COMPUTERWOCHE
VOICE OF DIGITAL

Link: https://www.computerwoche.de/a/wird-erp-software-durch-objektorientierte-datenbanken-flexibler,1901281

Vorsicht Technik

Wird ERP-Software durch objektorientierte Datenbanken flexibler?

Datum: 01.09.2010
Autor(en): Christian Riethmüller

ERP-Software soll Prozesse beherrschbarer machen. Doch Anwender sehen sich immer komplexeren Systemen konfrontiert. Hier lesen Sie, ob Objektorientierung einen Ausweg bietet.

Die gute Nachricht kommt gleich am Anfang: Komplexität ist beherrschbar. Zum Hindernis wird sie für ein ERP-System erst dann, wenn sie sich nicht überwinden lässt. Schuld daran sind jedoch nicht die betriebswirtschaftlichen Zusammenhänge, denn Buchungswerte, Konten, Valutadatum und die daraus resultierenden Buchungssätze machen eine ERP-Umgebung nicht kompliziert.

ERP-Systeme und Relationen

Die gegenwärtigen Entwicklungen von **ERP-Systemen**[1] streben nach immer komplexeren Entwicklungs-Tools sowie nach Redundanzfreiheit in der Relationalität tausender Tabellen und Relationen zwischen ihnen. Doch die Anwender wünschen sich vor allem Systeme, die es ihnen gestatten, Abläufe einfach und schnell zu verändern. Dies gelingt nur, wenn sich Prozesse flexibel gestalten lassen. Doch oft wird aber die Flexibilität mit noch mehr Komplexität erkauft. Einige Hersteller versuchen, über die Oberfläche für mehr Flexibilität zu sorgen. Nicht selten führt dies aber zu "funktionalen und gedanklichen Verrenkungen" des Anwenders.

Aus mangelndem Verständnis haben Technologen bisher meist auf Anforderungen reagiert, indem sie bereits komplexe, Strukturen auf der Grundlage relationaler Datenmodelle noch komplizierter gemacht haben.

Objektorientierte Datenbanken sind nicht automatisch proprietär

Eine provokante Hypothese lautet: Relationalität führt in die Sackgasse, einen Ausweg weisen objektorientierte Datenbanken. Die "Relationalisten" werden nun sofort argumentieren, dass das nicht stimmen kann, weil nur sehr wenige ERP-Systeme mit **objektorientierter Datenbank**[2] auf dem Markt vertreten waren oder sind. Außerdem gelten objektorientierte Datenbanken für sie als proprietär.

Mehr zum Thema ERP-Software

- **ERP-Vergleich, Teil 1: Vier SAP-Konkurrenten im Praxistest** [16]
- **ERP-Vergleich, Teil 2: SAP, IFS, Infor und Microsft im Test** [17]
- **ERP und Liquiditäts-Management**[18]
- **Was taugen ROI-Bewertungen von ERP-Software?** [19]
- **ERP-Zufriedenheitsstudie**[20]

Diese Sichtweise ist jedoch mehr als einseitig und beruht letztlich nur auf der Annahme, dass alles, was nicht relational und nicht mit **SQL**[3] bearbeitbar ist, proprietär zu sein hat.

Objekt-relationale Mapper

Ketzerisch gesprochen liegen die Ursachen dafür im relationalen Modell, das auf permanenter Neubildung von Relationen beruht. Jede neu zu berücksichtigende Besonderheit zieht neue Relationen nach sich, um die sich der Programmentwickler zu kümmern hat.

Mit objekt-relationalen (OR-) Mappern lassen sich Klassenmodelle in einer **relationalen Datenbank**[4] abbilden. Das ermöglicht die Koppelung von objektorientierter Programmier- und relationaler Datenbankwelt. Jedoch beschränken OR-Mapper die Performance und Flexibilität, weil sie die technische Komplexität allein schon durch ihren Einsatz erhöhen.

Darüber hinaus können relationale Datenbanktabellen nur eindimensionale Relationen abbilden. Für mehrdimensionale Relationen werden pro Relation jeweils entsprechend viele neue Tabellen benötigt. Damit wächst die Komplexität in der Lösung bis sie im Prinzip nicht mehr beherrschbar ist.

Das Objektmodell ist demgegenüber abstrakt und damit von Natur aus komplex. Es bildet keine Funktionen ab, sondern handhabt den Umgang mit Informationen.

Erforderlich ist eine holistische Sicht auf die Realität und damit auch auf das Modell, welches so zwangsläufig vollständig beschrieben sein muss. Denn eine Teilbeschreibung führt sofort zu Beschränkungen in der Anwendung.

Betriebswirtschaftliche Modelle

Die wesentliche Voraussetzung für die Modellentwicklung ist, dass sich ein betriebswirtschaftliches Modell unabhängig von seinem Einsatzziel entwickeln und effizient zusammenbauen lässt. Das verlangt nach einer semantischen Abbildung der Realität. Die jedoch scheint gegenwärtig im Zusammenhang mit **ERP-Systemen**[5] weiter denn je entfernt.

Worauf es bei den genannten Modellen ankommt, ist deren Granularität. Dabei ist nicht etwa das Möbelstück ein Objekt, sondern zum Beispiel die Spax-Schraube als kleinste Einheit. Denn je aggregierter das Objekt wird, desto weniger flexibel ist die ERP-Lösung.

Klassenmodelle in Datenbanken abbilden

Natürlich lässt sich jedes noch so beliebig komplexe Klassenmodell in einer relationalen Datenbank ablegen. Nur muss man dabei auch die Randbedingungen berücksichtigen, denn von einfachen SQL-Abfragen kann dann keine Rede mehr sein. Auch die eigentlich in der **Softwareentwicklung**[6] bevorzugte Unabhängigkeit der Daten von der Programmiersprache ist kaum mehr gegeben: Geschäftsdaten sind nur durch Bindung an die Methoden ihrer Klassen aussagekräftig.

[Hinweis auf Bildergalerie: **ERP im Mittelstand**] gal1

Enthalten Klassenmodelle Konstrukte, die ein direktes **relationales Mapping**[7] nicht erlauben, steigt der Aufwand, diese Daten in einer relationalen Datenbank zu speichern. Je mehr Daten assoziativ modelliert werden, das heißt nicht bloße Attribute eines Objekts sind, desto mehr werden Hilfstabellen für die Zuordnung der Daten untereinander benötigt.

Seit Jahren propagieren ERP-Hersteller, beliebig variable Funktionalität durch einfaches "Zusammenstecken" fertiger Software-Bausteine zu ermöglichen. Dieses Ziel bleibt bislang jedoch in weiter Ferne.

Wie funktionieren objektorientierte Datenbanken?

Wie eine objektorientierte Datenbank arbeitet, zeigt etwa das Beispiel "**ObjectStore**[8]" der Firma **Progress**[9]: Bei dieser Datenbank ist die Persistenz (Speicherung) eines Objekts nicht als Eigenschaft der Klasse definiert. Vielmehr wird ein Objekt bei seiner Instanziierung (Erstellung) als persistent oder transient festgelegt. Beispielsweise wird in der Programmiersprache C++ ein Objekt einer Klasse mit dem "new Operator" angelegt. Wenn diesem Operator ein "Persistenzparameter" übergeben wird, ist dieses Objekt damit sofort in der Datenbank gespeichert.

Eine transiente oder persistente Instanz eines gleichen Objekts unterscheidet sich dadurch, dass sich das transiente Objekt nur im Hauptspeicher des Rechners befindet, das persistente Objekt identisch in der Datenbank. Gleichzeitig lässt sich das persistente Objekt als transientes bearbeiten. Seine Attribute kann man durch einfache Wertzuweisungen direkt in der Datenbank verändern. Da sich das persistente Objekt in der Datenbank befindet, kann die Veränderung natürlich nur über eine Datenbank-Transaktion erfolgen. Am Ende der Transaktion sind die Veränderungen in der Datenbank gespeichert. Bricht die Transaktion ab, führt ein "Roll Back" das Objekt in seinen Ursprungszustand zurück.

Der Programmcode unterscheidet nicht, ob es sich um transiente oder persistente Objekte handelt. Einen expliziten Lese- oder Schreibvorgang gibt es nicht. Ein Objekt ist im Quellcode des Programms über eine nicht leere Variable verfügbar oder eben nicht. Ein weiterer entscheidender Unterschied besteht darin, dass das persistente Objekt in der Datenbank liegt und beliebig viele weitere persistente Objekte erzeugt werden können.

Nur, wie findet man diese Objekte wieder? Objekte werden nicht per se in Tabellen abgelegt. Jedes Objekt steht für sich als individuelle Einheit allein. Hier muss der **Entwickler**[10] selbst aktiv werden. Aber das ist ein nicht zu unterschätzender Vorteil.

Abbildung komplexer Klassenmodelle

- Eingebettete Klassen (zum Beispiel Zahlen-Klassen mit Einheiten für Mengen- oder Wertangaben)
- Eingebettete Objektrelationen, bei denen ein (Ursprungs-)Objekt auf beliebig viele (Gegen-)Objekte verweisen und jedes Gegen-Objekt auf beliebig viele Ursprungs-Objekte zurückverweisen kann (zum Beispiel zur Verknüpfung von Vorgänger- und Nachfolgerbelegen in Geschäftsprozessen)
- Polymorphe Relationen (Relationen auf verschieden vererbte Tabellen) und um eigene Eigenschaften erweiterbare Relationen (Wrapper)
- Gültigkeitseigenschaften von Objekten, OLAP-Datenwürfeln und so weiter

Die scheinbare Exotik objektorientierter Datenbanken löst sich auf, wenn man sie als "OR-Mapper mit integrierter Datenbank" oder als "Object Broker" bezeichnet.

Ein Anwendungsmodell kümmert sich nicht um die Speicherung eines Objekts, was letztlich als große Freiheit gelten darf: Zwar werden die Modelle dadurch abstrakter und komplexer, aber auf der Anwendungsseite führt das zu mehr Flexibilität. Metainformationen lassen sich dabei über Hierarchien vererben.

Beispielmodell Buchungskreis

Für die "eingebaute Funktionalität" heißt das, dass eine Unterscheidung von Buchungssätzen verschiedener Buchungskreise bei der Nutzung einer relationalen Datenbank in der Regel so modelliert wird, dass jeder Buchungssatz ein Attribut "Buchungskreis" erhält. Häufig hält man derartige, identifizierende Informationen nur in Parameter- oder Systemdateien vor. Beispielsweise kann Buchungskreis "1" etwa die GmbH & Co. KG einer Firma und Buchungskreis "2" die zugehörige Komplementär-GmbH sein. Der Buchungskreis hat keine weitere Eigenschaft als mittels Angabe der Nummer und Text die Trennung der Buchhaltung in zwei Untergruppen zu ermöglichen.

COMPUTERWOCHE-Ratgeber: ERP-Einführung

- **Einführungstrategie: Verbiegt man besser die Firma oder das ERP-System?** [11]
- **Mittelstand wrackt ERP-Software ab**[12]
- **Böse Überraschungen bei der Auswahl vermeiden** [13]
- **ERP-Zufriedenheitsstudie**[14]
- **ERP-Modernisierung bei A.T.U., Reifen Ihle, Alfing Kessler und Gutex** [15]

Es erscheint durchaus lohnenswert, einen Schritt weiter zu gehen und im Modell eine übergeordnete Systematik zu entwickeln, die die Buchungskreise verwaltet. Ein Buchungskreis ist dann zum Beispiel ein separates, ordnendes Objekt, das als eine seiner Eigenschaften seine Buchungssätze aktiv verwaltet. Derartige Buchungskreis-Objekte lassen sich auch nachträglich einbauen. Außerdem können mehrere parallel arbeitende Buchungskreise existieren. In einer objektorientierten Datenbank kann jedes Buchungskreis-Objekt direkt eine Liste mit (Referenzen zu) seinen Buchungssätzen halten. Es verhält sich so, als ob jedes Buchungskreis-Objekt eine eigene Tabelle nur seiner Buchungssätze definieren würde. Aber es ginge noch mehr: Jedes Buchungssatzobjekt kann auch mehr als einem Buchungskreisobjekt zugeordnet sein. Dadurch wird also eine Änderung des Anwendungsmodells durch Instanziierung weiterer Objekte erzeugt. Der Umbau des Modells und damit die Erweiterung der Funktionalität werden durch Neuorganisation der Beziehungen der Objekte untereinander erreicht. Die Klassen selbst ändern sich dabei nicht.

Individuell verwaltbare Geschäftsobjekte

Möglicherweise ist es nicht immer empfehlenswert, alle Objekte desselben Typs in einer einzigen großen Tabelle zu verwalten. Das spart nicht nur Attribute zur Unterscheidung oder besseren Zuordnung der Einträge, es hilft auch beim Verständnis des Gesamtmodells.

Die Tabellen relationaler Datenbanken können in einer objektorientierten Datenbank auf derartige, der Geschäftslogik folgenden Verwaltungsobjekte aufgeteilt werden.

Natürlich sind solche Beispiele auch anders darstellbar. Hier geht es jedoch um die Frage, ob die Nutzung einer relationalen Datenbank für ein ERP-System nicht nur zu einer bestimmten technischen Lösung führt, sondern auch zu einem bestimmten Verständnis der geforderten Anwendungslösung und andersherum. Beides scheint sich gegenseitig zu bedingen.

Die Vorstellung von individuell zu verwaltenden Geschäftsobjekten in einer objektorientierten Datenbank integriert die Rolle der Tabellen in die Modellierung. Die in einem relationalen System zu verwaltenden Tabellen gehen in das Modell der objektorientierten Datenbank über. Die objektorientierte Datenbank ist dann - sicherlich nicht nur im Vergleich zu einer relationalen Datenbank - keinesfalls mehr zu kompliziert.

Fazit

Fast alle ERP-Lösungen entfernen sich immer weiter von der Beherrschbarkeit der Komplexität, statt sich um den Anwender zu kümmern. Dabei gehört die Applikation nicht den Technologen, sondern ihren Benutzern. Nur wo bleiben die ERP-Programme mit "Komplexitätsabschaltung"? Auf die

Wirtschaftsinformatik wartet man hier vergebens. Von ihr gehen keinerlei Impulse aus. Anstatt die semantischen Modelle der Betriebswirtschaftslehre voranzutreiben, werden Technologen ausgebildet, die sich in Applikationen nicht auskennen. Die Fantasie und das Verständnis für Anwendungen fehlen ihnen. (fn)

Was unterscheidet relationale und objektorientierte Datenbanken?

Der **fundamentale Unterschied** zwischen relationaler und objektorientierter Datenbank-Philosophie besteht darin, dass die Relationen, die zeilenmäßige Abbildung in einem relationalen Datenbankmodell, nicht Bestandteil der Daten sind.

Metainformation eines Objekts in einer objektorientierten Datenbank wird in dem relationalen Datenbankmodell zu einer Globalinformation umgewandelt. Diese Global-Informationen beschreiben, wie **Geschäftsinformationen miteinander verdrahtet** sind, sind aber selbst nicht in den Daten hinterlegt.

Damit erstarrt die Beschreibung des relationalen Datenbankenmodells. Die Beziehungen sind hart verdrahtet. Das Modell bietet nur eine eingeschränkte, auf den Definitionsumfang begrenzte Flexibilität. Die Inhalte der mikro-relationalen Verbindung müssen am Objekt hängen, um Flexibilität in der Anwendung zu sichern.

Ein **Objekt ist eine eigenständige Instanz** einer Klasse und hat damit grundsätzlich eine größere Individualität als ein Datensatz (Record) innerhalb einer Tabelle. Ein Objekt wird nicht erst durch einen Primärschlüssel zu einem individuellen Objekt. Objekte können existieren, ohne in einer Tabelle (collection) registriert zu sein. Objektorientiert gedacht, ist das Vorhandensein eines Objekts in einer Gemeinschaft mit anderen (also in einer gemeinsamen Tabelle) bereits eine besondere Eigenschaft dieser Objekte (zum Beispiel die Liste der Standard-Skontobedingungen). Ein Objekt kann auch in mehr als nur einer "Gemeinschaft" geführt sein.

Solche Unterscheidungen erfordern in einem RDBMS immer ein (neues, zusätzliches) **Identifikationsfeld**, mittels eines Objektorienterten Datenbank-Management-Systems (OODBMS) kann dieses allein durch unterschiedliche Organisation von Objekten (Instanzen der gleichen Klasse) bewerkstelligt werden. "Mikro-relationale" Verbindungen sind also objektindividuelle Verbindungen.

Der Unterschied ist am zugrunde liegenden Modell festzustellen: Mit einer objektorientierten Datenbank erhält der Entwickler die Möglichkeit, ein Modell mit **höherem Abstraktionsgrad ohne relationale Beschränkung** und umfangreicherer Flexibilität zu entwickeln. Diese Chance muss der Hersteller eines ERP-Systems allerdings selbst nutzen.

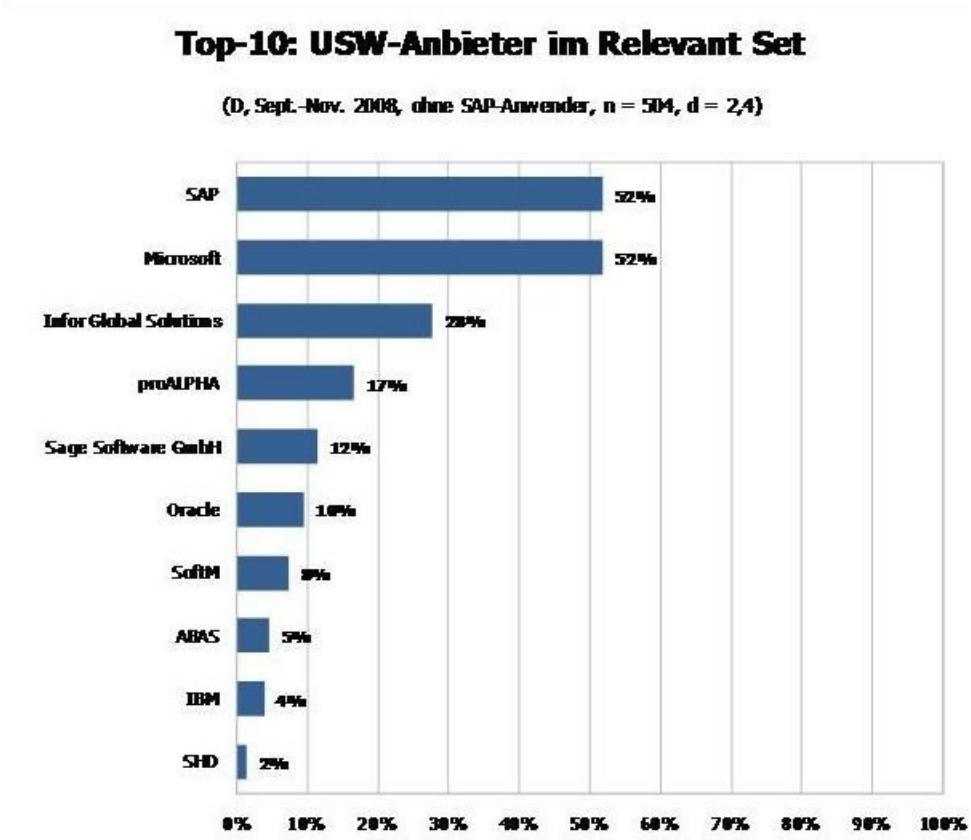
Beispiele für gut eingeführte ERP-Systeme mit objektorientierten Datenbanken sind Abas-ERP mit einer selbst entwickelten Datenbank oder CyberEnterprise, deren Applikation auf ObjektStore von Progress. Beide Systeme verfolgen aber unterschiedliche Realisierungskonzepte.

Links im Artikel:

1 **https://www.computerwoche.de/schwerpunkt/e/ERP-System.html**
2 **http://de.wikipedia.org/wiki/Objektdatenbank**
3 **http://de.wikipedia.org/wiki/SQL**
4 **http://de.wikipedia.org/wiki/Relationale_Datenbank**
5 **https://www.computerwoche.de/schwerpunkt/e/ERP-System.html**
6 **https://www.computerwoche.de/job_karriere/karriere_gehalt/1899389/**
7 **http://de.wikipedia.org/wiki/Objektrelationale_Abbildung**
8 **http://www.progress.com/objectstore/index.ssp**
9 **http://web.progress.com/index.html**
10 **https://www.computerwoche.de/job_karriere/karriere_gehalt/1899662/**
11 **https://www.computerwoche.de/knowledge_center/erp/1892251**
12 **https://www.computerwoche.de/knowledge_center/erp/1890779**
13 **https://www.computerwoche.de/knowledge_center/erp/1863035**
14 **https://www.computerwoche.de/knowledge_center/erp/1874362**
15 **https://www.computerwoche.de/knowledge_center/erp/1868459/index2.html**
16 **https://www.computerwoche.de/knowledge_center/erp/1894053/**
17 **https://www.computerwoche.de/knowledge_center/erp/1896160**
18 **https://www.computerwoche.de/knowledge_center/erp/1892314/**
19 **https://www.computerwoche.de/knowledge_center/erp/1884626/**
20 **https://www.computerwoche.de/knowledge_center/erp/1874362**

Bildergalerien im Artikel:

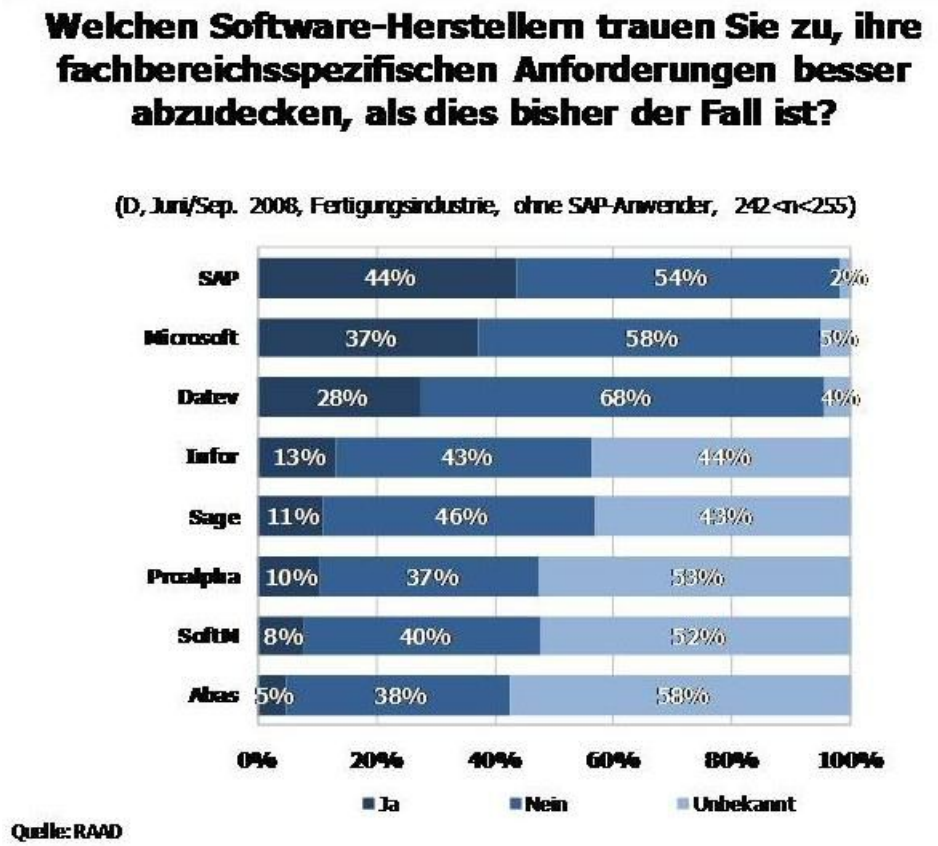
gal1 **ERP im Mittelstand**



Quelle: RAAD

Raad-Studie ERP im Mittelstand

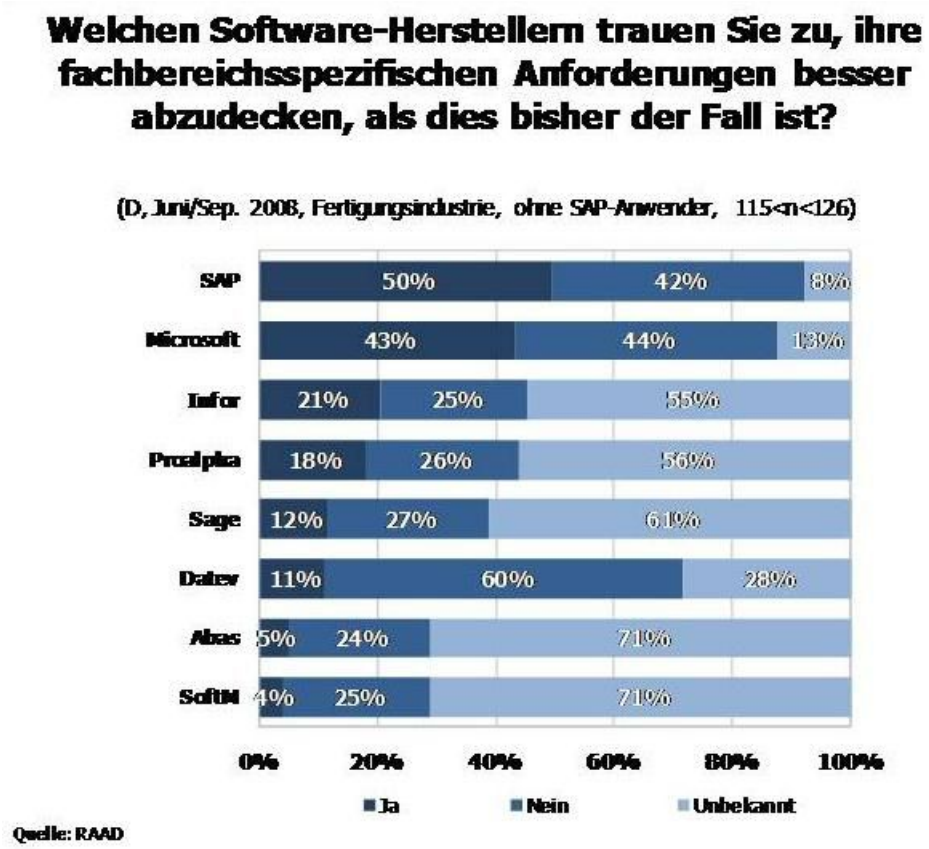
In der Raad-Studie wurden die Anwender unter anderem gefragt, welche Anbieter sie für relevant halten. SAP-Nutzer waren nicht darunter. Hier gehen offenbar Markenbekanntheit und tatsächliche Verbreitung deutlich auseinander.
Foto: RAAD Research



Quelle: RAAD

Raad-Studie ERP im Mittelstand

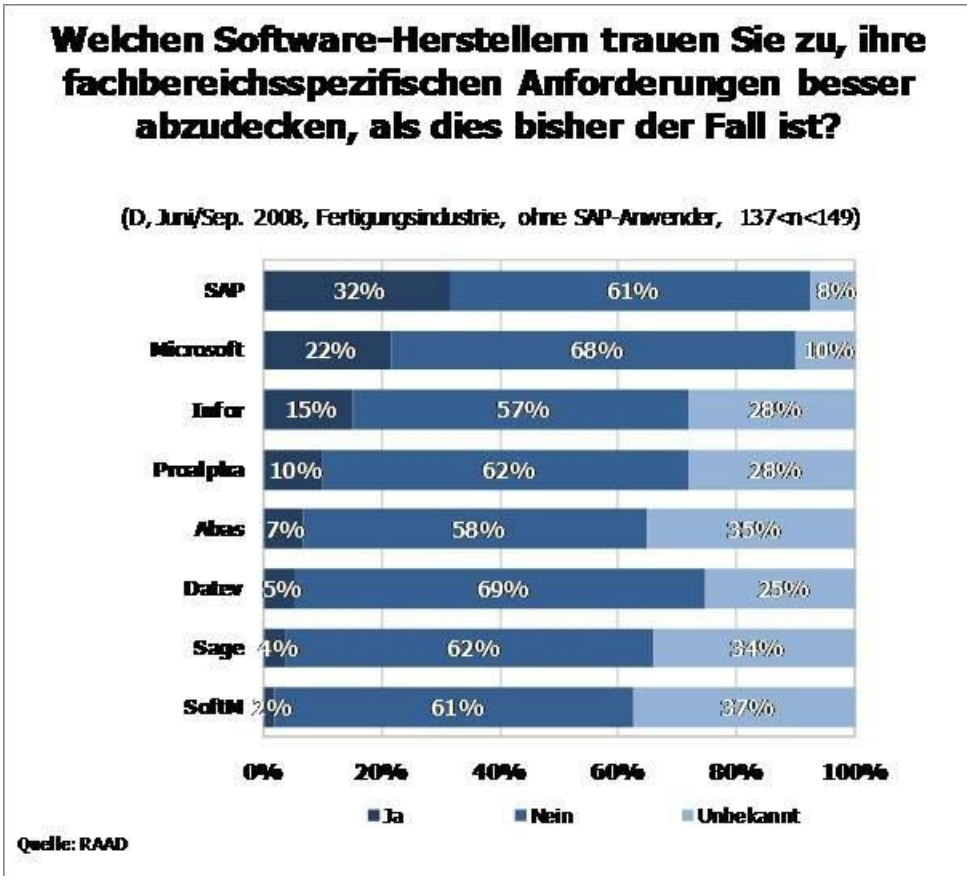
Die Fertigungsindustrie ist eine Kernbranche der ERP-Anbieter. Der SAP trauen die befragten Leiter des Finanzwesens/Controlling eine hohe fachspezifische Kompetenz zu. Microsoft kommt hier auf 37 Prozent. Lässt man die Microsoft-Kunden außen vor, sinkt die Quote auf 28 Prozent. Infor leidet Raad Research zufolge darunter, dass vielen Nutzern in den Fachabteilungen noch nicht klar geworden ist, dass die von ihnen genutzte Software (etwa das Rechnungswesen von Varial oder die auf Fertigung spezialisierte ERP-Lösungen „Baan“ beziehungsweise „ERP LN“) nun zu diesem Softwarekonzern gehören.
Foto: RAAD Research



Raad-Studie ERP im Mittelstand

Auch die Vertriebsleiter kennen vor allem SAP und Microsoft. Ihr Votum ähnelt dem der Finanzverantwortlichen im Unternehmen. Auch hier liegt Infor etwas hinten, da laut Raad Research den Firmenbereichen mit wenig Softwareherstellerbezug wie dem Vertrieb Infor als Markenname nur schwer zu vermitteln ist.

Foto: RAAD Research



Raad-Studie ERP im Mittelstand

Produktionsleiter beurteilen SAP und Microsoft ebenfalls gut. Sind aber deutlich skeptischer als ihre Vertriebs- und Finanzkollegen. Unklar bleibt dabei, welche bestehende Software die befragten Unternehmen einsetzen.

Foto: RAAD Research

IDG Tech Media GmbH
Alle Rechte vorbehalten. Jegliche Vervielfältigung oder Weiterverbreitung in jedem Medium in Teilen oder als Ganzes bedarf der schriftlichen Zustimmung der IDG Tech Media GmbH. dpa-Texte und Bilder sind urheberrechtlich geschützt und dürfen weder reproduziert noch wiederverwendet oder für gewerbliche Zwecke verwendet werden. Für den Fall, dass auf dieser Webseite unzutreffende Informationen veröffentlicht oder in Programmen oder Datenbanken Fehler enthalten sein sollten, kommt eine Haftung nur bei grober Fahrlässigkeit des Verlages oder seiner Mitarbeiter in Betracht. Die Redaktion übernimmt keine Haftung für unverlangt eingesandte Manuskripte, Fotos und Illustrationen. Für Inhalte externer Seiten, auf die von dieser Webseite aus gelinkt wird, übernimmt die IDG Tech Media GmbH keine Verantwortung.