finden.

Dokumentieren ist für die meisten Entwickler einfach langweiliger; zudem gibt es auch kaum eine Möglichkeit, sich hiermit einen guten Namen zu machen.

Einen Abschluss von Open-Source-Projekten im klassischen Sinne gibt es in der Regel nicht. Vielmehr kann die Entwicklungstätigkeit eingestellt werden, wenn das Problem zur Zufriedenheit der Nutzer gelöst ist bzw. wenn der Maintainer oder wichtige Entwickler ihr Interesse am Projekt verloren haben. Liegt es an der Inaktivität des Maintainers, wird von der Gemeinschaft der Entwickler bisweilen auch ein neuer Maintainer bestimmt. Gelegentlich kommt es dann zu einer Aufspaltung des Projekts, das so genannte „forking“. Forking ist auch zu beobachten, wenn sich das Kernteam eines Projekts nicht über Grundsatzfragen einigen kann.

Eine wichtige Rolle in Open-Source-Projekten spielen internetbasierte Sourcecode-Verwaltungssysteme wie CVS (Concurrent Versions System). Über das CVS laden sich die beteiligten Programmierer die letzte Version eines Moduls herunter, bearbeiten und testen diese mit ihren eigenen Entwicklungswerkzeugen und kopieren die Ergebnisse über das CVS wieder in das Repository zurück. Wenn andere Entwickler gleichzeitig dieselbe Datei bearbeitet haben, werden die Veränderungen im Idealfall in eine neue Gesamtdatei im Repository verschmolzen. Lässt sich dieser Konflikt nicht automatisiert lösen, müssen sich die beteiligten Entwickler entsprechend untereinander verständigen. In regelmäßigen Abständen werden Zweige des Quellbaums durch das Kernteam im Repository als neues Release ausgezeichnet.

8.3.2 Zur Motivation der Beitragenden

Open-Source-Projekte basieren auf der Zusammenarbeit einer häufig weltweiten Gemeinschaft von Softwareentwicklern. Diese engagieren sich freiwillig und zumindest die meisten werden für ihre Mitarbeit nicht entlohnt. Durch ihr Mitwirken nehmen sie also Opportunitätskosten in Kauf, sei es in Form von entgangener Freizeit, dem Verzicht auf alternative Verdienstmöglichkeiten oder einer Vernachlässigung ihrer beruflichen Aufgaben. In besonderem Maße gilt dies natürlich für Mitglieder in den Kerngruppen eines Open-Source-Projekts.

Damit stellt sich die Frage, aus welchen Motiven sich Entwickler an Open-Source-Projekten beteiligen. Ein Zweig der ökonomischen Literatur versucht das Phänomen so zu erklären, dass die Entwickler durchaus einen privaten Nutzen aus ihrer Mitarbeit ziehen (Lerner und Tirole 2002). Andere Arbeiten gehen davon aus, dass die Programmierer überwiegend intrinsische Motive verfolgen, die sie zu einer Mitarbeit bewegen (Kollock 1999). Franck (2003) unterscheidet vor diesem Hintergrund die Entwickler in

- Rentensucher und
- Spender.

Der Rentensucher verhält sich wie ein klassischer „*homo oeconomicus*“. Die Entwickler suchen also nach Vorteilen jenseits der klassischen Entlohnung. Tatsächlich haben empirische Studien vielfach gezeigt, dass Beitragende sich von ihrem Mitwirken eine Steigerung ihrer Reputation am Arbeits- oder Kapitalmarkt versprechen, eine Verbesserung ihres Know-hows erwarten oder von einer Erleichterung ihrer täglichen Arbeit ausgehen (Lerner und Tirole 2002). Diese Entwickler engagieren sich also genau dann, wenn sie davon einen positiven Nettonutzenzuwachs erwarten – oder anders ausgedrückt: Sie suchen eine Rente.

Klassisch-rationale Motive alleine erklären das Phänomen der Mitarbeit in Open-Source-Projekten allerdings nicht vollständig. Empirische Untersuchungen zeigen immer wieder, dass sich Beitragende von ihrem Engagement in Open-Source-Projekten auch Spaß und Unterhaltung sowie das Voranbringen der Open-Source-Idee an sich versprechen. Darüber hinaus verfolgen viele Open-Source-Entwickler auch andere Ziele, wie Informationsfreiheit. Bisweilen sind die Aktivitäten nicht zuletzt direkt gegen den Marktführer Microsoft gerichtet, um dessen „Quasimonopol“ zu brechen. Personen, die primär dieses Motiv verfolgen, bezeichnet Franck treffend als Spender (Franck 2003). Sie sind der Überzeugung, dass sie ihre Zeit für eine gute Sache einsetzen. Dabei sind gerade solche Open-Source-Projekte für Spender attraktiv, die unter der GPL oder einer ähnlichen Lizenz geführt werden, da sie nicht befürchten müssen, dass ihre „Spende“ letztlich doch kommerzialisiert wird.

Erfolgreichen Open-Source-Projekten gelingt es häufig, eine Governance-Struktur zu schaffen, die sowohl für Rentensucher als auch für Spender attraktiv ist. Rentensucher können ihre Reputation steigern, da durch die Offenheit des Quellcodes ihre Programmierbeiträge sichtbar und belegbar sind. Für Spender wird ein Open-Source-Projekt sogar umso attraktiver, je mehr Rentensucher sich beteiligen, da dadurch die Wahrscheinlichkeit des Projekterfolgs grundsätzlich steigt. Damit kann die sonst häufig zu beobachtende Verdrängung zwischen Rentensuchern und Spendern verhindert werden (Franck 2003).

Allerdings gilt bei weitem nicht mehr für alle Open-Source-Projekte, dass sich das Team aus unentgeltlich arbeitenden Hobbyentwicklern zusammensetzt (Brügge et al. 2004, S. 101 f.). Vielmehr gibt es eine Reihe von Open-Source-Projekten, in denen Programmierer und weitere Spezialisten als Angestellte von Unternehmen mitwirken. So wurden ca. 30 % der Entwickler der Open-Source-Plattform „SourceForge.net“ für ihre Entwicklungsarbeit bezahlt. Auch geht die Initiative zu einem Open-Source-Projekt nicht selten von einem Unternehmen selbst aus. Beispielsweise waren die Initiatoren des bereits erwähnten Apache-Projekts Mitarbeiter von Unternehmen, die dort für den Betrieb von Web Servern verantwortlich sind. Aber auch in diesem Fall gilt, dass – falls das Projekt unter der GPL oder einer GPL-ähnlichen Lizenz betrieben wird – diese Konstellation eine Zusammenarbeit zwischen Rentensuchern (bezahlte Programmierer) und Spendern nicht verhindert, sondern sogar fördert.

8.4 Open Source Software aus Sicht des Anwenders

OSS hat aus Anwendersicht zunächst einen geradezu bestechenden Vorteil: Sie ist frei verfügbar, d. h., die sonst üblichen Lizenzkosten für eine Standardsoftware bzw. die Ausgaben für eine unternehmensspezifische Entwicklung entfallen. Aber natürlich sind die Lizenzkosten nicht die einzigen relevanten Parameter, die bei der Entscheidung über den Softwareeinsatz zu beachten sind.

Um dies deutlich zu machen, betrachten wir im Weiteren exemplarisch eine Studie zur Ablösung von Windows XP und Office-Lösungen durch Linux und Open-Source-Anwendungssoftware bei der Stadt München. Der Fall wurde nicht zufällig ausgewählt, da OSS überproportional oft im öffentlichen Sektor eingesetzt wird. Die Studie wurde im Jahr 2002 durch das Beratungsunternehmen Unilog durchgeführt (Unilog 2002). Berücksichtigt wurden die direkten Kosten der zu diesem Zeitpunkt rund 14.700 PC-Arbeitsplätze der Münchner Stadtverwaltung. Die Studie kommt zu dem Ergebnis, dass ein Wechsel auf eine OSS in diesem Fall zumindest aus kurzfristiger, rein kostenorientierter Perspektive nicht anzuraten ist, weil der Wegfall der Lizenzkosten durch deutlich höhere Migrations- und Schulungskosten überkompensiert wird. Abbildung 8.1 zeigt die Höhe der Kostenarten für die beiden Alternativen in Millionen Euro.

Trotz dieses Ergebnisses der kurzfristigen Kostenbetrachtung hat sich die Stadt München zunächst für einen Umstieg auf die Open-Source-Lösung entschieden und dies mit einer deutlich geringeren Abhängigkeit vom Hersteller Microsoft bei gleicher Leistungsfähigkeit bzw. einem identischen Basisnutzen der beiden Alternativen begründet.

Bislang lassen sich noch keine abgesicherten allgemeinen Aussagen zur Wirtschaftlichkeit von OSS treffen. Die hierzu vorliegenden Untersuchungen kommen zu recht unterschiedlichen Ergebnissen (Brügge et al. 2004). So weist zum Beispiel eine Studie

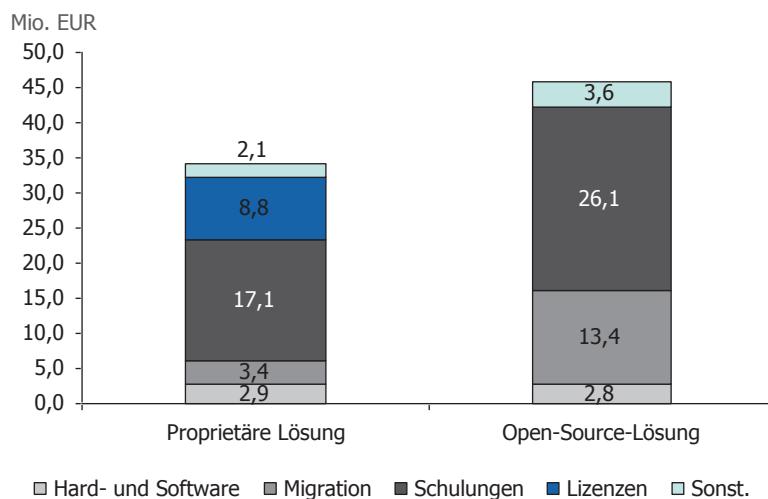


Abb. 8.1 Kurzfristiger Kostenvergleich für die Stadt München

von Berlecon Research darauf hin (Berlecon Research 2002), dass neben dem Wegfall der Lizenzkosten auch geringere Kosten für Einführung und Administration sowie eine höhere Stabilität des Systembetriebs für Open-Source-Lösungen sprechen. Doch genau die letztgenannten Vorteile waren in unserem Beispiel nicht zu beobachten. Die unterschiedlichen Ergebnisse können dabei auch durch die in der Praxis zum Teil sehr problematische Abgrenzung der Kosten begründet sein. Zudem sind sowohl die Debatte als auch die Studien zu dem Thema nicht selten ideologisch vorbelastet.

8.5 Engagement kommerzieller Softwareanbieter

Im vorangehenden Abschnitt sind wir u. a. der Frage nachgegangen, warum Entwickler und andere Software-Spezialisten an Open-Source-Projekten teilnehmen. Im Folgenden wird untersucht, welche Motive Unternehmen mit der Förderung solcher Projekte verfolgen. Hierfür gibt es insbesondere drei wesentliche Gründe (Hecker 1999; Raymond 1999; Henkel 2004):

- die Unterstützung des Verkaufs komplementärer Produkte und Dienstleistungen,
- die Verwendung von OSS in eigenen Produkten sowie
- die Reduktion der Marktmacht proprietärer Software von Wettbewerbern.

Am häufigsten findet sich in der Literatur der Hinweis auf die Möglichkeiten des Verkaufs komplementärer Produkte und Dienstleistungen. Es handelt sich somit um eine Follow-the-free-Strategie, die wir bereits in Abschn. 3.3.2.6 vorgestellt haben. Anders ausgedrückt: OSS schafft beim Nutzer einen zusätzlichen Bedarf; diesen kann das anbietende Unternehmen decken und so indirekte Netzeffekte generieren und nutzen. Derartige Komplemente können Hardware, Softwarelösungen sowie ergänzende Dienstleistungen, wie etwa Schulungen, sein. Ein Beispiel hierfür ist das Geschäftsmodell des Linux-Distributors SuSe.

Der Linux-Distributor SuSe

Linux ist ein frei verfügbares Betriebssystem, das Multitasking und Multiusing unterstützt. Das System wird inzwischen in zahlreichen Bereichen (Desktop, Server, Mobiltelefone, Router etc.) eingesetzt. Meist werden in der Praxis fertige Softwarepakete genutzt, so genannte Distributionen. Die Firma SuSe war das erste Unternehmen in Deutschland, das erfolgreich eine Linux-Distribution breiter vermarktet hat – und damit eines der ersten Unternehmen weltweit, das Linux als Basis eines Geschäftsmodells nutzte. Den Einstieg in den Markt fand SuSe über das erste deutschsprachige Installationsprogramm für Linux.

Hintergrund

Im Jahre 1992 haben Roland Dyroff, Burchard Steinbild, Hubert Mantel und Thomas Fehr die Software und Systementwicklungsgesellschaft mbH gegründet. Ihr erstes Produkt war lediglich die Weiterentwicklung einer bereits bestehenden Linux-Distribution, bis SuSe dann 1996 ihre erste selbst entwickelte Distribution auf den Markt brachte. Hauptsitz der Firma war zunächst Fürth, bevor 1998 entschieden wurde, diesen nach Nürnberg zu verlegen. Zusätzlich eröffnete SuSe 1997 ein Büro in Oakland. Sechs nationale und drei weitere internationale Standorte (Italien, Tschechische Republik, Großbritannien) folgten. Im Jahr 2004 wurde SuSe von der amerikanischen Firma Novell aufgekauft. Der Kaufpreis betrug 210 Mio. \$. Zu diesem Zeitpunkt hatte SuSe ca. 380 Mitarbeiter weltweit, die alle von Novell übernommen wurden. Der Umsatz von SuSe betrug 37 Mio. €. Mit dem Kauf von SuSe wurde Novell zu einem der weltweit führenden Anbieter für Linux-Komplettlösungen. In 2011 wurde Novell von Attachmate für 2,2 Mrd. \$ übernommen.

Leistungssportfolio

Die Produktpalette von Novell enthält sowohl Angebote für Firmen als auch für private Nutzer. Firmenseitig bietet SuSe eine Komplettlösung, die SuSe-Linux-Enterprise-11-Platform, an. Sie beinhaltet Datenbank, Server, Desktop und Hardwareverwaltung. Mit openSuSe spricht das Unternehmen private Anwender an; die Entwicklung ist öffentlich. Bis Dezember 2006 hieß diese Distribution SuSe-Linux. Um sie von den kostenpflichtigen Produkten besser abzuheben, wurde sie in openSuSe umbenannt.

Hauptlösquelle für Novell sind jedoch Serviceleistungen. Daher enthält das Leistungssportfolio ein umfassendes Angebot an Support, Beratung und Training. Zu nennen sind hier heute vor allem die zertifizierten Schulungen für Unternehmen. Um die Serviceorientierung des Unternehmens stärker zu betonen, gibt es außerdem eine Reihe kostenloser Services, wie z. B. ein Diskussionsforum, Datenbanken und Dokumentationen, die dem Nutzer bei der Beantwortung von Fragen helfen.

Quellen: www.novell.com, www.opensuse.org

Ebenfalls relativ häufig verwenden Unternehmen OSS als Teil eigener Produkte und Lösungen. Daher ist es naheliegend, diese Entwicklung auch finanziell zu unterstützen. Hierzu gehört die Integration von OSS in so genannte „eingebettete Systeme“. Charakteristisch hierfür ist die dedizierte Verbindung von Hardware- und Softwarekomponenten, die dann auch genau nur zu einem spezifischen Zweck eingesetzt werden kann. Ein typisches Anwendungsbeispiel ist die Maschinensteuerung, für die häufig Linux und andere OSS eingesetzt werden. Ein anderes prominentes Beispiel ist der digitale Videorecorder TiVO, in dem sich ein solches eingebettetes System befindet. Ist die im Produkt weitgehend unsichtbare verwendete Software unter der GPL lizenziert, kann ein Unternehmen Erlöse nur durch den Verkauf (spezifischer) Hardware erzielen. Im Kern ist dies wieder die Idee einer komplementären Ergänzung, die wir oben dargestellt haben.

Ein drittes mögliches Motiv für Softwareanbieter, sich aktiv an Open-Source-Projekten zu beteiligen und Mitarbeiter hierfür bereitzustellen, ist die Begrenzung der Marktmacht von konkurrierenden Anbietern. Ein Beispiel ist das Engagement von IBM bei der Entwicklung von Linux. Auf diese Weise gelang es schließlich, der Dominanz von Microsoft bei PC-basierten Betriebssystemen zumindest etwas Einhalt zu gebieten. Mit den beträchtlichen Investitionen in die Linux-Entwicklung ist für IBM insbesondere der große Vorteil einer Reduzierung der Abhängigkeit von Microsoft als Betriebssystemanbieter verbunden.

Alternativ zur Bezahlung von Mitarbeitern für ihre Aktivitäten in Open-Source-Projekten engagieren sich einige Unternehmen – so auch IBM –, indem sie intern entwickelte Software freigeben und der Open Source Community spenden. Bekannte Beispiele hierfür sind Mozilla und Open-Office sowie die Übergabe von Java und JDK von Sun Microsystems an die Open-Source-Gemeinde.

8.6 Quelloffene ERP-Systeme

Wenn heute von OSS die Rede ist, wird in der Regel von bekannten und erfolgreichen Projekten, wie etwa Linux, gesprochen. So werden weltweit mehr als 30 Millionen Kopien dieses Betriebssystems eingesetzt. Dies entspricht einem Marktanteil bei den Betriebssystemen für Server von 28 % (Lindner 2013). Weitere Erfolgsgeschichten sind etwa Apache Webserver, die Entwicklungsumgebung Eclipse oder mySQL.

Bei Betrachtung der bekannten und erfolgreichen Open-Source-Projekte fällt auf, dass sich OSS überwiegend in verschiedenen system- und entwicklungsnahen Bereichen etabliert hat und dort zum Standard wurde. In Anbetracht dessen stellt sich die Frage, ob und inwieweit die OS-Idee auch für ERP-Systeme geeignet ist und damit zur Konkurrenz für etablierte Softwareanbieter in diesem Marktsegment, wie etwa SAP oder Oracle, werden kann.

Eine zentrale Fragestellung ist dabei, ob der Anwendungsbereich ERP für viele OS-Entwickler thematisch spannend genug ist bzw. sie über die entsprechenden ökonomischen Kenntnisse verfügen. Eine Untersuchung der größten Plattform für OS-Software, SourceForge.net, zeigt deutlich, dass Entwickler durchaus an betriebswirtschaftlichen Fragestellungen interessiert sind: Über 630 Projekte sind auf der Plattform dem Themengebiet ERP zugeordnet. Ein bekanntes Beispiel, das zeigt, dass sich OS-Software auch über die Infrastrukturebene hinaus in Unternehmen etablieren kann, ist etwa die Customer-Relationship-Management-Software SugarCRM (Sterne und Herring 2006). Bisher weniger bekannt sind quelloffene ERP-Systeme.

Eine Auswahl quelloffener und produktiver OS-ERP-Lösungen ist in der nachstehenden Tab. 8.2 dargestellt (Buxmann und Matz 2009). Dabei sind jeweils das verantwortliche Unternehmen bzw. die Organisation, die Eigenschaften des Softwareprojekts und die Charakteristika des verwendeten Geschäftsmodells abgebildet. Bei allen Angeboten handelt es sich um internationale, mehrsprachige Projekte. Unberücksichtigt bleiben OS-

Tab. 8.2 Eine Übersicht quelloffener ERP-Systeme. (Buxmann und Matz 2009)

	ADempiere	Compiere	ERP5 Express	Openbravo	OpenERP
<i>Anbieter</i>					
<i>Anbieter bzw. Trademark-Besitzer</i>	ADempiere	Compiere, Inc.	Nexedi SA	Openbravo, S.L.	Tiny company
<i>Standort</i>	USA	USA	Frankreich	Spanien	Belgien
<i>Website</i>	adempiere.com	compiere.com	erp5.org	openbravo.com	openerp.com
<i>Softwareprojekt</i>					
<i>Projektstart</i>	2006	1999	2007	2006	2005
<i>Typ</i>	Web-anwendung und Rich-Client	Rich-Client (Webinterface ist kostenpflichtig)	Webanwendung	Webanwendung	Webanwendung und Rich-Client
<i>Registrierte Entwickler bei SourceForge</i>	89, davon 9 Admins	75, davon 2 Admins	Nicht registriert	81, davon 15 Admins	Nicht registriert
<i>Programmiersprache</i>	Java	Java, JavaScript	Python	Java, JavaScript	Java, Python
<i>Unterstützte Plattformen (serverseitig)</i>	Plattformunabhängig	Plattformunabhängig	Linux, MacOSX, Unix, Windows	BSD, Linux, Solaris, Windows	Linux, MacOSX, Unix, Windows
<i>Unterstützte Datenbanken</i>	Oracle, Postgres	Oracle, Postgres	DB2, MySQL, Oracle, Postgres	Oracle, Postgres	Postgres
<i>Letzte Version</i>	1.9.2011 (Release 3.7.0 LTS)	(Release 3.8.2)	(Release 5.4.6)	27.6.2014 (Release 3.0.22913)	(Release 5.0.0)
<i>Geschäftsmodell</i>					
<i>Zielgruppe</i>	Keine Informationen	Für alle Größen	Für alle Größen	Für alle Größen	Keine Informationen
<i>Lizenz</i>	GPL	GPL für Community- und Standard-Edition und kommerzielle Lizenz für Professional Edition	GPL	Openbravo Public License (basiert auf Mozilla Public License)	GPL
<i>Preisdifferenzierung Software</i>	Keine	Community-, Standard- und Professional-Edition	Keine	Keine	Modul-abhängige Bepreisung

Tab. 8.2 Fortsetzung

	ADempiere	Compiere	ERP5 Express	Openbravo	OpenERP
Preisdifferenzierung Services	Keine	Community-, Standard- und Professional-Edition	Community Version, Starter-, Premium-, Elite-Pack	Community Edition, SMB Network, Basic Network, OEM Network	Modul-abhängige Bepreisung
	<i>OpenTaps</i>	<i>xTuple ERP</i>	<i>SQL-Ledger</i>	<i>LX-Office</i>	<i>webERP</i>
<i>Anbieter</i>					
Anbieter bzw. Trademark-Besitzer	Open Source Strategies, Inc.	xTuple	DWS Systems, Inc.	Lx-System - Holger Lindemann und LINET Services GbR	Administrator: Phil Daintree
Standort	USA	USA	Kanada	Deutschland	Neuseeland
Webseite	opentaps.org	xtuple.com	sql-ledger.org	lx-office.org	weberp.org
<i>Softwareprojekt</i>					
Projektstart	2005	2002	2000	2004	2003
Typ	Webanwendung	Rich-Client	Webanwendung	Webanwendung	Webanwendung
Registrierte Entwickler bei SourceForge	38, davon 1 Admin	39, davon 9 Admins	Nicht registriert	4, davon 3 Admins	9, davon 2 Admins
Programmiersprache	Java	C++, JavaScript	Perl	Perl, PHP	PHP
Unterstützte Plattformen (serverseitig)	Linux, MacOSX, Unix, Windows	Linux, MacOSX, Windows	Plattformunabhängig	Linux, Unix	Plattformunabhängig
Unterstützte Datenbanken	MySQL, Postgres	Postgres	Postgres	Postgres	MySQL
Letzte Version	23.2.2011 (Release 1.5)	4.4.2014 (Release 4.4)	(Release 3.0.6)	28.2.2014 (Release 3.1)	8.2.2014 (Release 4.11.3)
<i>Geschäftsmodell</i>					
Zielgruppe	Keine Informationen	Kleine und mittlere Unternehmen	Keine Informationen	Keine Informationen	Kleine Unternehmen

Tab. 8.2 Fortsetzung

	ADempiere	Compiere	ERP5 Express	Openbravo	OpenERP
<i>Lizenz</i>	Honest Public License (basiert auf GPL) und kommerzielle Lizizenzen	PostBooks-Edition unter Common Public Attribution License 1.0 und Standard- sowie OpenManufacturing-Edition unter kommerzieller Lizenz	GPL	Artistic License, GPL und LGPL	GPL
<i>Preisdifferenzierung Software</i>	Keine	PostBooks-, Standard- und OpenManufacturing-Edition	Keine	Keine	Keine
<i>Preisdifferenzierung Services</i>	Die kommerzielle Lizenz bietet erweiterten Supportumfang	Keine	Nach Nutzergruppen: User, Tech, Dev	Keine	Keine

Projekte, die ausschließlich einzelne Funktionen eines ERP-Systems abdecken, wie das bereits angesprochene System SugarCRM oder GnuCash, welches die Finanzbuchhaltung unterstützt.

Bei einer genaueren Betrachtung der Geschäftsmodelle der Anbieter zeigen sich deutliche Unterschiede: Systeme wie ADempiere und webERP werden hauptsächlich über das Engagement der Community von freien Entwicklern vorangetrieben. Die Software steht im vollen Funktionsumfang unter einer GPL-Lizenz zur Verfügung, und kostenpflichtige Services werden ausschließlich von Drittanbietern vermarktet. Andere Angebote sind kommerzieller ausgerichtet. Ihr Geschäftsmodell ähnelt hierbei dem klassischer Softwareanbieter und unterscheidet sich im Wesentlichen nur dadurch, dass sie neben den kommerziellen Softwarepaketen auch eine Version – häufig lediglich mit Basisfunktionalitäten ausgestattet – unter einer OS-Lizenz vertreiben. Ein solches Duales Lizenzmodell (Mundhenke 2007, S. 130–131; Hecker 1999, S. 49) kann sowohl aus Anbieter- als auch aus Kundensicht vorteilhaft sein: Die Kunden haben die Möglichkeit, die freie Version uneingeschränkt zu testen und zu nutzen. Bei Bedarf können sie zu den proprietären Versionen wechseln und die damit verbundenen Funktionserweiterungen und Dienstleistungen nutzen. Der Anbieter kann auf diese Weise gleichzeitig verschiedene Kundengruppen ansprechen und die Nutzerbasis erweitern. Darüber hinaus versprechen sich viele Anbieter

von der Herausgabe einer Softwareversion unter einer OS-Lizenz einen positiven Marketingeffekt.

Vor dem Hintergrund der Fragestellung, ob sich die Entwicklung von OS-Software im ERP-Bereich auch durch das Engagement freier Entwickler vorantreiben lässt, liefert eine empirische Untersuchung der Entwicklungsaktivitäten sowie der Beiträge in Foren der SourceForge-Plattform (www.sourceforge.org) Aufschluss. SourceForge stellt Entwicklern und Nutzern verschiedene Werkzeuge zur Kommunikation und zur Softwareentwicklung gebührenfrei zur Verfügung. Den registrierten Mitgliedern wird so die Möglichkeit gegeben, sich sowohl an Projekten zu beteiligen als auch Leistungen der Community in Anspruch zu nehmen bzw. zu erbringen. Jedem SourceForge-Projekt sind in der Regel mehrere Foren zugeordnet. Innerhalb jedes Forums können so genannte Threads eröffnet werden, die in der Regel eine Frage beinhalten, auf die registrierte SourceForge-Nutzer antworten können. Nach eigenen Angaben hostet SourceForge derzeit über 430.000 OS-Projekte mit fast vier Millionen registrierten Nutzern. Voraussetzung für die Registrierung eines Projekts ist, dass die Software unter einer von der OSI anerkannten Lizenz steht. In Zusammenarbeit mit der University of Notre Dame, Indiana (USA) stellt SourceForge einen Auszug aus der Datenbank zu Forschungszwecken zur Verfügung (vgl. Van Antwerp und Madey 2008). Dadurch wird es möglich, die Projekte über die standardmäßigen Statistiken von SourceForge hinaus zu untersuchen.

Zunächst wollen wir einen Blick auf das Ranking der aktivsten Projekte von SourceForge werfen. Die Aktivität eines Projekts setzt sich bei SourceForge aus den Besucherzahlen der jeweiligen Projektwebseite, der Entwicklungstätigkeit sowie der Kommunikation, beispielsweise innerhalb der Foren, zusammen. Das Standardranking basiert auf einer kumulierten Auswertung aller Daten seit dem Projektbeginn von SourceForge.

Bei Betrachtung dieses Rankings zeigt sich, dass OS-ERP-Projekte nicht unter den ersten 20 zu finden und nur sechs innerhalb der Top1000 aufgeführt sind. Demgegenüber zeichnen aktuelle Rankings, die lediglich einen kurzfristigen Zeitraum der jüngsten Vergangenheit umfassen, beispielsweise Project of the Month, ein anderes Bild. Hier sind OS-ERP-Projekte zu finden. So war 2013 beispielsweise Potbooks Project of the Month. Diese Rankings sagen natürlich nichts über die Qualität der entsprechenden Beiträge aus, sind aber ein Hinweis darauf, dass das OS-Modell grundsätzlich auch für ERP-Software geeignet zu sein scheint.

Für einen weiter gehenden Vergleich zwischen ERP- und anderen OS-Projekten sollen nun die Entwicklungs- und Kommunikationsaktivitäten detaillierter analysiert werden. Hierbei werden Projekte in die Stichprobe einbezogen, welche die folgenden Kriterien erfüllen:

- für das Projekt sind mindestens zwei Entwickler registriert,
- das Projekt existiert seit mindestens einem Jahr und
- die Foren enthalten mindestens einen Eintrag.

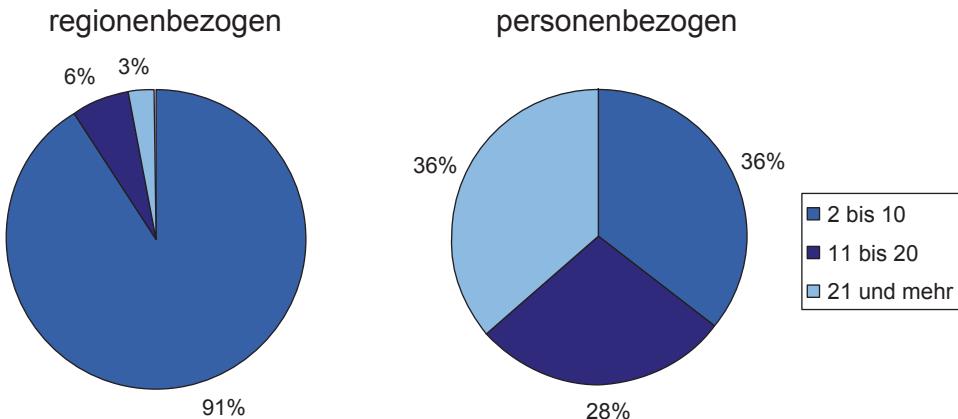


Abb. 8.2 a Größe der Teilnehmergruppen in OS-ERP-Projekten. **b** Gruppengröße in der Vergleichsgruppe. (Buxmann und Matz 2009)

Diese Abgrenzung filtert die wenig aktiven und jungen Projekte aus der zu untersuchenden Stichprobe heraus. Für die ERP-Gruppe konnten so 208 Projekte identifiziert werden. Im nächsten Schritt wurden zufällig 208 Projekte ausgewählt, die nicht der ERP-Gruppe zugeordnet sind.

Für die Analyse wurden zunächst die Größen der Communities miteinander verglichen. Unter der Voraussetzung, dass Projekte mit einem einzelnen registrierten Teilnehmer unberücksichtigt bleiben, sind durchschnittlich 5,7 Teilnehmer in OS-ERP-Projekten registriert. Wie Abb. 8.2a zeigt, sind bei über 90 % dieser Projekte zwischen zwei und zehn Teilnehmer registriert. An insgesamt rund 9 % der Projekte sind elf und mehr Beitragende beteiligt. Das größte Projekt (Openbravo) hat 77 Teilnehmer.

Die Gruppengröße in den anderen zufällig gewählten Projekten liegt demgegenüber bei durchschnittlich 26,7 und maximal 391 Teilnehmern. Abbildung 8.2b zeigt, dass in den Projekten jeweils ähnlich viele Teilnehmer innerhalb der drei Kategorien, zwei bis zehn, elf bis 20 sowie 21 und mehr, registriert sind. Im Ergebnis lässt sich also festhalten, dass in den OS-ERP-Projekten deutlich weniger Nutzer bzw. Entwickler registriert sind als in der Vergleichsgruppe.

Im nächsten Schritt soll das Kommunikationsverhalten der Teilnehmer in den verschiedenen OS-Projekten untersucht und verglichen werden. Dazu wurden zunächst die Foren der OS-ERP-Projekte betrachtet. Die Analyse ergibt, dass rund 80 % der Nutzer Threads eröffnen, während sich knapp 48 % an der Beantwortung der verschiedenen Themen beteiligen. Dabei bleiben gut 32 % der eröffneten Beiträge unbeantwortet.

Vergleicht man diese Ergebnisse nun mit der Vergleichsgruppe, so zeigen sich überraschende Parallelen: Auch dort eröffnen rund 80 % der Teilnehmer Themen. Gleichzeitig beteiligen sich insgesamt 35 % aller Beteiligten an der Beantwortung, was rund 12 Prozentpunkte weniger als bei den ERP-Foren sind. Unter den Foreneinträgen der zufällig gewählten Projekte bleiben gut 72 % unbeantwortet. Dabei bleibt unklar, ob ein Eintrag zufriedenstellend beantwortet wurde oder ob es sich beispielsweise um eine Rückfrage oder

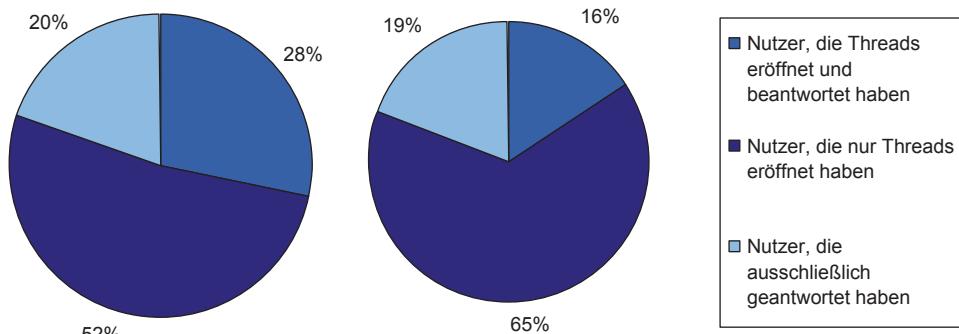


Abb. 8.3 a Kommunikationsverhalten in OS-ERP-Foren. b Kommunikationsverhalten in der Vergleichsgruppe. (Buxmann und Matz 2009)

Ergänzung gehandelt hat. Um solche Rückfragen des Thread-Eröffners herauszufiltern, wurden nur Einträge von Nutzern gezählt, die diesen Thread nicht selbst eröffnet haben. Bei Betrachtung der Ergebnisse fällt auf, dass sich weniger als ein Drittel ausschließlich an der Beantwortung der online gestellten Threads beteiligt und damit selbst keine eigenen Threads eröffnet haben. Eine Übersicht der Aufteilung zeigt (Abb. 8.3).

Als Ergebnis können wir eine tendenziell höhere Aktivität der Nutzer bzw. Entwickler in den OS-ERP-Foren festhalten. So ist etwa der Anteil der Foreneinträge, auf die nicht reagiert wird, in der Gruppe der zufällig gewählten Projekte mehr als doppelt so hoch.

Im Weiteren soll untersucht werden, ob sich die Antwortzeiten – also die Zeit zwischen der Fragestellung und der ersten Antwort (Lee et al. 2009, S. 431) – voneinander unterscheiden. Abbildung 8.4 zeigt einen Vergleich zwischen den Antwortzeiten für OS-ERP-Projekte mit denen der anderen Projekte. Insgesamt wird deutlich, dass innerhalb eines Tages über 60 % der Threads beantwortet sind und nach einer Woche über 83 %. Hier zeigen sich keine wesentlichen Unterschiede zwischen den OS-ERP- und den anderen Projekten.

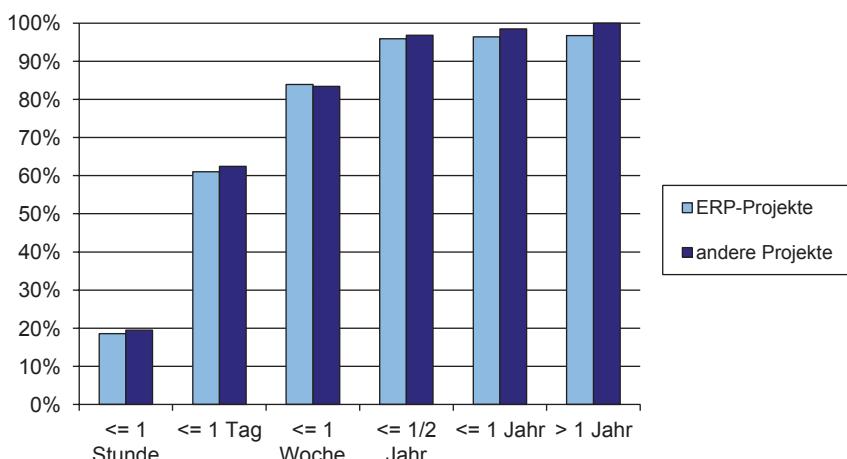


Abb. 8.4 Kumulierte Antwortzeiten in den ERP- und anderen Foren. (Buxmann und Matz 2009)

Zusammenfassend lässt sich festhalten, dass in der OS-Welt mittlerweile durchaus Alternativen zu proprietären ERP-Angeboten existieren. Unsere Analyse zeigt, dass einerseits zwar tendenziell weniger Entwickler an OS-ERP-Projekten beteiligt sind, diese aber andererseits umso intensiver miteinander kommunizieren, und Fragen in den Foren häufiger und bereitwilliger beantwortet werden. Bernerkenswert ist auch, dass die Rankings der aktivsten Projekte der SourceForge-Datenbank zuletzt immer häufiger von OS-ERP-Projekten angeführt werden.

Natürlich sind diese Ergebnisse insofern zu relativieren, als sie, wie bereits dargestellt, nichts über die Qualität der erbrachten Leistungen aussagen – beispielsweise wie brauchbar bestimmte Antworten in Foren waren oder wie intensiv sich Programmierer an der Entwicklung beteiligt haben. Darüber hinaus ist zu beachten, dass es auf Basis der von uns genutzten Daten nicht möglich ist, freiwillige private Entwickler von bezahlten Programmierern zu unterscheiden. So beschäftigen viele Software- und IT-Unternehmen eine Vielzahl von Programmierern, die sich ausschließlich oder überwiegend um die Weiterentwicklung bestimmter OS-Projekte kümmern. Die Ergebnisse der Analyse sind demzufolge nur als Indiz zu werten, dass das OS-Modell auch für ERP-Software erfolgreich sein könnte. Daraus können jedoch noch keine Einschätzungen der zukünftigen Marktchancen für OS-ERP-Anwendungen abgeleitet werden. Vielmehr muss beachtet werden, dass auf Softwaremärkten besondere Spielregeln gelten und sich nicht zwangsläufig die beste Lösung durchsetzen wird. So existieren auf Softwaremärkten – wie wir in Kap. 2 ausführlich diskutiert haben – erhebliche Lock-in-Effekte. Ein Wechsel ist für die Anwender aufgrund des Pinguineffekts mit einem hohen Risiko sowie hohen Switching Costs verbunden.

Die Konsequenz für OS-ERP-Software lautet, dass es letztlich nicht ausreicht, ein gutes Produkt anzubieten. Vielmehr gilt es, die Softwarelösungen auf der Basis attraktiver und kundenfreundlicher Angebote zu vermarkten. Beispielsweise lässt sich mithilfe von ökonomischen Simulationsmodellen zeigen, dass auf Softwaremärkten der Preis häufig der bessere Hebel ist, um Marktanteile zu erhöhen, als eine Erweiterung des Funktionsumfangs (Buxmann 2002). Dies gilt für einen Anbieter umso mehr, je geringer seine Marktanteile im Vergleich zum Marktführer sind. Vor diesem Hintergrund erscheint es problematisch, dass einige Anbieter von OS-ERP-Software mit Preisen in den Markt gehen, die mit denen des Marktführers SAP vergleichbar sind.

Auch die Außendarstellung einiger Anbieter erweckt den Eindruck, dass sie noch Nachholbedarf in Bezug auf eine professionelle Kommunikations- und Vertriebsstrategie haben. Insofern wird es OS-ERP-Software nicht leicht haben, etablierten ERP-Anbietern kurzfristig Marktanteile abzunehmen.

Jedoch sollte nicht vergessen werden, dass kaum ein Experte eine so starke Verbreitung von OS-Software in verschiedenen systemnahen Bereichen vorhergesehen hat, wie wir sie heute haben. Vielleicht gelingt es der Open-Source-Gemeinde ja auch im Bereich ERP-Systeme, die Softwareindustrie nachhaltig zu verändern.

Literatur

- Berlecon Research (2002) Use of open software in firms and public institutions. Evidence from Germany, Sweden and UK. Berlin
- Brügge B, Harhoff D, Picot A, Creighton O, Fiedler M, Henkel J (2004) Open-source-software. Springer, Berlin
- Buxmann P (2002) Strategien von Standardsoftware-Anbietern: Eine Analyse auf Basis von Netzeffekten. *Z betriebswirtsch Forsch* 54:442–457
- Buxmann P, Matz J (2009) ERP-Software: Von der Kathedrale zum Basar. Controlling und Management, Sonderheft 3:18–23
- Dietrich O (1999) In den Tiefen des Kernel. Interview mit Linux-Entwickler Alan Cox. *c't* 25:34
- Franck E (2003) Open Source aus ökonomischer Sicht. *Wirtschaftsinformatik* 45:527–532
- Grassmuck V (2004) Freie Software – Zwischen Privat- und Gemeineigentum. Bundeszentrale für politische Bildung, Bonn
- Hecker F (1999) Setting up shop: the business of open-source-software. *IEEE Softw* 16:45–51
- Henkel J (2004) Open source software from commercial firms – tools, complements, and collective invention. *Z Betriebswirtsch Ergänz* 4:1–23
- Kollock P (1999) The economies of online cooperation: gifts and public goods in cyberspace. In: Smith MA, Kollock P (Hrsg) *Communities in cyberspace*. Routledge, London, S 220–239
- Lee SYT, Kim H-W, Gupta S (2009) Measuring open source software success. *Omega Int J Manage Sci* 37:426–438
- Lerner J, Tirole J (2002) Some simple economics of open source. *J Ind Econ* 52:197–234
- Lindner M (2013) Servermarkt: x86 und Linux weiter auf dem Vormarsch. <http://www.pro-linux.de/news/1/20562/servermarkt-x86-und-linux-weiter-auf-dem-vormarsch.html>
- Mundhenke J (2007) Wettbewerbswirkungen von Open-Source-Software und offenen Standards auf Softwaremärkten, Berlin
- Perens B (1999) The open source definition. In: DiBona C, Ockman S, Stone M (Hrsg) *Open source: voices from the open source revolution*. O'Reilly, Sebastopol, S 171–188
- Raymond ES (1999) The cathedral and the bazaar. O'Reilly, Sebastopol
- Sharma S, Sugumaran V, Rajagopalan B (2002) A framework for creating hybrid-OSS communities. *Inf Syst J* 12:7–2
- Sterne P, Herring N (2006) SugarCRM – a sweet mix of commercial and open source. In: Linux 16.11.2006, <http://linux.sys-con.com/node/173436>
- Unilog Integrata Unternehmensberatung GmbH (2002) Client Studie der Landeshauptstadt München, München
- Van Antwerp M, Madey G (2008) Advances in the source forge research data archive (SRDA). In: Fourth International Conference on Open Source Systems, IFIP 2.13 (WoPDaSD 2008)

Weiterführende Literatur

- Adelman MA (1955) Concept and statistical measurement of vertical integration. In: Stigler GJ (Hrsg) Business concentration and price policy. Princeton University Press, Princeton, S 279–328
- Afuah A, Tucci CL (2000) Internet business models and strategies: text and cases. McGraw-Hill Higher Education, New York
- Aier S, Riege C, Winter R (2008) Unternehmensarchitektur—Literaturüberblick und Stand der Praxis. Wirtschaftsinformatik 50:292–304
- Aktas N, De Bodt E, Declerck F, Van Oppens H (2007) The PIN anomaly around M&A announcements. J Financ Mark 10:169–191
- Al-Debei MM, Avison D (2010) Developing a unified framework of the business model concept. Eur J Inf Syst 19:359–376
- Alonso G, Casati F, Kuno H, Machiraju V (2004) Web services – concepts, architectures and applications. Springer, Berlin
- Amit R, Zott C (2001) Value creation in E-Business. Strateg Manage J 22:493–520
- Anderson JC, Narus JA (1984) A model of the distributor's perspective of distributor-manufacturer working relationships. J Mark 48(4):62–74
- Anderson JC, Jain DC, Chintagunta PK (1992) Customer value assessment in business markets. J Bus Bus Mark 1:3–29
- Andig M, Hess T (2004) Modularization, individualization and the first-copy-cost-effect – shedding new light on the production and distribution of media content. In: Arbeitspapiere des Instituts für Wirtschaftsinformatik und Neue Medien, LMU München, Nr. 1, München
- Aspray W, Mayadas F, Vardi M (2006) Globalization and offshoring of software. Association for Computing Machinery, New York
- Averesch D (2013) Private Cloud-Speicher: Wege zur sicheren Datenwolke. <http://www.spiegel.de/netzwelt/web/cloud-computing-wege-zum-sicheren-online-speicher-a-938957.html>
- Bacharach SB (1989) Organizational theories: some criteria for evaluation. Acad Manage Rev 14:496–515
- Bain JS (1956) Barriers to new competition. Their character and consequences in manufacturing industries. Harvard University Press, Cambridge
- Barney J (1991) Firm resources and sustained competitive advantage. J Manage 17:99–120

- Becker J, Delfmann P, Knackstedt R (2007) Konfigurierbare Handelsinformationssysteme – Referenzmodelle als Beitrag zur Sicherung des Softwarestandorts Deutschland? Wirtschaftsinformatik 49(Sonderheft):17–27
- Bitkom E, Young AG (2007) Digitale Spiele in Deutschland – Trends und Perspektiven. Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e. V. (Hrsg), Berlin
- Boehmer E, Masumeci J, Poulsen AB (1991) Event-study methodology under conditions of event-induced variance. J Financ Econ 30:253–272
- Boes A, Schwemmle M (2004) Herausforderung Offshoring. Hans-Böckler-Stiftung, Düsseldorf
- Bonaccorsi A, Giannangeli S, Rossi C (2006) Entry strategies under competing standards: hybrid business models in the open source software industry. Manage Sci 52:1085–1098
- Bonakdar A, Weiblen T, Di Valentin C, Zeißner T, Pussep A, Schief M (2013) Transformative influence of business processes on the business model: classifying the state of the practice in the software industry. In: Proceedings of the 46th Hawaii International Conference on System Sciences (HICSS) IEEE, Wailea, USA, S 3920–3929
- Booth D, Haas H, McCabe F, Newcomer E, Champion M, Ferris C, Orchard D (2004) Web services architecture. W3C Working Group Note. <http://www.w3.org/TR/2004/NOTE-ws-arch-20040211>
- Borchers G (2003) The software engineering impacts of cultural factors on multi-cultural software development teams. In: 25th International Conference on Software Engineering, Portland, S 540–545
- Bourantas D (1989) Avoiding dependence on suppliers and distributors. Long Range Plann 22(3):140–149
- Bouwman CHS, Fuller K, Nain AS (2009) Market valuation and acquisition quality: empirical evidence. Rev Financ Stud 22:633–679
- Bowman RG (1983) Understanding and conducting event studies. J Bus Financ Acc 10:561–584
- Brauck M, Müller M, Schulz T (2013) Gnadenlos.com. <http://www.spiegel.de/spiegel/print/d-123826489.html>
- Brislin RW (1970) Back-translation for cross-cultural research. J Cross Cult Psychol 1:185–216
- Broß P (2005) Herausforderungen an Unternehmen und Politik – Chancen und Risiken aus politischer Sicht für Offshoring. Bitkom, München
- Brown SJ, Warner JB (1980) Measuring security price performance. J Financ Econ 8:205–258
- Brown D, Wilson S (2006) 50 best managed global outsourcing vendors. Sourcing Mag. <http://www.sourcingmag.com/content/c060712a.asp>
- Bruner RF (2004) Where M & A pays and where it strays: a survey of the research. J Appl Corp Financ 16:63–77
- Buchta D, Eul M, Schulte-Croonenberg H (2005) Strategisches IT-management. Gabler, Wiesbaden
- Buhl H-U, Heinrich B, Steck W, Winkler V (2004) Konzept zur individualisierten Finanzdienstleistungsberatung für Privatkunden und dessen prototypische Umsetzung. Wirtschaftsinformatik 46:427–438
- Burkard C, Draisbach T, Widjaja T, Buxmann P (2010) Ein Framework zur Erhebung von applikationspezifischen Metadaten in Online-Marktplätzen – Vorstellung und beispielhafte Anwendung im SaaS-Kontext, Darmstädter Arbeitspapier
- Burkhart T, Werth D, Krumeich J, Loos P (2011) Analyzing the business model concept – a comprehensive classification. In: Proceedings of the 32nd International Conference on Information Systems (ICIS), Shanghai, China, S 1–19
- Burkhart T, Wolter S, Schief M, Krumeich J, Di Valentin C, Werth D, Loos P, Vanderhaeghen D (2012) A comprehensive approach towards the structural description of business models. In: Proceedings of the 4th International Conference on Management of Emergent Digital EcoSystems (MEDES) ACM, Addis Ababa, Ethiopia, S 88–102
- Büst (2013) <http://clouduser.de/analysen/die-eigene-community-cloud-prism-break-fuer-profis-oder-die-es-werden-wollen-21681>

- Buxmann P (1996) Standardisierung betrieblicher Informationssysteme. Gabler, Wiesbaden
- Buxmann P, Lehmann S (2009) Preisstrategien von Softwareanbietern. Wirtschaftsinformatik 51:519–529
- Buxmann P, Diefenbach H, Hess T (2008a) Die Softwareindustrie. Ökonomische Prinzipien, Strategien, Perspektiven. Springer, Wiesbaden
- Buxmann P, Gerlach J, Ristl J (2009) Ökonomie von Business-Open-Source-Software am Beispiel ERP. Linux-Magazin 1:51–54
- Buxmann P, Diefenbach H, Hess T (2013) The software industry: economic principles, strategies, perspectives. Springer, Heidelberg
- Capon N, Farley JU, Hoenig S (1990) Determinants of financial performance: a meta-analysis. Manage Sci 36:1143–1159
- Casadesus-Masanell R, Ricart JE (2010) From strategy to business models and onto tactics. Long Range Plan 43:195–215
- Casper J, Di Valentin C, Maier S, Mayer D, Pussep A, Schief M (2013) Vom Geschäftsmodell zum Geschäftsprozess und zurück – Konfiguration, Analyse, Transformation, Controlling. Prax Wirtschaftsinformatik 50:13–22
- Chesbrough H (2010) Business model innovation: opportunities and barriers. Long Range Plan 43:354–363
- Choudhary V (2007) Software as a service: implications for investment in software development. In: Proceedings of the 40th Hawaii International Conference on System Sciences (HICSS 2007), Hawaii, 2007
- Clarke RN (1989) SICs as delineators of economic markets. J Bus 62:17–31
- Cohen J (1960) A coefficient of agreement for nominal scales. Educ Psychol Meas 20:37–46
- Cool K, Schendel D (1988) Performance differences among strategic group members. Strateg Manage J 9:207–223
- Cowan AR (1992) Nonparametric event study tests. Rev Quant Financ Acc 2:343–358
- Cronin B, Catchpole L, Hall D (2004) Outsourcing and offshoring. CESifo Forum 2:17–21
- Cusumano MA (2002) Platform leadership: how Intel, Microsoft, and Cisco drive industry innovation. Harvard Business School Press, Boston
- Dalkey N, Helmer O (1963) An experimental application of the Delphi method to the use of experts. Manage Sci 9:458–467
- Davey H, Allgood B (2002) Offshore development, building relationships across international boundaries: a case study. Inf Strategy 18:13–16
- Dechow PM, Skinner DJ (2000) Earnings management: reconciling the views of accounting academics, practitioners, and regulators. Am Horizons 14:235–250
- De Laat PB (2005) Copyright or copyleft? An analysis of property regimes for software development. Res Policy 34:1511–1532
- Dewenter R (2007) Das Konzept der zweiseitigen Märkte am Beispiel von Zeitungsmonopolen. Medienwirtschaft Sonderh 7–14
- Dibbern J (2004) The sourcing of application software services. Empirical evidence of cultural, industry and functional differences. Physica, Berlin
- Dillman DA (1978) Mail and telephone surveys: the tailored design method. Wiley, New York
- Di Valentin C, Weiblen T, Pussep A, Schief M, Emrich A, Werth D (2012) Measuring business model transformation. In: Proceedings of the 9th European, Mediterranean and Middle Eastern Conference on Information Systems (EMCIS), Munich, Germany, S 1–9
- Domschke W, Drexl A (1996) Logistik: Standorte, 4. Aufl. Oldenbourg, München
- Dostal W, Jeckle M, Melzer I, Zengler B (2005) Service-orientierte Architekturen mit Web Services – Konzepte, Standards, Praxis. Spektrum, Heidelberg
- Ekanayaka Y, Currie WL, Seltsikas P (2003) Evaluating application service providers benchmarking. Int J 10:343–354

- Endres A (2004) Sind Outsourcing und Offshoring die neuen Heilmittel bei Informatik-Problemen? *Informatik Spektrum* 27:546–550
- Engelhardt L (2004) Entrepreneurial models and the software sector. *Competit Change* 8:391–410
- Engelhardt SV (2008) The economic properties of software. Competition and change. Friedrich Schiller University Jena and Max Planck Institute of Economics, Jena, S 1–24
- Erl T (2006) Service-oriented architecture – concepts, technology and design. Prentice Hall, Upper Saddle River
- Fama EF, Fisher L, Jensen MC, Roll R (1969) The adjustments of stock prices to new information. *Int Econ Rev* 10:1–21
- Fettke P (2006) State-of-the-Art des State-of-the-Art Eine Untersuchung der Forschungsmethode „Review“ innerhalb der Wirtschaftsinformatik. *Wirtschaftsinformatik* 48:257–266
- Finch H (2005) Comparison of distance measures in cluster analysis with dichotomous data. *J Data Sci* 3:85–100
- Frankenberger K, Weiblen T, Csik M, Gassmann O (2013) The 4I Framework of business model innovation: a structured view on process phases and challenges. *Int J Prod Dev* 18:249–273
- Franz A (2003) Management von business webs. Gabler, Wiesbaden
- Gay J (2013) Interview with Frank Karlitschek of ownCloud. <http://www.fsf.org/blogs/licensing/interview-with-frank-karlitschek-of-owncloud>
- Gold T (2005) Outsourcing software development offshore. Auerbach, Boca Raton
- Gordijn J, Akkermans H (2001) Designing and evaluating e-business models. *IEEE Intell Syst* 16:11–17
- Görisch F (2009) Portalbasierte Softwaredistribution für mobile Endgeräte. Diplomarbeit am Fachgebiet Information Systems, TU Darmstadt
- Grau C, Hess T (2007) Kostendegression in der digitalen Medienproduktion: Klassischer First-Copy-Cost-Effekt oder doch mehr? *MedienWirtschaft: Zeitschrift für Medienmanagement und Kommunikationsökonomie*, Sonderheft 1: Theoriebezüge von Medienökonomie und Medienmanagement, S 26–37
- Grover V, Saeed KA (2004) Strategic orientation and performance of Internet-based businesses. *Inf Syst J* 14:23–42
- Hamel G (2002) Leading the revolution: how to thrive in turbulent times by making innovation a way of life. Harvard Business Press, Boston
- Harford J (2005) What drives merger waves? *J Financ Econ* 77:529–560
- Hedman J, Kalling T (2003) The business model concept: theoretical underpinnings and empirical illustrations. *Eur J Inf Syst* 12:49–59
- Heinrich B (2002) Methode zur wertorientierten Analyse und Gestaltung der Kundeninteraktion – Zur Rolle des Service Integrators im Privatkundengeschäft von Kreditinstituten. Universität St. Gallen, St. Gallen
- Helweg-Larsen M, Shepperd J (2001) Do moderators of the optimistic bias affect personal or target risk estimates? *Personal Soc Psychol Rev* 5:74–95
- Henn J, Khan A (2007) Serviceorientierung – Mehr als nur IT-Architekturen. In: Fink D, Gries A, Lünenbeck T (Hrsg) Consulting-Kompendium. FAZ Frankfurt, S 208–217
- Herrmann W (2005) Offshore-Anbieter plagen Personalsorgen. *Computerwoche* 39:14
- Hess T, Hippe A (Hrsg) (2006) Industrialisierung des Controllings. Zeitschrift für Controlling und Management. Sonderheft 2, Wiesbaden
- Hess T, von Walter B (2006) Toward content intermediation: shedding new light on the media sector. *Int J Media Manage* 8(1):2–8
- Hess T, Loos P, Buxmann P, Erek K, Frank U, Gallmann J, Gersch M, Zarnekow R, Zencke P (2012a) ICT providers: a relevant topic for business and information systems engineering? *Bus Inf Syst Eng* 4:367–373

- Hess T, Veit D, Steininger D, Kundisch D, John T, Spann M, Dechant A, Benlian A, Stefi A, Lübcke J (2012b) Geschäftsmodelle als Thema der Wirtschaftsinformatik. S 1–23
- Heuser L (2006) Business Webs – eine zentrale Vision von SAP Research. <http://www.gi-ev.de/fileadmin/redaktion/Presse/Statement-Heuser-INFORMATIK2006.pdf>
- Hevner AR, March ST, Park J, Ram S (2004) Design science in information systems research. MIS Q 28:75–105
- Hildenbrand T, Rothlauf F, Heinzl A (2007) Ansätze zur kollaborativen Softwareerstellung. Wirtschaftsinformatik 49(Sonderheft):72–80
- Hoch DJ (2000) Secrets of software success: management insights from 100 software firms around the world. Harvard Business Press, Boston
- Hodel M, Berger A, Risi P (2006) Outsourcing realisieren. Vieweg, Wiesbaden
- Hoffer CW (1975) Toward a contingency theory of business strategy. Acad Manage J 18:784–810
- Holweg M, Pil FK (2006) Evolving from value chain to value grid. MIT Sloan Manage Rev 47:72–80
- Houthoofd N, Heene A (1997) Strategic groups as subsets of strategic scope groups in the Belgian brewing industry. Strategic Manage J 18:653–666
- Hunt MS (1972) Competition in the major home appliance industry. Harvard University Press, Boston
- Hutzschenreuter T, Gröne F (2009) Changing vertical integration strategies under pressure from foreign competition: the case of US and German multinationals. J Manage Stud 46:269–307
- Iansiti M, Clark KB (1994) Integration and dynamic capability: evidence from product development in automobiles and mainframe computers. Ind Corporate Change 3:557–605
- Illgner H (2008) SaaS für den Mittelstand: SAP Business ByDesign, die neue On-Demand-Lösung der SAP. In: Hess T (Hrsg) Software as a service: Strategische Perspektiven und praktische Bedeutung. Springer, Berlin, S 15–24
- Jacobsen R (1988) The persistence of abnormal returns. Strateg Manage J 9:415–430
- Johnson MW, Christensen CM, Kagermann H (2008) Reinventing your business model. Harv Bus Rev 86:57–68
- Jope F, Schiereck D, Zeidler F (2010) Value generation of mergers and acquisitions in the technology, media and telecommunications industry. J Telecommun Manage 2:369–386
- Kagermann H, Österle H (2006) Geschäftsmodelle 2010 – Wie CEOs Unternehmen transformieren. Frankfurter Allgemeine Buch, Frankfurt a. M.
- Kahle KM, Walkling RA (1996) The impact of industry classifications on financial research. J Finance Quant Anal 31:309–335
- Käkölä T (2003) Software business models and contexts for software innovation: key areas for software business research. In: Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS) IEEE, Waikoloa, USA, S 1–8
- Kauffman RJ, Wang B (2008) Tuning into the digital channel: evaluating business model characteristics for Internet firm survival. Inf Technol Manage 9:215–232
- Ketchen DJ, Combs JG, Russell CJ, Shook C, Dean MA, Lohrke FA, Naumann SE, Haptonstahl DE, Baker R, Beckstein BA, Handler C, Honig H, Lamoureux S (1997) Organizational configurations and performance: a meta-analysis. Acad Manage J 40:223–240
- Ketchen DJ, Snow CC, Hoover VL (2004) Research on competitive dynamics: recent accomplishments and future challenges. J Manage 30:779–804
- Kieser A (1996) Moden und Mythen des Organisierens. Betriebswirtschaft 56:21–39
- Kilian-Kehr R, Terzidis O, Voelz D (2007) Industrialisation of the software sector. Wirtschaftsinformatik 49(Sonderheft):62–71
- Kittlaus HB, Clough P (2009) Software product management and pricing. Key success factors for software organizations. Springer, Berlin

- Klosterberg M (2010) Die Bewertung von Softwareunternehmen. In: Drukarczyk J, Ernst D (Hrsg) Branchenorientierte Unternehmensbewertung, 3. Aufl. Franz Vahlen, München, S 255–273
- Knolmayer GF (2007) Compliance-Nachweise bei Outsourcing von IT-Aufgaben. Wirtschaftsinformatik 49(Sonderheft):98–106
- Koch S (2004) Das Open-Source-Entwicklungsmodell: Grundprinzipien, Fragen und Erfahrungen. HMD-Prax Wirtschaftsinformatik 231:55–62
- König W, Beimborn D, Franke J, Weitzel T (2005) Sourcing von Finanzprozessen – Ein Modell zur simultanen Bewertung von Economies of Scale und Scope. In: Internationale Tagung Wirtschaftsinformatik (WI 2005), Bamberg
- Kontio J, Jokinen J-P, Mäkelä MM, Leino V (2005) Current practices and research opportunities in software business models. In: Proceedings of the 7th Economics-Driven Software Engineering Research (EDSER) ACM, St. Louis, S 1–4
- Krafcig D, Banke K, Slama D (2006) Enterprise SOA. Best Practices für Serviceorientierte Architekturen – Einführung, Umsetzung, Praxis. Prentice Hall PTR, Upper Saddle River
- Kretschmer T (2004) Upgrading and Niche usage of PC operating systems. Int J Ind Org 22:1155–1182
- Krumeich J, Burkhardt T, Werth D, Loos P (2012) Towards a component-based description of business models: a state-of-the-art analysis. In: Proceedings of the 18th American Conference on Information Systems (AMCIS), Seattle, USA, S 1–12
- Kundisch D, John T, Honnacker J, Meier C (2012) Approaches for business model representation: an overview. In: Proceedings of the Multkonferenz Wirtschaftsinformatik (MKWI), Braunschweig, Germany, S 1821–1832
- L W (2013) NSA-Skandal kostet USA bis zu 35 Milliarden Dollar. <http://www.welt.de/wirtschaft/article122307814/NSA-Skandal-kostet-USA-bis-zu-35-Milliarden-Dollar.html>
- Laabs K (2004) Offshore outsourcing und co-sourcing. In: Gründer T (Hrsg) IT Outsourcing in der Praxis. Erich Schmidt, Berlin, S 116–129
- Lacity M, Hirschheim R, Willcocks L (1994) Realizing outsourcing expectations. Inf Syst Manage 11:7–18
- Lambert SC, Davidson RA (2012) Applications of the business model in studies of enterprise success, innovation and classification: an analysis of empirical research from 1996 to 2010. Eur Manage J 1–14
- Landeta J (2006) Current validity of the Delphi method in social sciences. Technol Forecast Soc Change 73:467–482
- Lang JC, Widjaja T, Buxmann P, Domschke W, Hess T (2008) Optimizing the supplier selection and service portfolio of a SOA service integrator. In: Proceedings of the 41st Hawaii International Conference on System Sciences (HICSS), Hawaii
- Lee H, Smith KG, Grimm CM (2003) The effect of new product radicality and scope on the extent and speed of innovation diffusion. J Manage 29:753–768
- Léger P-M, Yang S (2005) Network effects and the creation of shareholders' wealth in the context of software firm mergers and acquisitions. In: 13th European Conference on Information Systems, Regensburg, Germany, S 1–13
- Lehmann S, Buxmann P (2009) Pricing strategies of software vendors. Bus Inf Syst Eng 1:452–462
- Leidner DE, Lo J, Preston D (2011) An empirical investigation of the relationship of is strategy with firm performance. J Strateg Inf Syst 20:419–437
- Lewin R (2001) Whitepaper on Offshore software development in Russia. American Chamber of Commerce in Russia, Moskau. http://russiansoftwaredev.esolutions.ru/en/amcham_whitepaper.doc
- Likert R (1932) A technique for the measurement of attitudes. Arch Psychol 22:136–165
- Lindner M (2013) Servermarkt: x86 und Linux weiter auf dem Vormarsch <http://www.pro-linux.de/news/1/20562/servermarkt-x86-und-linux-weiter-auf-dem-vormarsch.html>

- Lovelock J-D (2013) Forecast Alert: IT spending, Worldwide, 4Q12 Update. <http://www.gartner.com/id=2291618>. Zugegriffen: 23. März 2013
- MacCormack A, Verganti R, Iansiti M (2001) Developing products on „Internet Time“: the anatomy of a flexible development process. *Manage Sci* 47:133–150
- Mackie K (2013) Google apps making inroads against Microsoft Office, Gartner Says. <http://rpp-mag.com/articles/2013/04/23/google-apps-vs-microsoft-office.aspx>
- Magretta J (2002) Why business models matter. *Harv Bus Rev* 80:3–8
- Malone TW, Weill P, Lai RK, D’Urso VT, Herman G, Apel TG, Woerner SL (2006) Do some business models perform better than others? MIT Sloan Research Paper, Cambridge, S 1–34
- Mason KJ, Leek S (2008) Learning to build a supply network: an exploration of dynamic business models. *J Manage Stud* 45:774–799
- Mathrani A, Viehland D, Parsons D (2005) Dynamics of Offshore software development success: the outsourcingers’ perspective. In: Proceedings of International Conference of Knowledge Management in Asia Pacific. Victoria University, Wellington, S 28–29
- McCormick T (2010) Understanding costs using the value chain – a Ryanair example. *Accountancy Ireland* 42:28–30
- McGahan AM, Porter ME (1997) How much does industry matter, really? *Strateg Manage J* 18(Summer Special Issue):15–30
- McGee J, Thomas H (1986) Strategic groups: theory, research and taxonomy. *Strateg Manage J* 7:141–160
- McNamara G, Deephouse DL, Luce RA (2003) Competitive positioning within and across a strategic group structure: the performance of core, secondary, and solitary firms. *Strateg Manage J* 24:161–181
- Mehra A (1996) Resource and market based determinants of performance in the U.S. banking industry. *Strat Manage J* 17:307–322
- Mergerstat Free Reports (2009) Mergerstats free reports: industry rankings for year 2009. https://www.mergerstat.com/newsite/free_report.asp
- Messerschmitt DG, Szyperski C (2005) Software ecosystem: understanding an indispensable technology and industry. MIT Press Books, Cambridge
- Meyer T (2006) Nearshoring nach Mittel- und Osteuropa. Deutsche Bank Research, Frankfurt a. M.
- Michalski RS, Carbonell JG, Mitchell TM (1983) Machine learning: an artificial intelligence approach. Morgan Kaufmann Publishers, Los Altos
- Miller D (1981) Toward a new contingency approach: the search for organizational gestalts. *J Manage Stud* 18:1–26
- Moore GE (1965) Cramming more components onto integrated circuits. *Electronics* 38:114–117
- Nalebuff BJ, Brandenburger AM (2008) Coopetition: kooperativ konkurrieren – Mit der Spieltheorie zum Geschäftserfolg. Rieck, Eschborn
- Nelson RR (1991) Why do firms differ, and how does it matter? *Strateg Manage J* 12:61–74
- Nutt P (1997) Better decision-making: a field study. *Bus Strat Rev* 8:45–52
- Okoli C, Pawlowski SD (2004) The Delphi method as a research tool: an example, design considerations and applications. *Inf Manage* 42:15–29
- Osterwalder A (2004) The business model ontology: a proposition in a design science approach. Universite de Lausanne, Lausanne
- oV (2006) Temporary workers. U.S. department of state. http://travel.state.gov/visa/temp/types/types_1271.html
- oV (2011) A look at Dropbox’s business model. <http://rocksaucestudios.com/tapsauce/post/dropboxbusinessmodel2011/>
- oV (2011a) Dropbox: Angebot von Steve Jobs abgelehnt. <http://futurezone.at/b2b/dropbox-angebot-von-steve-jobs-abgelehnt/24.572.177>

- Pateli AG, Giaglis GM (2004) A research framework for analysing eBusiness models. *Eur J Inf Syst* 13:302–314
- Penrose E (1959) The theory of the growth of the firm. Oxford University Press, Oxford
- Peteraf MA (1993) The cornerstones of competitive advantage: a resource-based view. *Strateg Manage J* 14:179–191
- Petrasch R, Meimberg O (2006) Model driven architectures. dpunkt, Heidelberg
- Petrovic O, Kittl C, Teksten R (2001) Developing business models for Ebusiness. In: International Electronic Commerce Conference (BLED), Vienna, Austria, S 1–6
- Pfeffer J, Sutton RI (2006) Evidence-based management. *Harv Bus Rev* 84:12
- Picot A (1982) Transaktionskostenansatz in der Betriebswirtschaftslehre: Stand der Diskussion und Aussagewert. *Betriebswirtschaft* 42:267–284
- Pigou AC (1929) The economics of welfare, 3. Aufl. Macmillian, London
- Pohl A, Onken BR (2003) Outsourcing und Offshoring mit indischen IT-Unternehmen. Deloitte & Touche, München
- Popp KM, Meyer R (2010) Profit from software ecosystems: business models, ecosystems and partnerships in the software industry. Books on Demand, Norderstedt
- Porter ME (1979) The structure within industries and companies' performance. *Rev Econ Stat* 61:214–227
- Porter ME (1980) Competitive strategy: techniques for analyzing industry and competitors. Free Press, New York
- Porter ME (2001) Strategy and the Internet. *Harv Bus Rev* 79:62–79
- Powers S (2013) Readers' Choice Awards 2013. <http://www.linuxjournal.com/rc2013?page=28>
- Prescott JE, Kohli AK, Venkatraman N (1986) The market share-profitability relationship: an empirical assessment of major assertions and contradictions. *Strateg Manage J* 7:377–394
- Pussep A, Schief M, Widjaja T (2012b) The software value chain: methods for construction and their application. In: Proceedings of the 20th European Conference on Information Systems (ECIS), Barcelona, Spain, S 1–12
- R Development Core Team (2012) R: a language and environment for statistical computing
- Rajala R (2009) Antecedents to and performance effects of software firms' business models. In: (Hrsg) Determinants of business model performance in software firms. Helsinki School of Economics, Helsinki, S 166–194
- Rajala R, Rossi M, Tuunainen VK (2003) A framework for analyzing software business models. In: Proceedings of the 11th European Conference on Information Systems (ECIS), Naples, S 1–15
- Rajala R, Westerlund M, Möller K (2012) Strategic flexibility in open innovation—designing business models for open source software. *Eur J Market* 46:1368–1388
- Rajkumar TM, Mani RVS (2001) Offshore software development. *Inf Syst Manage* 18:63–74
- Rao MT (2004) Key issues for global IT sourcing: country and individual factors. *Inf Syst Manage* 21:16–21
- Rappa MA (2004) The utility business model and the future of computing services. *IBM Syst J* 43:32–42
- Redis J (2009) The impact of business model characteristics on IT firms' performance. *Int J Bus* 14:291–307
- Rönkkö M, Valtakoski A (2009) Business models of software firms. In: Proceedings of the 42nd Hawaii International Conference on System Sciences (HICSS) (CD-ROM), Waikoloa, USA, S 1–10
- Rothaermel FT, Hitt MA, Jobe LA (2006) Balancing vertical integration and strategic outsourcing: effects on product portfolio, product success, and firm performance. *Strateg Manage J* 27:1033–1056

- Royce WW (1970) Managing the development of large software systems. In: Proceedings of the Western Electronic Show and Convention Conference (WESCON) IEEE Los Angeles, USA, S 1–9
- Ruiz BE, Claus R (2005) Offshoring in der deutschen IT Branche. Informatik Spektrum 28:34–39
- Rumelt RP (1991) How much does industry matter? Strateg Manage J 12:167–185
- Sahay S, Nicholson B, Krishna S (2003) Global IT-outsourcing. Cambridge University Press, Cambridge
- SAP (2013) Annual Report 2012, S 1–304
- Schaaf J (2004) Offshoring: Globalisierungswelle erfasst Dienstleistungen. Deutsche Bank Research, Frankfurt a. M.
- Schaaf J, Weber M (2005) Offshoring-Report 2005. Ready for take-off. In: Deutsche Bank Research, Nr. 52 vom 14. Juni 2005. http://www.expeditiondeutschland.de/PROD/DBR_INTERNET_DE-PROD/PROD000000000188321.pdf
- Schaaf J, Allweyer T, Besthorn T (2004) IT outsourcing: between starvation diet and nouvelle cuisine. Deutsche Bank Research, Frankfurt a. M.
- Scheer A-W (2002) ARIS – Vom Geschäftsprozess zum Anwendungssystem, 4. Aufl. Springer, Berlin
- Schelp J, Winter R (Hrsg) (2006) Integrationsmanagement. Planung, Bewertung und Steuerung von Applikationslandschaften. Springer, Berlin
- Schief M, Pussep A (2013) Software business model determinants of performance – insights from Germany. In: Proceedings of the 21st European Conference on Information Systems (ECIS), Utrecht, Netherlands, S 1–12
- Schief M, Bonakdar A, Weiblen T (2012a) Transforming software business models into business processes. In: Proceedings of the 14th International Conference on Enterprise Information Systems (ICEIS), Wroclaw, Poland, S 167–172
- Schief M, Pussep A, Buxmann P (2012b) Performance of business models: empirical insights from the software industry. In: Proceedings of the 16th Pacific Conference on Information Systems (PACIS), Ho Chi Minh City, Vietnam, S 1–14
- Schief M, Pussep A, Buxmann P (2012c) Performance of business models: empirical insights from the software industry. In: Proceedings (Poster Version) of the European Conference on Information Systems (ECIS), Barcelona, Spain, 1–4
- Schief M, Buxmann P, Schiereck D (2013a) Mergers & acquisitions in the software industry – research results in the area of success drivers. Bus Inf Syst Eng 5(6):Forthcoming
- Schief M, Pussep A, Buxmann P (2013b) The impact of software business model characteristics on firm performance. In: 4th International Conference on Software Business Lecture Notes in Information Systems, Potsdam, S 1–12
- Schmalen K (2012) IDC Predictions 2013: Mobile- und Cloud-Deployments Heizen 2013 den Kampf um die Dritte Plattform an. http://www.idc.de/press/presse_idc-prediction2013_mobile_cloud_deployments.jsp. Zugegriffen: 6. April 2013
- Schmalensee R (1985) Do markets differ much? Am Econ Rev 75:341–351
- Schmidt RC (1997) Managing Delphi surveys using nonparametric statistical techniques. Decis Sci 28:763–774
- Schmidt J (2002) Nicht Trustworthy – IE gefährdet Rechner und Netz. <http://www.heise.de/ct/02/25/100/>
- Schmidt DC (2006) Model-driven-engineering. IEEE Comput 39:25–31
- Schmidt B, Schief M (2010) Towards agile business processes based on the Internet of things. In: Dangelmaier W, Blecken A, Delius R, Klöpfer S (Hrsg) Advanced manufacturing and sustainable logistics. Lecture Notes in Business Information Processing, S 257–262
- Schmitz L (2013) Jena baut sich seine OwnCloud. <http://www.computerwoche.de/a/jena-baut-sich-seine-owncloud,2543027>

- Schott E (2006) Kostensenkungspotenziale im IT-Outsourcing. In: Clement R (Hrsg) *IT-Controlling und Beteiligungscontrolling in Forschung und Praxis. Tagungsband zur dritten Fachtagung IT-Controlling*. St-Augustin, S 28–35
- Schultz S (2013) Neuer IT-Hype: Dropbox soll Börsengang vorbereiten. <http://www.spiegel.de/wirtschaft/unternehmen/dropbox-soll-boersengang-vorbereiten-a-883688.html>
- Schumpeter JA (1934) *The theory of economic development: an inquiry into profits, capital, credit, interest, and the business cycle*. Harvard University Press, Cambridge
- SDC (2012) Thomson one banker deals database. http://thomsonreuters.com/products_services/financial/financial_products/deal_making/investment_banking/thomsononecom_ib/. Zugegriffen: 16. April 2012
- SEC (2013) U.S. Securities and exchange commission: form 10-K. <http://www.sec.gov/>. Zugegriffen: 6. April 2013
- Seddon PB, Lewis GP, Freeman P, Shanks G (2004) The case for viewing business models as abstractions of strategy. *Commun Assoc Inf Syst* 13:427–442
- Shah S (2006) Motivation, governance, and the viability of hybrid forms in open source software development. *Manage Sci* 52:1000–1014
- Short JC, Ketchen DJ, Palmer TB, Hult DTM (2007) Firm, strategic group, and industry influences on performance. *Strateg Manage J* 28:147–167
- Skiera B (2000) Wie teuer sollen die Produkte sein? – Preispolitik, Ecommerce: Einstieg, Strategie und Umsetzung im Unternehmen. In: Albers S, Clement M, Peters K, Skiera B (Hrsg) *eCommerce: Einstieg, Strategie und Umsetzung in Unternehmen*. Frankfurter Allgemeine Buch, Frankfurt a. M., S 95–108
- Sokolovsky Z, Löschenkohl S (Hrsg) (2005) *Handbuch Industrialisierung der Finanzwirtschaft – Strategien, Management und Methoden für die Bank der Zukunft*. Gabler, Wiesbaden
- Stadtler H (2005) Supply chain management and advanced planning – basics, overview, challenges. *Eur J Oper Res* 163:575–588
- Staten J (2012) Cloud predictions: we'll finally get real about cloud. http://blogs.forrester.com/james_staten/12-12-03-2013_cloud_predictions_well_finally_get_real_about_cloud
- Steininger DM, Huntgeburth JC, Veit DJ (2011) Conceptualizing business models for competitive advantage research by integrating the resource and market-based views. In: Proceedings of the 17th American Conference on Information Systems (AMCIS), Detroit, USA, S 1–12
- Stelzer D (2004) Produktion Digitaler Güter. In: Braßler A, Corsten H, Blecker T, Schneider H (Hrsg) *Entwicklungen im Produktionsmanagement*. Vahlen, München, S 233–250
- Suerie C (2005) Time continuity in discrete time models – new approaches for production planning in process industries. Springer, Berlin
- Suermann JC (2006) Bilanzierung von Software nach HGB, US-GAAP und IFRS – Integrative Analyse der Regelungen zu Ansatz, Bewertung und Umsatzrealisation von Software aus Hersteller- und Anwendersicht. Diss. Universität Würzburg. <http://www.opus-bayern.de/uni-wuerzburg/volltexte/2006/1933>
- Tanriverdi H, Lee C-H (2008) Within-industry diversification and firm performance in the presence of network externalities: evidence from the software industry. *Acad Manage J* 51:381–397
- Thibaut JW, Kelley HH (1959) *The social psychology of groups*. Wiley, New York
- Thome R, Böhn M, Haqn A (2004) Model driven architecture. *WISU* 33:348–357
- Timmers P (1998) Business models for electronic markets. *Electron Mark* 8:3–8
- Tuch C, O'Sullivan N (2007) The impact of acquisitions on firm performance: a review of the evidence. *Int J Manage Rev* 9:141–170
- Tucker IB, Wilder RP (1977) Trends in vertical integration in the U.S. manufacturing sector. *J Ind Econ* 26:81–94

- Valtakoski A, Rönkkö M (2010) Diversity of business models in software industry. In: Proceedings of the 1st International Conference on Software Business (ICSOB) Lecture Notes in Business Information Processing, Brussels, Belgium, S 1–12
- Van der Linden F, Bosch J, Kamsties E, Känsälä K, Obbink H (2004) Software product family evaluation. In: Proceedings of Software Product Lines, Third International Conference, SPLC 2004, Boston, MA, S 110–129
- Van Putten B-J, Schief M (2012a) The relation between dynamic business models and business cases. In: Proceedings of the 5th European Conference on Information Management and Evaluation (ECIME), Como, Italy, S 562–569
- Van Putten B-J, Schief M (2012b) The relation between dynamic business models and business cases. *Electron J Inf Syst Evaluat* 15:138–148
- Venkatraman N (1989) Strategic orientation of business enterprises: the construct, dimensionality, and measurement. *Manage Sci* 35:942–962
- Venkatraman N, Lee C-H, Iyer B (2008) Interconnnect to win: the joint effects of business strategy and network positions on the performance of software firms. *Adv Strateg Manage* 25:391–424
- Vom Brocke J, Simons A, Niehaves B, Riemer K, Plattfaut R, Cleven A (2009) Reconstructing the giant: on the importance of rigour in documenting the literature search process. In: 17th European Conference on Information Systems, Verona, S 1–13
- Vossen R (2013) Dropbox bereitet Börsengang vor. <http://www.basicthinking.de/blog/2013/02/15/dropbox-bereitet-borsengang-vor/>
- Wahrenburg M, König W, Beimborn D, Franke J, Gellrich T, Hackethal A, Holzhäuser M, Schwarze F, Weitzel T (2005) Kreditprozess-Management – Status Quo und Zukunft des Kreditprozesses bei Deutschlands 500 größten Kreditinstituten. Books on Demand, Norderstedt
- Webster J, Watson RT (2002) Analyzing the past to prepare for the future: writing a literature review. *MIS Q* 26:13–23
- Weinstein N, Klein W (1996) Unrealistic optimism: present and future. *J Soc Clin Psychol* 15:1–8
- Wendt O, von Westarp F, König W (2000) Pricing in network effect markets. In: 8th European Conference on Information Systems (ECIS2000), Wien
- Wernerfelt B (1984) A resource-based view of the firm. *Strateg Manage J* 5:171–180
- Wessa P (2009) A framework for statistical software development, maintenance, and publishing within an open-access business model. *Comput Stat* 24:183–193
- Williamson OE (1981) The economics of organization: the transaction cost approach. *Am J Sociol* 87:548–577
- Williamson OE (1991) Comparative economic organization: the analysis of discrete structural alternatives. *Adm Sci Q* 36:269–296
- Wirtz BW (2011) Business model management: design, instruments, success factors. Gabler, Wiesbaden
- Wissenbach I (2014) SAP setzt auf die Cloud. <http://www.dw.de/sap-setzt-auf-die-cloud/a-17376391>
- Witten IH, Frank E (2005) Data mining: practical machine learning tools and techniques. Morgan Kaufmann, San Francisco
- Zacharias R (2007) Produktlinien: Der nächste Schritt in Richtung Software-Industrialisierung. *Java Mag* 3:69–82
- Zencke P (2007) Einsatz von Standardsoftware im Mittelstand. *Wirtschaftsinformatik* 49(Sonderheft):122–124
- Zott C, Amit R (2007) Business model design and the performance of entrepreneurial firms. *Org Sci* 18:181–199
- Zott C, Amit R (2008) The fit between productmarket strategy and businessmodel: implications for firm performance. *Strateg Manage J* 29:1–26
- Zott C, Amit R, Massa L (2011) The business model: recent developments and future research. *J Manage* 37:1019–1042

Sachverzeichnis

A

- Accenture 170, 186
- Adobe 116, 122
- Application Service Providing 231
- Arbeitsteilung 44, 45, 47, 144, 146, 148, 150–152, 187, 210, 211
- Architektur, serviceorientierte 7
- Automatisierung 210, 211

B

- Backsourcing 195
- Basisnutzen 22–26, 36, 38, 266
- Behavioral Pricing 124
- Bemessungsgrundlage 95, 108, 111, 112, 231, 248–254
- Beratungsdienstleistung 16, 113
- Best-of-Breed 39, 40, 44
- Branchenplattform 207, 213, 216
- Broker 68, 70
- Browserkrieg 27
- Build-Operate-Transfer-Modell 173
- Bündelungsstrategie 28, 116, 118–121

C

- Cloud Computing 221–224, 228–231, 245
- CMMI 129
- Concurrent User 250, 253
- Conjoint-Analyse 252
- Co-opetition 66
- Copyleft 259
- CRM-Software 77
- Customer-Relationship-Management 269
- Customizing 7, 122, 191, 192, 235, 236

D

- Dienstleistung 8, 9, 17, 20, 21, 32, 45, 50, 61, 75, 85, 170, 173, 175, 211, 267
- Dienstvertrag 54
- Distributor 267

E

- Early Adopter 27
- Enterprise Resource Planning 4, 12, 21, 29, 36, 67, 76, 77, 87, 89, 145, 146, 148, 149, 235, 238, 239, 248, 254, 269, 270, 272–276
- Entlohnung 51, 53–56, 265
- Erfahrungswert digitaler Güter 14
- Erlösmodell 16, 231

F

- Farshoring 173, 176, 186, 199, 200
- First Copy 19, 108
- Follow-the-Free-Strategie 121, 122
- Follow-the-Sun-Prinzip 174, 176, 203, 204
- Free Software 258–260
- Freeware 258
- Fremdleistung 188, 190–192, 194, 196–198
- Function-Point-Methode 123, 124

G

- Geschäftsmodell 4, 10, 16, 17, 105, 141, 152, 153, 155, 158–160, 216, 223, 225, 232, 267, 269–272
- Global Delivery Model 186
- GNU 258–261, 263
- Google 13, 71, 221, 247
- GPL-Lizenz 259, 261, 262, 265, 268, 272
- Güter, digitale 17, 108

H

- Hidden Action 52, 53
 Hidden Characteristics 52, 53
 Hidden Intention 53

I

- IBM 4, 30, 32, 71, 269
 iBS Banking Solution 65
 Increasing Returns 22, 26
 Industrialisierung 50, 210–212
 Informationskosten 36–38, 40–43
 Inkompatibilität 35, 77
 Intermediationstheorie 50

J

- Joint Venture 65, 172, 173, 204

K

- Kennzahl 55, 93–95, 98
 Kennzahlensystem 85, 92, 93
 Key Account Manager 91
 Key Performance Indicator ► Kennzahl 95
 Kommunikationskosten 37
 Kompatibilität 46
 Komplementäre 213, 215, 217, 218
 Konsolenkrieg 32
 Kooperationsbeziehung 145
 Kooperationsstrategie 49, 61
 Koordinationsform 47, 49, 144, 147, 150

L

- Lessor 68
 LGPL-Lizenz 259, 261, 262
 Linux 259, 260, 263, 266–269
 Lizenz 17, 32, 89, 90, 100, 101, 108, 110, 111, 122, 250, 257, 259–262, 265, 270, 272, 273
 Lizenzerlöse 16, 17
 Lizenzkosten 29, 36, 233, 235, 266, 267
 Lizenzmodell 108, 155, 158, 272
 Local Attractiveness Index 186
 Lock-in-Effekt 29, 110, 115, 121, 122, 236, 276

M

- Maintainer 263, 264
 Make-or-Buy-Entscheidung 44

- Microsoft 4–6, 22–24, 27–29, 31, 32, 67, 71, 77, 113, 117, 147, 221, 247, 262, 265, 266, 269

- Mittelstand 88, 89
 Move-to-the-Market 49
 Move-to-the-Middle 49

N

- Nash-Gleichgewicht 42
 Nearshoring 173, 186
 Netzeffekte 19–25, 27, 29–34, 43, 61, 72, 75, 77, 81, 110, 114, 116, 121, 122, 267
 Netzeffektfaktor 23, 29, 122
 Netzeffektnutzen 22–24, 26, 33, 34, 36
 Niedrigpreisstrategie 30, 108, 121, 122
 Öffnung 48, 213, 215, 216

O

- Offshore 48, 171, 173–176, 186, 187, 199–203
 Offshoring 170–175, 177, 186, 200, 201, 203
 Onshoring 170, 186
 Open Source Initiative 259, 261
 Open Source Software 22, 62, 257, 266
 OpenDocument Format 71
 Oracle 30, 71, 73, 76, 77, 269, 270
 Outsourcing 9, 10, 48, 53, 169, 170, 173–175, 188, 191, 194, 195, 198, 201, 221, 223, 233, 236

P

- Parametrisierung ► Customizing 191
 Partnerschaft 64, 65, 67–71
 Penetration Pricing 121, 122
 Person-Job-Fit 130
 Pfadabhängigkeit 25, 26
 Pflichtenheft 11
 Pinguineffekt 21, 24, 25, 29, 276
 Plattformsponsor 216
 Plattformstrategie 21, 30
 Point of Marginal Cheapness 253
 Point of Marginal Expensiveness 253
 Positive Feedbacks 22, 27
 Preisbildung 109, 110
 Preisbündelung 108, 115–117, 119, 121
 Preisdifferenzierung 112–116, 122, 248–250, 254, 270–272
 Preisermittlung 109, 110

Preismodell 108–111, 113, 115, 247, 248, 250, 252, 253
Preisstrategie 17, 23, 61, 107, 113, 121
Price Sensitivity Meter 252
Principal-Agent-Theorie 19, 51–53, 95
Produktplattform 207–209

R

Raubkopie 20
Rendite 79, 80, 107
Reservationspreis 113, 116, 118–122

S

SaaS 89, 110–112, 221, 231–241, 247–250, 252–254
Safe Passage Program 29
Sales Funnel 100–102
Salesforce.com 207, 232, 247
SAP 3, 4, 6, 7, 9, 13, 22, 29, 30, 65–67, 70, 76, 88, 90, 91, 169, 186, 187, 269, 276
Skaleneffekte 62, 175, 221, 241
Skimming Pricing 121, 122
Skimming-Strategie ► Skimming Pricing 122
Software 5–9, 11, 22, 23, 36, 37, 45, 51, 87, 88, 122, 123, 188, 190–194, 198, 221, 231, 236, 266
Software AG 73, 172, 173, 204
Softwareanbieter 5, 8
Software-Auswahl 10, 12–15
Softwareentwicklung 50, 51, 55, 57, 65, 125–130, 147, 169–171, 173, 175, 202, 211, 236, 262, 263, 273
Softwarelizenz 20, 90, 91, 113, 115, 259, 262
Spezialisierung 87, 210, 211
Spieleindustrie, digitale 30, 31
Standardisierung 34, 37, 41, 42, 210–212
Standardisierungsproblem 35, 38, 41
Standortentscheidung 186
Standortwahl 170, 185, 186
Startup-Problem 25, 29, 121

SuSe 267, 268
Switching Costs 22, 29, 235, 276

T

Transaktion 45–47, 50, 51, 79–81, 253, 254
Transaktionskosten 19, 44–46, 48, 50, 73, 87
Transaktionskostentheorie 44–50, 171

V

Value Net 62
Vertrieb 86–88
Vertriebscontrolling 85, 92, 94, 95
Vertriebsorganisation 85
Vertriebspipeline 102
Vertriebsstrategie 85
Vertriebssystem 85, 89
Vertriebsweg 85, 86, 88, 89

W

Web Service 117, 222, 224
Werkvertrag 53
Wertschöpfungskette 21, 31, 44, 61, 73, 85
Wertschöpfungsstruktur 31, 33, 146, 148
Wertschöpfungsstufe 73, 144, 145, 147–152
Winner-takes-it-all-Markt 4, 21, 27, 32, 75

X

XML 21, 71

Z

Zahlungsbereitschaft 108, 113–116, 118–120, 122, 124, 251–254
Zahlungsstrom 110, 248, 249
Zeitverschiebung ►
Follow-the-Sun-Prinzip 176
Zielsystem 11