1 Business Software – das Nervensystem moderner Unternehmen

Ralf Wölfle

Unternehmen müssen sich in einer immer anspruchsvolleren Umwelt behaupten. Der Wettbewerb ist hoch, Rahmenbedingungen und Märkte verändern sich schnell. Arbeitsteilung hat generell zugenommen. Agilität ist gefragt, um Chancen zu nutzen und Risiken zu minimieren. Agilität erfordert Transparenz und die Möglichkeit zu schnellem, effizientem und wirksamem Handeln. Das erfordert eine ebenso agile, schnelle und effiziente Informationsinfrastruktur. Deren Kern ist Business Software, das Nervensystem moderner Unternehmen. Anders als in der Natur muss dieses Nervensystem durch die Organisation selbst gestaltet werden. Dem Gestalteten kann Business Software Reichweite und effiziente Wirksamkeit verleihen, sie ist der Intelligenzverstärker einer Organisation [Kagermann/Österle 2006].

In diesem Beitrag wird Business Software zunächst von anderen Anwendungen im Unternehmen abgegrenzt. Nach einer Begriffsklärung folgt eine vertiefte Erörterung der Charakteristika von Standardsoftware und Individualsoftware. Dabei werden einige Aspekte zu ihrem Aufbau und den typischen Betriebsformen erläutert und in Hinblick auf die Konsequenzen im Lebenszyklus der Systeme besprochen. Die vergleichsweise ausführliche Betrachtung der Systemmerkmale soll helfen, die in den Fallstudien jeweils nur mit wenigen Begriffen bezeichneten Lösungen einzuordnen. Nach den Systemmerkmalen wird das Titelthema des Buchs aufgegriffen. Zunächst wird aufgezeigt, dass es miteinander konkurrierende Anforderungen an Business Software gibt und dass diese gegeneinander ausbalanciert werden müssen. Nach einer Überleitung mit Aussagen aus einer Befragung zu Misserfolgen mit Business Software werden einige zentrale Faktoren für Erfolg herausgearbeitet. Dauerhafter Erfolg ist nur in einem kontinuierlichen Entwicklungsprozess mit einem systematischen Regelkreis zu erreichen. Das Kapitel ist kurz gefasst, weil das Thema in den anderen Fachbeiträgen und den Fallstudien vertieft wird. Abschliessend wird für einige Fallstudien gesagt, welche Haltung und Herangehensweise an Business Software sie repräsentieren.

1.1 Die Vielfalt der Anwendungen eines Information Workers

Im beruflichen Alltag fallen viele Arten von Tätigkeiten an, die mit Informationsverarbeitung zu tun haben. Sie werden durch viele Arten von Informationssystemen unterstützt. Die Übersicht in Abb. 1.1 und die Erläuterungen dazu sollen Business Software von anderen Anwendungen im Unternehmen abgrenzen.

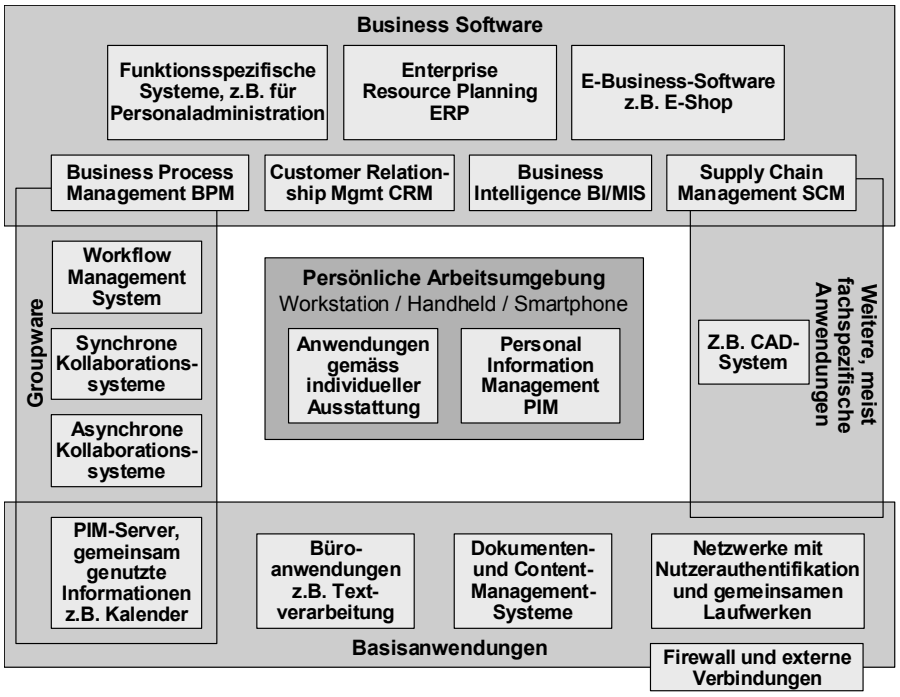

Abb. 1.1: Übersicht über Anwendungen im Unternehmen

Persönliche Arbeitsumgebung

Ausgangspunkt der Betrachtung ist die *persönliche Arbeitsumgebung* eines Mitarbeitenden. Sie ist durch ein Benutzerendgerät bestimmt, das ihm Zugriff auf die Informatikmittel der Organisation gewährt. Klassischerweise ist das eine Workstation an seinem Arbeitsplatz, zunehmend sind es auch mobile Geräte von Laptops bis zu Smartphones. Auf dem persönlichen Gerät kann die ganze Bandbreite der Unternehmensapplikationen installiert sein. Es kann aber auch gar kein Anwendungsprogramm lokal installiert sein, z.B. wenn alle Anwendungen von einem Unternehmensserver bereitgestellt werden. In vielen Fällen handelt es sich um eine

Mischung: Einige Anwendungen laufen lokal und stehen damit auch dann zur Verfügung, wenn kein Zugang zum Netzwerk besteht. Andere werden zentral betrieben oder sind aufgeteilt in eine Client- und eine Serverkomponente. Ein *Personal Information Management System (PIM)*, z.B. für Kontakte oder Termine, muss bei mobilen Geräten immer lokal betrieben werden. Ist es Teil eines Client-Server-Systems, ist lokal die Client-Komponente installiert (z.B. MS Outlook), die mit einem zentralen *PIM-Server* korrespondiert (z.B. MS Exchange Server). Die weitere Softwareausstattung auf dem Benutzerendgerät ist individuell festzulegen. Universell einsetzbarer *Basisanwendungen* wie Büroanwendungen werden häufig lokal installiert, obwohl der in der Grafik abgebildete zentrale Betrieb an Bedeutung gewinnt. Bei lokalen Anwendungen ist immer zu beachten, dass die Daten anderen Nutzern im Netzwerk ohne besondere Vorkehrungen nicht zur Verfügung stehen und dass die Organisation keine zentrale Datensicherung betreiben kann.

Basisanwendungen und lokales Netzwerk

Die meisten Computerarbeitsplätze sind in ein lokales *Netzwerk (LAN)* eingebunden. Zu den *Netzwerkdiensten* gehören die Authentifikation der Benutzer, das Bereitstellen gemeinsamer Laufwerke sowie sichere, aus- und eingehende Verbindungen zu externen Netzen. Gemeinsam genutzte *Basisanwendungen* wie Dokumentenmanagementsysteme müssen zentral ausgeführt werden.

Groupware

Groupware-Anwendungen stellen Personen in Unternehmen oder Arbeitsgruppen zentral Informationen und Kommunikationsmöglichkeiten zur Verfügung. *Workflow Management Systeme* bilden einmalige oder wiederkehrende Geschäftsprozesse ab und können dabei auch Applikationen aus dem Bereich Business Software integrieren. Alle übrigen Anwendungen unterstützen eher schwach strukturierte Vorgänge und erzeugen meist ebenfalls schwach strukturierte Daten in Form von Geschäftsdokumenten. Bei *Kollaborationssystemen* wird zwischen *synchronen* und *asynchronen* unterschieden. *Synchrone Kollaborationssysteme* unterstützen die gleichzeitige Arbeit mehrerer Personen, z.B. *Instant Messaging* oder *Gruppeneditoren*. Die zeitversetzte Zusammenarbeit wird durch *asynchrone Kollaborationssysteme* unterstützt, z.B. *E-Mail* oder *Wikis*.

1.2 Business Software

Der Begriff *Business Software* wird als Überbegriff für alle Arten betriebswirtschaftlicher Software verwendet. *Business Software* unterstützt Tätigkeiten mit zuvor definierten Funktionen. Sie sollte rollenbasiert sein und den Akteuren die benötigten Funktionen zur Verfügung stellen. Arbeitsergebnisse werden als struk-

turierte Informationen in Datenbanken gespeichert. *Business Software* schliesst sowohl ERP-Systeme als auch E-Business-Software ein.

Als *Enterprise Resource-Planning-System (ERP-System)* bezeichnet man integrierte betriebswirtschaftliche Standardsoftware, die auf einer gemeinsamen Datenbasis Funktionen für mehrere Fachbereiche eines Unternehmens bereitstellt. Die meisten ERP-Systeme bestehen aus Modulen, die die Funktionalität der einzelnen Fachbereiche enthalten, also z.B. für das Rechnungswesen, die Finanzwirtschaft, das Personalwesen, die Materialwirtschaft und den Vertrieb. Klassische ERP-Systeme stehen für die unternehmensinterne oder konzerninterne Integration. Zunehmend werden auch branchenspezifische Module oder branchenspezifisch vorkonfigurierte ERP-Systeme angeboten (vgl. ABACUS AbaBau in der Fallstudie Weiss+Appetito, S. 51). Viele moderne ERP-Systeme integrieren auch E-Business-Software und machen sich Vorteile von Internetanwendungen zu Nutze, die Gartner Group prägte in diesem Zusammenhang den Begriff ERP II (vgl. OpaccOne WebSales in der Fallstudie F. + H. Engel, S. 65).

E-Business-Software unterstützt primär unternehmensübergreifende Geschäftsprozesse (z.B. ein Onlineshop oder eine Buy-Side-E-Procurement-Lösung). *E-Business* ist die Unterstützung der Beziehungen und Prozesse eines Unternehmens mit seinen Geschäftspartnern, Kunden und Mitarbeitenden durch elektronische Medien [in Anlehnung an Schubert/Wölfle 2000]. *E-Commerce* ist derjenige Teil des E-Business, der auf den Verkauf von Produkten und Dienstleistungen ausgerichtet ist. E-Commerce-Applikationen dienen der elektronischen Unterstützung des Kaufprozesses [Schubert et al. 2001]. *E-Procurement* ist die elektronische Unterstützung der Beschaffungsprozesse (Einkauf) eines Unternehmens. Während Warenwirtschaftsmodule in ERP-Systemen primär für die Beschaffung direkter Güter eingesetzt werden, unterstützen E-Purchasing-Lösungen vor allem den Einkauf indirekter Güter und Dienstleistungen (vgl. Fachbeitrag Tanner S. 151). *Business-Intelligence-Systeme (BI)* unterstützen das interne Reporting durch die Zusammenstellung, Analyse und Verdichtung von Daten für das Management. Die Daten stammen meist aus verschiedenen Quellen und werden in *Data-Warehouse-Systemen (DWH)* dauerhaft und unabhängig von den Quellsystemen gespeichert.

Customer Relationship Management (CRM) und *Supply Chain Management (SCM)* sind Managementkonzepte, für die sich eigenständige Softwaregattungen herausgebildet haben. *Customer Relationship Management* ist verkaufsorientiert und fokussiert auf Kundenbedürfnisse. Ziele von CRM-Massnahmen sind die Steigerung der Kundenbindung und die Optimierung des Lifetime Values von Kunden. *Supply Chain Management* (Management eines Wertschöpfungsnetzwerks) ist die Koordination einer strategischen und langfristigen Zusammenarbeit von Ko-Herstellern und Händlern zu Entwicklung, Herstellung und Vertrieb von Produkten. Dies beinhaltet sowohl Produktion, Beschaffung und Distribution als auch Produkt- und Prozessinnovationen [angelehnt an Schönsleben 2004].

Umfang der Funktionalität von Business Software

Business Software kann einen sehr unterschiedlichen Funktionsumfang haben. Wenn ein Mitarbeitender im Vertrieb eines Maschinenbauunternehmens ein kleines Programm zur Kalkulation eines Leasingvertrags nutzt, so handelt es sich um eine Business Software. Sie unterstützt in diesem Fall nur eine einzige Funktion. Geht die Funktionalität etwas weiter und können z.B. auch noch die für einen Vertrag erforderlichen Dokumente erzeugt werden, ist der Fokus ein Prozess. Werden Funktionen für verschiedene Prozesse unterstützt, handelt es sich um eine Fachbereichslösung, z.B. für die Personaladministration oder Buchhaltung. Die nächste Stufe ist die Zusammenfassung einer Lösung für mehrere Fachbereiche in einer Unternehmenssoftware. Grössere Unternehmen können aus Teilorganisationen bestehen, was eine Software mit Fähigkeiten einerseits zur Integration, andererseits aber zur Unterscheidung der Geschäftsbereiche oder sogar rechtlich eigenständigen Teilunternehmen eines Konzerns erfordert. Datentechnisch und organisatorisch voneinander getrennte Bereiche innerhalb einer Software heissen *Mandant*. Schliesslich gibt es Softwarelösungen, die eine unternehmensübergreifende Zusammenarbeit unterstützen, z.B. in einer Supply Chain.

1.3 Standardsoftware und Individualsoftware

Auf allen Granularitätsebenen kann Business Software individuell entwickelt werden. Alternativ kann für viele Fachbereiche eine Standardsoftware auf dem Markt bezogen werden. Die wichtigsten Vor- und Nachteile der beiden Konzepte sind in Tab. 1.1 dargestellt. Wegen der hohen Kosten für die Pflege individueller Lösungen und des grossen Angebots an leistungsfähiger Standardsoftware hat letztere ab etwa 1990 in vielen Bereichen einen Siegeszug angetreten. Daraus kann aber nicht abgeleitet werden, dass Standardsoftware in jedem Fall eine bessere oder kostengünstigere Lösung ist als Individualsoftware.

Tab. 1.1: Vor- und Nachteile von Standardsoftware und Individualsoftware

	Vorteile	**Nachteile**
Standard-software	• schnell verfügbar • ist mit Know-how-Bezug von Softwareanbieter und Einführungspartner verbunden	• Anforderungen sind möglicherweise suboptimal erfüllt • fremdbestimmte Weiterentwicklung
Individual-software	• kann Bedürfnisse exakt abbilden • ist der Konkurrenz nicht zugänglich	• hohe Kompetenz erforderlich • permanenter Pflegeaufwand muss individuell geleistet werden

1.3.1 Standardsoftware

Die Grundannahmen von Standardsoftware sind, dass Geschäftsprozesse bei Wiederholung gleich ablaufen und damit eine definierbare Struktur haben, und dass die Geschäftsprozesse in verschiedenen Unternehmen viele Gemeinsamkeiten aufweisen. Standardsoftware bietet deshalb ein Datenmodell und einen Funktionsvorrat, der die Summe der in allen Anwenderunternehmen erwarteten Anforderungen abdeckt. Dabei sind teilweise sehr mächtige Systeme entstanden, die eine Vielzahl von Anforderungen in unzähligen Kombinationsmöglichkeiten abdecken. Jedes einzelne Unternehmen nutzt nur eine Auswahl der verfügbaren Funktionen. Einige davon können zudem noch an individuelle Bedürfnisse angepasst werden. Die Anpassung von Standardsoftware an die Bedürfnisse einzelner Unternehmen heisst *Customizing*. Dies kann erfolgen durch:

- die Auswahl von Programmmodulen,
- durch Parametrisierung, das heisst das Vornehmen von Einstellungen, die im Programm als Varianten vorgesehen sind und nicht in den Quellcode der Programme eingreifen, z.B.
  - Ländereinstellungen wie Sprache, Währungen, Masse etc.,
  - die Abbildung der Organisationsstruktur, der Produktstruktur, des Kontenplans etc.,
  - die Auswahl vorgesehener Varianten bei Prozessen, Funktionen, Daten,
- durch Programmierung mit mitgelieferten Werkzeugen, z.B.
  - Masken-/Formular-/Reportgeneratoren,
  - erweiterbare Tabellen,
  - kundenspezifische Programmierung an vorgesehenen Stellen durch Skripts oder in der Programmiersprache des Systems geschriebenen Code,
- durch Eingriff in den Quellcode (ermöglichen nicht alle Anbieter) sowie
- durch externe Programmierung oder die Integration mit externer Software, z.B. Integration eines Webshops mit einem ERP-System.

Je umfassender die Konfigurationsmöglichkeiten sind, desto grösser ist der mit dem Customizing verbundene Aufwand. In der Regel kann es nur durch Spezialisten ausgeführt werden. Dadurch entstehen Kosten, die bei komplexeren Anwendungen die Kosten der Softwarelizenzen deutlich überschreiten können. Um diese zu begrenzen, bieten manche Implementierungsanbieter branchenorientiert vorkonfigurierte Lösungen an. Die umstrittenste Anpassungsmöglichkeit ist der Eingriff in den Quellcode. Er bewirkt, dass auch ungewöhnliche Anforderungen mit relativ wenig Initialaufwand realisiert werden können, was diese Option zu Beginn attrak-

tiv erscheinen lässt. Eine Hinterfragung der Anforderung unterbleibt dann oft. Der kurzfristige Zeitgewinn muss allerdings mit dem Verlust der Releasefähigkeit bezahlt werden.

Releasefähigkeit von Standardsoftware

Ein zentraler Vorteil von Standardsoftware ist, dass der Aufwand zur Pflege der Software nicht individuell, sondern von einem Anbieter zentral für alle Anwender geleistet werden kann. *Softwarepflege* beinhaltet die Anpassung an sich ändernde technische Umgebungen, z.B. eine neue Version eines Betriebssystems, an sich ändernde externe Sachverhalte, z.B. eine neue Vorschrift zur Mehrwertsteuerabrechnung, die Beseitigung von Fehlern und ggf. die Erweiterung der Funktionalität. Zur Übernahme der Änderungen bieten die Softwareanbieter ihren Kunden regelmässige Updates oder, bei grösseren Änderungen, Releasewechsel an. Im Idealfall installiert der Kunde die neue Version entsprechend den Anweisungen des Anbieters und kann anschliessend in gewohnter Weise weiterarbeiten. Ist das möglich, wird die Software als *releasefähig* bezeichnet. In der Praxis ist Releasefähigkeit oft nur eingeschränkt gegeben. In jedem Einzelfall muss dann geprüft werden, ob die neue Version unerwünschte Nebeneffekte nach sich zieht. Das ist mit aufwendigen Tests, Fehlersuche und Korrekturmassnahmen verbunden. Ursache der Probleme bei Updates können anbieterseitig Mängel an der Software, anwenderseitig unsachgemässe Anpassungen oder extern durch die Systemumgebung (Plattformen, Schnittstellen) ausgelöste Störungen sein. Bei einem Eingriff in den Quellcode geht die Releasefähigkeit definitiv verloren und die Lösung verliert ihren Charakter einer Standardlösung. Deshalb erfordert Standardsoftware Anstrengungen für die Aufrechterhaltung des Standards. Das bedeutet für den Anbieter, dass er ein Systementwicklungskonzept einhält, das ihm ermöglicht, Aufwärtskompatibilität zu gewährleisten. Der Anwender muss diese Rahmenbedingungen kennen und so weit wie möglich einhalten.

Abb. 1.2 veranschaulicht das grafisch: Der Softwarekern mit seinem Quellcode bleibt für externe Anpassungen unangetastet, denn durch seine Integrität garantiert der Hersteller die Qualität seiner Software – auch über Versionswechsel hinweg. Das *Customizing* erfolgt mit den zur Verfügung gestellten Instrumenten nach den Anweisungen des Anbieters. Dabei greifen die Customizing-Werkzeuge über die systeminterne Schnittstelle A auf den Kern zu. Aufwärtskompatibilität bedeutet, dass bei einem Update des Softwarekerns die Einstellungen des Customizings erhalten bleiben und die ergänzte Funktionalität weiterhin funktioniert, weil sich A auch in der neuen Version gleich verhält wie in der alten. Auch eine externe Schnittstelle B würde nach einem Update weiterhin funktionieren, wenn sie ausschliesslich im Rahmen der Instrumente des Customizings realisiert wurde. Um Stabilität sicherzustellen, betreibt der Anwender die Software nur auf Plattformen, die den Vorgaben des Anbieters entsprechen.

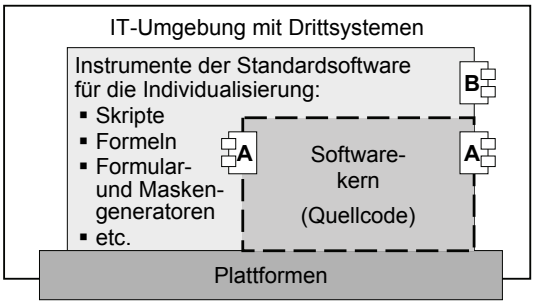

Abb. 1.2: Elemente einer Standardsoftware

Dem Konzept der Einhaltung des Standards kann entsprochen werden, wenn die Software die wichtigen Anforderungen des Anwenders im Standard erfüllt. Zudem ist es erforderlich, dass der Anbieter den Umfang der Standardfunktionalität laufend entsprechend den sich ändernden Arbeitsweisen in der Branche und den Anforderungen seiner Kunden erweitert, so dass auch die Weiterentwicklung im Rahmen des Standards erfolgen kann. Ein Beispiel dafür, dass dies über 15 Jahre möglich ist, zeigt die Fallstudie F. + H. Engel (S. 65).

Werden diese Überlegungen nicht beachtet und Anpassungen vorgenommen, die die Releasefähigkeit beeinträchtigen, verliert die Standardsoftware in der Betriebsphase ihren Standardcharakter. Bei jedem Update sind die Anpassungen zu testen und allenfalls wieder herzustellen. In einem eng begrenzten Bereich mag das tragbar sein. Je weiter aber Anpassungen vorgenommen wurden, desto unkalkulierbarer wird der Aufwand für die Zukunft. Das schlägt sich in hohen laufenden Kosten für die Softwarewartung nieder. Ein Know-how-Problem kann aufkommen, da die Betreuer der Standardsoftware sowohl deren historische Entwicklung als auch die Anpassungen gut kennen müssen, um das System richtig zu beurteilen. Es kann eine Situation entstehen, die schwieriger ist als bei einer Eigenentwicklung, da das Originalsystem ja nicht selbst entwickelt wurde und viele Zusammenhänge darin für den Entwickler nicht transparent sind. Irgendwann kann man das System nur noch so nutzen, wie es ist (inkl. der alten Plattform), oder abstellen.

Wartungsvertrag

Um von der laufenden Systemwartung des Anbieters zu profitieren, sind Wartungsverträge üblich. Das Anwenderunternehmen bezahlt jährlich einen bestimmten Prozentsatz auf die Lizenzkosten bei der Anschaffung, meistens in einer Bandbreite zwischen knapp unter 15 % und gut 20 %. Bei den enthaltenen Leistungen gibt es grosse Unterschiede. Bei einigen Anbietern beinhalten sie lediglich Updates zur Aufrechterhaltung des Status quo, bei anderen auch Upgrades oder Release-

wechsel, die mit Funktionserweiterungen verbunden sind. Nur wenige Anbieter garantieren Aufwärtskompatibilität. Das ist insofern relevant, als dass die Kosten, die im Zusammenhang mit der Installation der neuen Version anfallen, nicht abgedeckt sind. Einige Anbieter verbinden den Wartungsvertrag mit einem Supportvertrag, der z.B. Anwendersupport beinhaltet. Aus Sicht der Anwenderunternehmen ist eine flexible und anpassbare Definition des Leistungsumfangs wünschenswert.

Das Zusammenspiel mehrerer Systeme

Schon die vorangegangenen Ausführungen machen deutlich, dass für einen erfolgreichen Betrieb ein störungsfreies Zusammenspiel mehrerer Elemente erforderlich ist. Je mehr Elemente beteiligt sind und aus je heterogeneren Quellen sie kommen, desto höher sind die zu erwartenden Schwierigkeiten und Kosten im dauerhaften Betrieb. Die Abhängigkeiten treten auf drei Ebenen auf:

- Plattformen: Hardware und Betriebssysteme, auf denen die Anwendungssoftware läuft
- Logisches Systemmodell der Anwendungssoftware: Datenschicht, Applikationsschicht und Client-Schicht
- Anwendungslandschaft: das Zusammenspiel mehrerer Anwendungssysteme

Während bei Plattformen allgemein die Strategie einer Reduktion der Vielfalt verfolgt wird, gibt es beim Logischen Systemmodell und der Anwendungslandschaft unterschiedliche Ansätze. Die Definition und Beherrschung der logischen Systemarchitektur ist die Domäne des Softwareanbieters. Seit einigen Jahren sind die Anforderungen aber v.a. auf der Client-Seite immer heterogener geworden, weil der Zugriff einerseits von immer mehr Endgeräten und auch mobil erfolgen soll, andererseits durch E-Business neue Zugriffsanforderungen entstanden sind. Bei den Datenbanken suchen die Anwenderunternehmen insofern Synergien, als dass sie für verschiedene Systeme nicht verschiedene Datenbanken betreiben wollen – analog zu den Plattformen. Deshalb beziehen sie bisweilen die Unterstützung eines bereits vorhandenen Datenbanksystems als Anforderung in die Systemevaluation mit ein. Es ist zu beobachten, dass Anbieter ihre Anwendungssysteme für Zugriffe unterschiedlicher Art und die Anbindung an verschiedene Datenbanken vorbereiten. Damit erhält das Anwenderunternehmen Wahlmöglichkeiten bei der Realisierung der drei Schichten. Zudem gibt es Anforderungen, die auf eine Erweiterung der Drei-Schichten-Architektur hinauslaufen, z.B. durch eine grafische Modellierungsschicht für die Gestaltung anwenderspezifischer Prozesse. Die Komplexität im Systemaufbau nimmt dadurch zu. Je nach Alter und Architektur des Softwaresystems sind derartige Änderungen gar nicht möglich. Einige Anbieter haben deshalb begonnen, ihre Systemarchitektur schrittweise zu ändern oder die Anwendung komplett neu zu schreiben. In der Übergangsphase, in der sich viele Anbieter befinden, sind Komponenten aus unterschiedlichen Generationen parallel im Einsatz.

Für Anwenderunternehmen ist die Qualität der Systemarchitektur einer Business Software kaum zu beurteilen. Darauf kommt es aber auch nicht an. Entscheidend ist, dass der Anbieter in der Lage ist, sein System dauerhaft zu pflegen und in engem Kontakt mit seiner Zielgruppe entsprechend deren Anforderungen weiterentwickelt. Indikatoren dafür, ob ein Anbieter das kann, sind die Weiterentwicklung des Systems in der Vergangenheit, Kontinuität beim Management, die Ertragskraft des Unternehmens und allenfalls die Zukunftsfähigkeit der Programmiersprache.

Bei der Anwendungslandschaft stellt sich zunächst die Frage, wie viele Business-Software-Systeme ein Unternehmen braucht. Im einfachsten Fall, der *All-in-One-Strategie*, beschränkt sich ein Unternehmen auf ein System, z.B. ein umfassendes, integriertes ERP-System. Das besteht in sich zwar auch aus mehreren Komponenten, aber das Anwenderunternehmen kann alle Entscheidungen immer auf die Gesamtheit des Systems beziehen und der Anbieter ist im gesamten Lebenszyklus für die Orchestrierung der Teile verantwortlich (vgl. Fallstudie F. + H. Engel S. 65, Fallstudie Weiss+Appetito S. 51). Bei der *Hybrid-Strategie* ist das nicht möglich. Der Differenzierungsgrad einzelner Anforderungsbereiche ist so hoch, dass teilweise von der All-in-One-Strategie abgewichen werden muss. Anwenderunternehmen versuchen meist dennoch, sich auf die Produkte eines Anbieters zu konzentrieren, weil sie erwarten, dass dieser deren Integration am besten unterstützt. Dennoch trifft das Anwenderunternehmen Entscheidungen für jedes Teilsystem separat. Im Lebenszyklus kann es dann vorkommen, dass ein System auf eine neue Version migriert werden soll, die anderen aber nicht. Dabei können Konstellationen entstehen, die der Anbieter nicht unterstützt. Ein Upgrade bei einer Komponente erzwingt dann auch Upgrades bei anderen.

Um die Flexibilität zu erhöhen, werden zunehmend Integrationsplattformen eingesetzt (vgl. Fallstudie Variosystems S. 81). Ihr Vorteil besteht darin, dass jedes einzelne System nicht mehr direkt an ggf. mehrere Umsysteme angebunden wird, sondern indirekt. Dadurch werden die einzelnen Anwendungen *lose gekoppelt*, das heisst, sie sind in ihrem Lebenszyklus weniger abhängig von einander und können allenfalls auch einzeln ausgetauscht werden. Diese Anforderung stellt auch die dritte Variante, die *Best-of-Breed-Strategie*. Hier kombiniert ein Unternehmen Anwendungen ohne Rücksicht auf ihre Herkunft und bindet ggf. auch externe, als Service betriebene Lösungen, mit ein.

Je mehr Systeme ein Unternehmen im Einsatz hat, desto mehr IT-Know-how und eigene Ressourcen sind erforderlich. Es ist schwierig, Systemanbieter für Fehlfunktionen in die Pflicht zu nehmen, wenn Störungsursachen nicht klar lokalisiert werden können. Um die Situation dennoch zu beherrschen, ist eine *IT-Strategie* erforderlich. Sie definiert mit Bezug zur Unternehmensstrategie Grundpositionen, nach denen die Systemlandschaft zu entwickeln ist. Organisatorisch legt sie fest, wie Entscheidungen gefällt werden und Verantwortlichkeiten geregelt sind.

Das Zusammenspiel mehrerer Programme ist nicht nur eine technische Herausforderung. Sehr häufig haben die Anwendungen Überschneidungen in der Funktionalität und bilden gleiche Daten ab. In diesem Fall ist die Arbeitsteilung zunächst auf konzeptioneller Ebene festzulegen. Welche Funktionen werden wo ausgeführt, z.B. die Pflege von Adressen, und wo werden die Daten gehalten. Doppelspurigkeiten sind so weit wie möglich zu vermeiden. Wenn Daten an mehreren Stellen geführt werden müssen, ist festzulegen, wie der Abgleich zwischen diesen erfolgt. Um die Arbeitsteilung in Anwendungslandschaften zu visualisieren werden in der Anwendungssicht der eXperience-Systematik den einzelnen Systemen immer die wichtigsten Funktions- und Datenbereiche zugeordnet.

Eine Frage, die am Anfang der Arbeit mit einem neuen System immer gestellt werden sollte, ist die nach dem Ende. Das Interesse gilt dann den Daten. Denn die Lebensdauer von Daten ist meistens länger als die der Systeme. Am Ende des Lebenszyklus einer Anwendung steht deshalb die Aufgabe, die mit dem System verwalteten Daten ohne Verlust an Strukturinformationen so aus dem System zu exportieren, dass sie mit einem neuen System weiter verwendet werden können. Wenn geeignete Exportfunktionen fehlen, die Datenstruktur nicht bekannt ist, die Daten in einem proprietären Datenformat oder mit herstellerabhängigen Schutzmechanismen gehalten werden, können sich hier erhebliche Probleme ergeben.

1.3.2 Individualsoftware

Viele der vorangegangenen Ausführungen gelten auch für Unternehmen, die Individualsoftware einsetzen. Denn kein Unternehmen arbeitet mehr gänzlich mit individuellen Systemen. In aller Regel wird Individualsoftware neben Standardsoftware eingesetzt. Sie läuft auf Standardplattformen und nutzt z.B. handelsgängige Datenbanksysteme oder universelle Clientsoftware. Der Trend zu *serviceorientierten Architekturen* unterstützt ein Systemverständnis, das auf Komponenten beruht und Monolithen ablöst. Am Markt ist ein sehr grosses Angebot verfügbar an Open Source Software auf allen Granularitätsebenen und freien oder kommerziellen Programmkomponenten (vgl. Fallstudie BSCC S. 261).

Es sind primär drei Gründe, aus denen sich Unternehmen heute für Individualsoftware entscheiden. Der eine sind Wettbewerbsgründe. Software kann ein spezielles Know-how beinhalten, das einen Wettbewerbsfaktor darstellt. Dieses soll keinem Wettbewerber und ggf. auch keinem externen IT-Anbieter zugänglich werden. Der andere Grund sind sehr spezielle Anforderungen. Bei ihnen ist die Zahl der Unternehmen, die gleiche Anforderungen haben, zu klein, als dass es sich für einen Softwareanbieter lohnen würde, sie in einer Standardlösung abzudecken. Der dritte Grund ist Agilität und Selbstbestimmung. Eine eigenentwickelte Lösung hat genau den Umfang, den das Anwenderunternehmen benötigt. Im Vergleich zu Standardsoftware, die aufgrund ihrer Universalität immer auch unbenötigte Funktionen und Parametrisierungen enthält, kann sie sehr viel schlanker und einfacher gehalten

werden. Eine kompetente Architektur und gute Softwarequalität vorausgesetzt, kann mit geringeren Hardwareressourcen eine höhere Performance erreicht werden. Soll etwas geändert werden, kann dies ohne Einschränkung auf allen Ebenen erfolgen und es müssen keine teuren externen Berater beigezogen werden. Es muss kein Ballast unbenötigter Schichten mitgeschleppt und keine Rücksicht auf extern bestimmte Updates genommen werden. Das bewirkt eine hohe Agilität im Auftreten am Markt, die unabhängig von der eigentlichen Funktionalität ein Wettbewerbsfaktor sein kann (vgl. Fallstudien LeShop S. 195 und Digitec S. 217).

Der kritische Erfolgsfaktor für Eigenentwicklungen ist das Know-how der für die Entwicklung verantwortlichen Personen. Es ist nicht nur zurzeit der Erstentwicklung, sondern über den gesamten Lebenszyklus des Systems erfolgskritisch für die Zielerreichung und Wirtschaftlichkeit der Lösung. Ist die Kompetenz personengebunden und die Person steht nicht mehr zur Verfügung, kann eine Individualsoftware zur Belastung werden. Wichtig sind auch hier die Architektur und Modularität des Systems. Sind seine Elemente lose gekoppelt, unterstützt das eine flexible Erneuerung. Z.B. konnte LeShop sein Ursprungssystem nach und nach durch Eigenentwicklungen ablösen (vgl. Fallstudie auf S. 195). Digitec geht von vorne herein davon aus, sein System alle paar Jahre neu zu entwickeln (vgl. Fallstudie auf S. 217).

1.3.3 Betriebsformen von Business Software

Für Betrieb und Nutzung von Business Software sind unterschiedliche Konzepte verfügbar. Abb. 1.3. zeigt eine Übersicht. Wichtige Unterscheidungsmerkmale sind u.a., ob auf dem Endgerät des Benutzers eine anwendungsspezifische Software installiert sein muss und welche Varianten in Bezug auf Eigenbetrieb, Outsourcing des Betriebs oder Miet-ähnliche Bezugsformen bestehen.

Business Software, die einzig auf einem *Personal Computer* installiert ist, ist eine Option für Kleinstbetriebe. In grösseren, arbeitsteiligen Organisationen mag sie für die isolierte Unterstützung einzelner Funktionen, wie z.B. der Kalkulation eines Leasingvertrags, noch vertretbar sein. Ansonsten überwiegen hier die Nachteile, die aus der Isoliertheit entstehen. Die am weitesten verbreitete Betriebsform ist die der Nutzung von Systemen mit *Client-/Server-Architektur* (vgl. Fallstudie ad AUGROS S. 109). Bei diesen ist die Software zweigeteilt. Auf der zentralen Serverkomponente liegen zumindest alle Daten, so dass mehrere Nutzer gleichzeitig auf dem System arbeiten können. Auf den Endgeräten der Nutzer ist die Client-Komponente der Business Software installiert. Vorteile dieses Konzepts sind die leichte Erweiterbarkeit der Anwendergruppe und die Flexibilität bei der Ausstattung des Anwenderendgeräts. Nachteilig sind dagegen der hohe Preis der Fat-Client-Endgeräte und die auf Grund der Dezentralität und Heterogenität aufwändige Wartung der Anwendergeräte. Diese Nachteile bestehen bei der alten Architektur des monolithischen *Host-Systems* mit unselbständigen Computerterminals

nicht. Bei ihr werden alle aktiven Komponenten zentral betrieben. Nach heutigen Massstäben ist sie aber zu unflexibel. Deshalb wurde sie in den vergangenen zwanzig Jahren weitgehend durch Client-/Server-Systeme ersetzt.

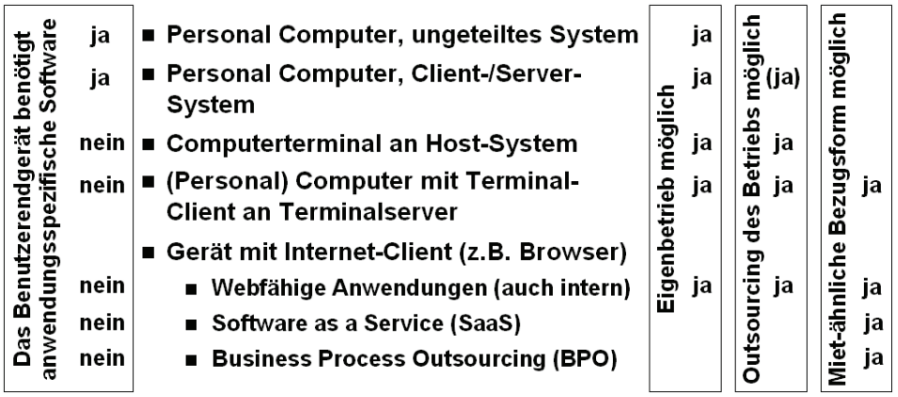

Abb. 1.3: Unterschiedliche Konzepte für Betrieb und Nutzung von Business Software

Um die Nachteile des teuren Systembetriebs auf Personal Computern zu vermeiden, wird die *Terminal-Server-Technologie* genutzt (vgl. Fallstudie Weiss+Appetito S. 51). Bei ihr werden die Programme, die normalerweise auf dem Rechner des Anwenders laufen, auf einem zentralen Terminal-Server emuliert. Der gibt nur noch Ein- und Ausgabesignale an die Terminal-Clients auf den Benutzerendgeräten. Diese benötigen dabei nur eine minimale Systemausstattung (*Thin-Client)*. Nachteil dieses Konzepts sind einerseits die Kosten der Terminal-Server-Technologie, andererseits die geringen Freiheitsgrade der Anwender.

Das modernste Konzept ist das der *Webanwendungen*. Hier sind die Systeme von vorne herein für einen zentralen Betrieb konzipiert, während die Anwender mit universellen Internet-Clients, z.B. einem Browser, auf das System zugreifen. Dieses Konzept zeichnet sich durch eine hohe Flexibilität beim Anwenderzugriff und die Vorteile eines vollständig zentralen Systembetriebs aus, wobei die Nachteile der starren Host-Systeme überwunden sind. Da die klassischen Internettechnologien den hohen Performance- und Stabilitätsanforderungen von Business Software nicht gerecht werden, erweiterte man diese mit Technologieergänzungen zu *Rich Internet Applications RIA*. Webanwendungen erlauben zudem neue Bezugsformen als Alternative zum Kauf von Software. Im Fall von *Software as a Service (SaaS)* (vgl. Fallstudie BSCC S. 261) erwirbt das Anwenderunternehmen keinen Besitz an der Software, sondern vergütet deren Nutzung ähnlich einer Miete. Damit ist das Anwenderunternehmen zudem von der Beschaffung der Hardware und Sicherstellung des Betriebs entlastet. Noch einen Schritt weiter geht das umfassende Konzept

des *Business Process Outsourcing (BPO)*, bei dem die Bereitstellung von Softwarefunktionalität eine Teilleistung einer weitergehenden Dienstleistung ist (vgl. Fallstudie UBS S. 161).

In Bezug auf dauerhaften Erfolg mit Business Software kann jedes der Konzepte „Personal Computer mit Client-/Server-System", „Terminalserver" und „Webanwendungen" zu guten Ergebnissen führen.

1.4 Dauerhafter Erfolg mit Business Software

Aufgabe der Business Software mitsamt ihrer Infrastruktur ist es, dem Unternehmen die erwartete Funktionalität zur Verfügung zu stellen. Die im engeren Sinne funktionale Erwartung ist allerdings mit weiteren Erwartungen verbunden. In der Summe stehen sie für Wirksamkeit und Wirtschaftlichkeit, Heinrich spricht von der *Strategischen Schlagkraft* der Informatik [1996]. Allerdings stehen einige Erwartungen in Konkurrenz zueinander und können nicht gleichzeitig optimiert werden (vgl. Abb. 1.4 in Anlehnung an Gliedman [2002]). Zu den weiteren Erwartungen gehört Sicherheit im Zusammenhang mit der Systemnutzung. Sie zeigt sich z.B. in einer hohen Systemverfügbarkeit oder Datenintegrität. Sicherheit steht in Konkurrenz zu Flexibilität, die für Anpassungsfähigkeit an zukünftige Anforderungen gefordert wird. Eine volle Erfüllung der funktionalen Anforderungen steht in Konkurrenz zu den im Allgemeinen angestrebten geringen Kosten.

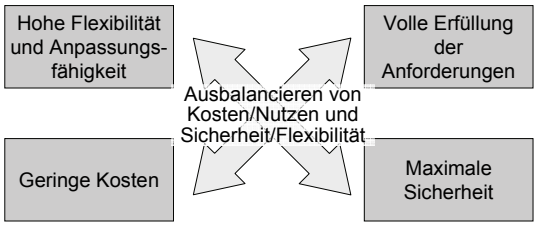

Abb. 1.4: Konkurrierende Anforderungen an Business Software [vgl. Gliedman 2002]

Um erfolgreich zu sein, müssen die konkurrierenden Anforderungen gegeneinander ausbalanciert werden. Dabei können die Gewichte in verschiedenen Anwendungsbereichen des Unternehmens unterschiedlich gelegt werden.

Misserfolge mit Business Software

Da Business Software auch immer wieder Anlass für kritische Bemerkungen gibt führte das Competence Center E-Business Basel an der FHNW im Frühjahr 2009

die Onlinebefragung *Call for Failures* durch. In ihr wurde öffentlich dazu aufgerufen, kurz zu berichten, welche Probleme, Hürden und Herausforderungen die Betroffenen in Business-Software-Projekten erlebt haben und was man hätte anders machen müssen. Den Antwortenden war Vertraulichkeit zugesichert worden. Auf diesen Call gingen 30 Antworten ein, zum Teil recht ausführlich und mit mehreren Einzelaspekten. Das Ergebnis liefert Stichworte zur Identifikation möglicher Problemfelder, hier sollen lediglich einige mehrfach genannte kurz genannt werden.

Zum Thema Standardsoftware werden ungenügende Funktionalität, Probleme infolge Abweichung vom Standard, aufwändige Implementierung und Kostenüberschreitungen beklagt. Eine Person behandelt ausführlich, wie sie Flexibilität teuer zu stehen kam, ein anderer empfiehlt anschaulich „Baue keinen Mercedes um, um damit Ackerfurchen zu pflügen". Bei Individualsoftware kommt die Bedeutung einer guten Planung der Tests mehrfach zur Sprache. Für die Evaluation stellen viele Personen im Nachhinein fest, dass ihr Verfahren ungeeignet war. Generell spielen Kompetenzdefizite sowohl auf Seiten der Anwenderunternehmen als auch bei den Anbietern eine grosse Rolle. Sie betreffen auch das Projektmanagement und Organisationsaufgaben, von ungenügender Planung über ungeeignete Spezifikationen und, mehrfach genannt, Problemen mit Produktstammdaten.

Einige Erfolgsfaktoren im Umgang mit Business Software

Die Fähigkeit, ein Geschäftskonzept effizient und wirksam umzusetzen, ist in vielen Organisationen davon abhängig, ob es gelingt, das Potenzial von Business Software dafür zu erschliessen. Deswegen ist es erforderlich, dass die Informatik in der Geschäftsleitung vertreten ist und wenigstens ein Geschäftsleitungsmitglied Kompetenzen im Bereich Business Software pflegt. Diese Person muss eine aktive Gestaltungsaufgabe einnehmen und eine optimale Verbindung von Organisation und IT vorantreiben. IT-Fragen im Einzelnen können dabei delegiert werden.

Voraussetzung für *dauerhaften Erfolg* ist, wie in anderen Bereichen auch, ein Regelmechanismus. Der Regelkreis erfordert eine Definition der Ziele, eine regelmässige Messung sowie Beurteilung der Zielerreichung und allfällige Massnahmen zur Verbesserung. Business Software ist dabei einerseits Werkzeug, andererseits zusammen mit den Prozessen Gegenstand der Erfolgsüberwachung.

Unter Abwägung der Anforderungen und Lösungsvarianten muss jedes Unternehmen für sich herausfinden, welche Art Business Software und Betriebsform zu ihm passt. Kernkriterium sind die eigenen Kompetenzen und Kapazitäten. Ergänzend dazu sind geeignete IT-Partner zu finden. Mit der Auswahl einer geeigneten Business Software beschäftigt sich der Beitrag von Gronau und Eggert auf S. 25. Meist ist der Beizug eines externen Spezialisten sinnvoll, denn derartige Evaluationen kommen zu selten vor, um die Kompetenz selbst zu unterhalten.

Bei der Evaluation von Standardsoftware ist der Abdeckungsgrad der Kernanforderungen im Standard das wichtigste Kriterium. Bei den Kosten sollte eine *Total-Cost-of-Ownership*-Betrachtung (TCO) über mehrere Jahre gewählt werden. Einem Verlust der Releasefähigkeit ist so weit wie möglich vorzubeugen und der Implementierungspartner ist diesbezüglich zu kontrollieren, denn für ihn sind Abweichungen vom Standard ein gutes Geschäft. Standardsoftware ist kein Zauberstab und kein Werkzeugkasten. Für klar abgrenzbare Bereiche können Eigenentwicklungen in Betracht gezogen werden. Je mehr man sich aber auf Abweichungen vom Standard einlässt oder Individuallösungen einsetzt, desto wichtiger ist eigenes Know-how zur Beurteilung und Entscheidung der damit verbundenen Fragen.

Ein Einführungsprojekt für Business Software ist fast immer eine grosse Herausforderung für die Organisation. Das Projekt braucht einen starken Rückhalt in der Geschäftsleitung. Diese muss die erforderlichen Ressourcen bereitstellen, Entscheidungen zügig treffen und Konflikte lösen. Dem Kompetenzaufbau und der Motivation der Mitarbeitenden ist grosse Beachtung zu schenken. Ein dauerhaftes Key-User-Konzept und ein früher Einbezug der Key User haben sich oft als vorteilhaft erwiesen. Mit den ausgewählten IT-Partnern ist eine starke Zielorientierung in Verbindung mit gegenseitiger Aufrichtigkeit und Loyalität anzustreben. Kritische Know-how-Konzentrationen bei Einzelpersonen sollten vermieden werden.

1.5 Die Fallstudien in diesem Buch

Die Auswahl der Fallstudien für dieses Buch repräsentiert eine grosse Bandbreite sehr unterschiedlicher Haltungen und Herangehensweisen an Business Software und veranschaulicht die auf den vorangegangenen Seiten geschilderten Varianten. F. + H. Engel konnte sich über 15 Jahre mit einem aufwärtskompatiblen System kontinuierlich weiterentwickeln und durch die Kontinuität eine hohe Produktivität erreichen. Variosystems hält sich streng an die Standardlösungen eines Anbieters und integriert das Zusammenspiel seiner vier weltweit verteilten Konzerngesellschaften. Weiss+Appetito benötigte ein nicht verfügbares Branchenmodul und ging für dieses eine Entwicklungspartnerschaft mit dem Systemanbieter ein. Die Firmen eltromat, ad AUGROS, Blizzard und Finzelberg setzten bei ihrer Auswahl auf Standard-ERP-Systeme, deren Anbieter auf die jeweilige Branche spezialisiert sind. HERWE entschied sich für eine CRM-Lösung, da deren Anbieter aufgrund einer Kooperation Schnittstellen zu ihrer existierenden ERP-Lösung hatte. LeShop und Digitec stehen für Unternehmen, die konsequent auf Individuallösungen setzen und damit in ihren dynamischen Märkten äusserst flexibel handeln können. Schindler und UBS haben tief gehendes eigenes IT-Know-how und optimieren ihre differenzierten Systemlandschaften unter Einbezug externer IT-Service-Provider für integrierte Business-to-Business-Geschäftsprozesse (B2B). BSCC hat eine Lösung gefunden, in der sie keine Business Software selbst betreiben muss.