Prof. Dr. Andreas Mielke
Institut für Theoretische Physik
Universität Heidelberg
Philosophenweg 19
69120 Heidelberg
Germany

http://www.tphys.uni-heidelberg.de/~mielke/
e-mail: mielke@tphys.uni-heidelberg.de

SAP für Physiker

Andreas Mielke

Sommersemester 2012
Zeit: Dienstag, 9-11
Ort: Philosophenweg 12, kHs

Dieses Script wird fortlaufend ergänzt.

Abstract

SAP ist der weltweit führende Softwarehersteller für Software zur Unternehmenssteuerung, der größte deutsche Softwareanbieter und einer der größten Arbeitgeber für Physiker in Deutschland.

Software zur Unternehmenssteuerung (Enterprise Resource Planning, ERP) wird in allen großen Unternehmen eingesetzt. Die Vorlesung bietet eine Einführung in ERP-Systeme auf einem hohen Niveau. Ziel der Vorlesung ist es, die allgemeine Struktur solcher Systeme, die Hard- und Softwarearchitektur, die Betriebsmodelle, Supportstrukturen, Nutzungs- und Qualitätskriterien, etc. verständlich zu machen. Unter anderem werden folgende Fragen diskutiert:

Was ist ein ERP-System?

Wofür werden ERP-Systeme in Unternehmen eingesetzt?

Wie sind SAP-Systeme softwaretechnisch aufgebaut?

Auf welchen Hardware-Architekturen laufen SAP-Systeme?

Wie werden SAP-Systeme genutzt?

Wie werden SAP-Systeme betrieben?

Was kostet es, SAP-Systeme zu betreiben?

Wie beurteilt man die Qualität eines SAP-Systems?

Wie lassen sich SAP-Systeme oder einzelnen Aspekte solcher Systeme modellieren?

Wie kann man diese Modelle in der Praxis anwenden?

Als Hörer der Vorlesung gewinnen Sie so einen allgemeinen Überblick und damit ein globales Verständnis für solche Systeme. Es geht also nicht um den Blick in tiefe Details, auch nicht um einen Programmierkurs für SAP, sondern darum, solche Systeme 'durch die Brille eines Physikers' zu betrachten, zu verstehen, zu modellieren.

Interessentenkreis, Literatur

Die Vorlesung kann von Studenten nahezu aller Semester besucht werden. Voraussetzung ist Interesse an diesem Thema und die Bereitschaft, etwas Zeit zu investieren, um sich mit dem Thema zu beschäftigen. In dem Bereich der Vorlesung, in der Aspekte der Modellierung behandelt werden, sind mathematische Kenntnisse aus verschiedenen Gebieten (Wahrscheinlichkeitstheorie und Statistik, Graphentheorie) nützlich, aber werden nicht vorausgesetzt.

Literatur zu dieser breit angelegten Vorlesung gibt es nicht. Fachliteratur im Bereich SAP behandelt immer nur sehr spezielle Aspekte von SAP-Systemen.

Hinweise zu diesem Text

Das Manuscript ist in unterschiedlichen Formaten verfügbar. Ich empfehle die Verwendung der pdf-Version und den Acrobat Reader, da an vielen Stellen von den zusätzlichen Möglichkeiten des pdf Gebrauch gemacht wird. Beispielsweise sind alle Verweise innerhalb des Textes aktive Verknüpfungen. Links für Zitate sind dunkelrot, Links auf Textstellen dunkelblau, Links auf http-Adressen dunkelgrün eingefärbt. Diese können sie nutzen, wenn Sie den Acrobat Reader so konfigurieren, daß bei Verweisen auf externe Quellen ein WWW-Browser die entsprechenden Seiten anzeigt.

Das Manuscript wird fortlaufend erweitert und ergänzt.

Inhaltsverzeichnis

1 Einführung

SAP-Systeme werden bei allen großen Unternehmen in Deutschland zur Steuerung der Geschäftsprozesse eingesetzt. Allgemeiner nennt man solche Systeme ERP (Enterprise Resource Planning) Systeme. SAP hat in diesem Markt nach eigener Darstellung weltweit einen Marktanteil von mehr als 50%, in Europa und speziell in Deutschland liegt der Marktanteil deutlich höher. Größter Wettbewerber ist Oracle. Andere Wettbewerber sind vor allem im Marktsegment der kleinen und mittleren Unternehmen unterwegs (Microsoft, Navision) und sind auf enge Märkte begrenzt, z.B. auf bestimmte Regionen oder Branchen, oder sind Spezialanbieter für spezielle Softwareprodukte (z.B. salesforce.com). Alle DAX-Unternehmen sind Kunden von SAP.

SAP ist das größte Softwareunternehmen in Europa und damit ein bedeutender Arbeitgeber, gerade auch für Physiker. Wichtige Kennzahlen sind in Tabelle 1 zusammengestellt.

Umsatzerlöse	14.233 Mio. Euro (Geschäftsjahr 2011)
Software- und softwarebezogene Serviceerlöse	11.319 Mio. Euro (Geschäftsjahr 2011)
Betriebsergebnis	4.881 Mio. Euro (Geschäftsjahr 2011)
Mitarbeiter	55.765 (Stand: 31. Dezember 2011)
Kunden	Über 82.000 in mehr als 120 Ländern
Partner	Über 2.400 zertifizierte Partnerlösungen
Branchenlösungen	Über 25 (vom Bankensektor bis zu öffentlichen Verwaltungen)
Investitionen in Forschung und Entwicklung	1.939 Mio. Euro/14 % der Umsatzerlöse (Geschäftsjahr 2011)

Tabelle 1: SAP in Zahlen: Kennzahlen (US-GAAP). Quelle: http://www.sap.com/germany/about/investor/ueberblick/index.epx (24.3.2012)

Der Betrieb von SAP-Systemen verursacht bei Unternehmen einen großen Aufwand und damit hohe Kosten.

Die VMS AG, Heidelberg beschäftigt sich seit zehn Jahren mit der Vermessung, der Modellierung, den Betriebskosten von SAP Systemen mit dem Ziel, durch Vergleiche und Prognosen ihren Kunden Hinweise zu geben, wo es beim Betrieb der Systeme Optimierungsmöglichkeiten gibt, wie hoch das Einsparungspotential ist und mit welchen Maßnahmen es realisiert werden kann.

Abbildung 1 liefert einen kurzen Überblick über die VMS AG, Abbildung 2 zeigt die Kunden, für die wir bisher aktiv waren.

Ziel der Vorlesung ist es, ein Verständnis dafür zu entwicklen, was SAP Systeme sind, welche Bedeutung sie für Unternehmen haben, wie sie aufgebaut sind, wie sie betrieben werden, welche Kosten dafür anfallen, etc. Im letzten Teil der Vorlesung wird es dann auch um die Fragen gehen, die das Geschäftsmodell von VMS und die Vorgehensweise betreffen: Wie lassen sich

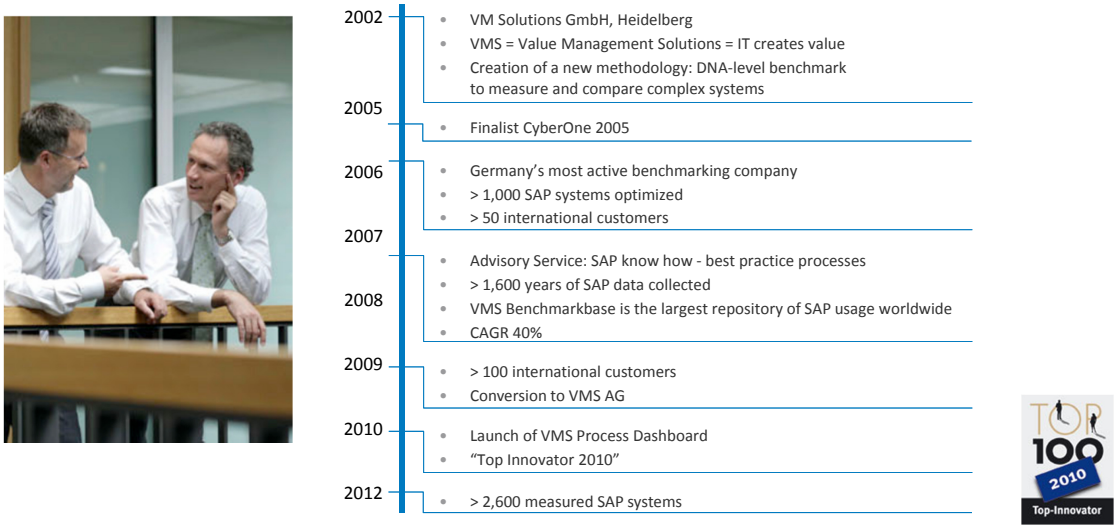

Abbildung 1: VMS AG

SAP-Systeme vermessen, wie lassen sie sich modellieren, wie kann man daraus Prognosen und Vergleiche machen, etc..

Die „SAP-Welt“ stellt einen eigenen Kosmos mit einer eigenen Sprache dar: Viele Mitarbeiter, die in Unternehmen mit SAP-Systemen intensiv zu tun haben, beschäftigen sich ihre komplette Berufstätigkeit mit den verschiedenen Aspekten dieser Systeme. Es gibt in jedem Unternehmen spezielle Abteilungen, die nur den Betrieb und die Pflege der SAP-Systeme als Auftrag haben. Es gibt vielfältige Veranstaltungen, die sich nur um SAP-Produkte drehen. Es gibt sehr viele kleine oder größere Firmen, die ausschließlich Produkte und Dienstleistungen im Zusammenhang mit SAP anbieten. Ein Ziel der Vorlesung ist es auch, die Begrifflichkeiten und damit die Sprache zu vermitteln, die in diesem Kosmos gesprochen wird.

2 Geschäftsprozesse und Informatisierung

Ziel dieses Kapitels ist es, ein grobes Verständnis für die Struktur von Abläufen in Unternehmen, von Geschäftsprozessen, zu vermitteln. Es geht dabei nicht um die möglichst vollständige Beschreibung eines Prozesses oder um eine möglichst umfassende Behandlung aller möglichen Prozesse, sondern darum, ein Gefühl dafür zu vermitteln, welche Aspekte für Geschäftsprozesse wichtig sind und welche Strukturen auftreten.

Geschäftsprozesse werden heute durch Informationssysteme elektronisch unterstützt. Man spricht von Informatisierung. Informatisierung führt zu Veränderungen in der Wirtschaft und ermöglicht erst die Globalisierung, von der heute viel gesprochen wird. Wichtige Aspekte von Informatisierung sind:

- Optimierung, Einsparung von Ressourcen.

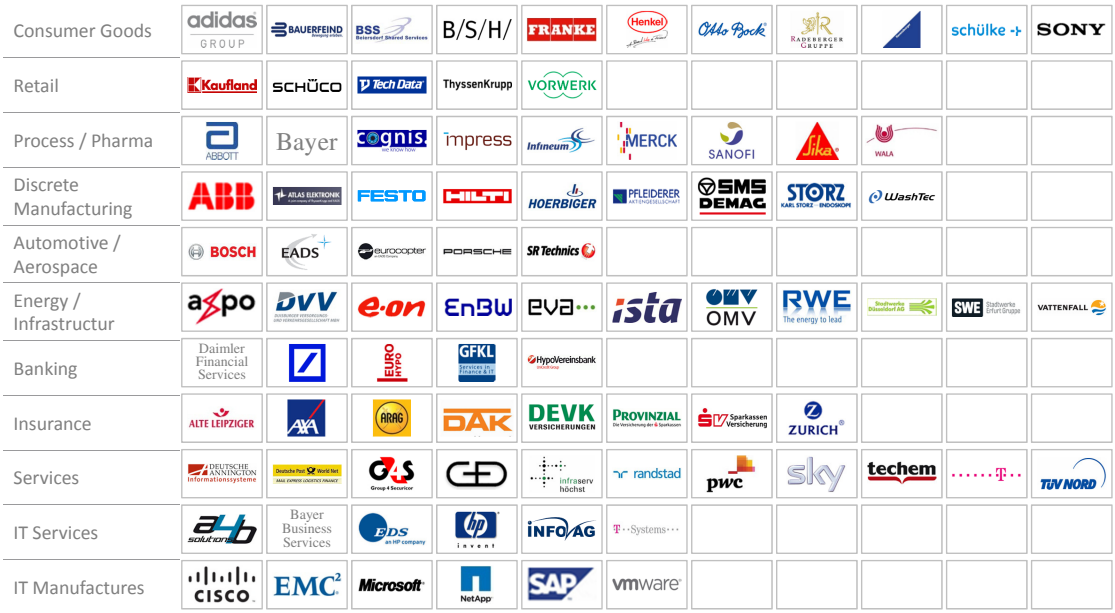

Abbildung 2: Kunden der VMS AG

- Neue Vertriebkanäle (Telefon, Internet, . . .).
- Unternehmensübergreifende Prozesse.

  Beispiel im Bereich Touristik: Reiseveranstalter, Fluggesellschaft, Hotels, Mietwagenfirmen, Kreditkartengesellschaften.
- Outsourcing: Ausgliederung von Unternehmensteilen; Entstehung von hoch spezialsierten Unternehmen, neuen Branchen, etc.
- Branchenübergreifende Zusammenschlüsse.

In Bezug auf Informatisierung geht es in diesem Abschnitt darum zu verstehen, welche Aspekte und Strukturen wichtig sind, wenn Geschäftsprozesse durch EDV abgebildet werden sollen.

2.1 Geschäftsprozesse

Was sind Geschäftsprozesse? Um die wesentlichen Strukturen von Geschäftsprozessen zu verständlich zu machen, versuche ich in diesem Abschnitt anhand eines stark vereinfachten Beispiels diese Strukturen zu veranschaulichen. Das Beispiel soll außerdem verdeutlichen, welche Vorteile ein Unternehmen durch die Einführung einer zentralen Softwarelösung für die Unterstützung von Geschäftsprozessen hat. Das Beispiel ist ein vereinfachter Beschaffungsprozesses bei dem Schreibwarengroßhandel Fritz Walter. Abbildung 3 zeigt die Struktur des Prozesses vor Einführung einer zentralen Softwarelösung.

In dem Beispiel werden zwar Prüfungen vorgenommen, aber es wird immer davon ausgegangen, daß die Prüfung positiv ausfällt. Warenrücksendungen, Reklamationen, etc. sind nicht enthalten.

Man sieht trotzdem, daß der Prozeß komplex ist. Es werden mehrere unterschiedliche Belege erzeugt, die konsistente Informationen enthalten müssen und die in unterschiedlichen Bereichen des Unternehmens relevant sind. Die Komplexität entsteht vor allem dadurch, daß in den drei Beteiligten Bereichen – Lager, Einkauf und Buchhaltung – mit jeweils eigenen Systemen (früher Papier, heute elektronisch) gearbeitet wird.

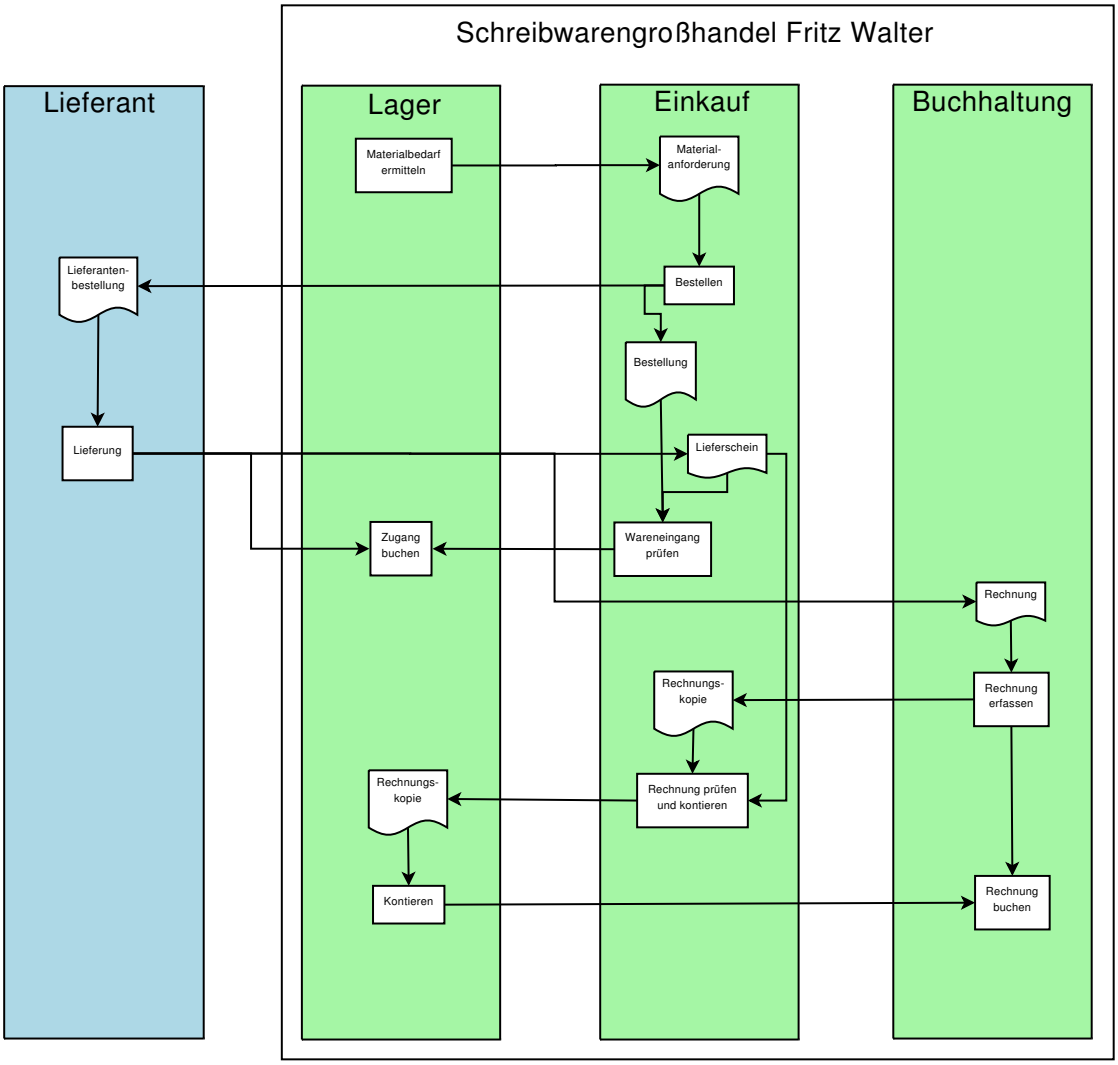

Abbildung 3: Vereinfachtes Beispiel für einen Beschaffungsprozeß

Der gleiche Vorgang kann auch in einem zentralen System realisiert werden. Das zentrale System eröffnet aber die Möglichkeit, Vereinfachungen und Verbesserungen vorzunehmen. Ziel dabei ist es unter anderem,

- die redunante Datenhaltung, im Beispiel die Rechnungskopien in Einkauf und Lager, zu vermeiden,

- durch einen gemeinsamen Datenbestand aufwendige Tätigkeiten zu vermeiden,
- klarere Zuständigkeiten zu erzielen.

Abbildung 4 zeigt den Prozess nach der Einführung eines zentralen Systems.

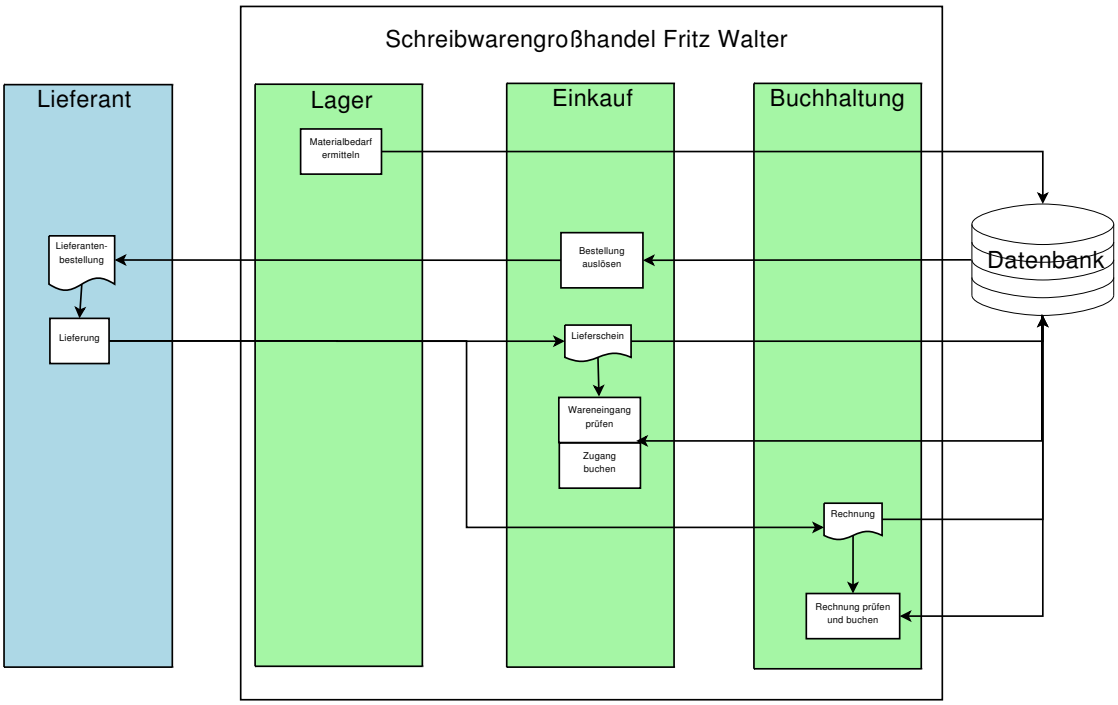

Abbildung 4: Vereinfachtes Beispiel nach Einführung eines zentralen Systems

Durch die zentrale Datenbank können alle Unternehmensbereiche auf die gleichen Daten zugreifen. Dadurch wird der Prozeß einfacher und sicherer. Die wichtigsten Verbesserungen sind

- Die nachträgliche Kontierung entfällt.
- Die Prüfung der sachlichen Richtigkeit (Wareneingang) und der rechnerischen Richtigkeit wird getrennt. Dadurch wird ein Vier-Augen-Prinzip angewandt, das die Sicherheit erhöht.
- Die Lagerbestände sind nach dem Zugang mengen- und wertmäßig sofort auf dem richtigen Stand.

Das Beispiel zeigt nur den Beschaffungsprozeß. In einem Unternehmen gibt es viele Bereiche mit jeweils unterschiedlichen Prozessen. Eine Liste mit Bereichen, in denen verschiedene Geschäftsprozesse in Unternehmen vorkommen, liefert Tabelle 2. Die Tabelle enthält für jeden Bereich eine Abkürzung. Diese Abkürzungen sind Standard in der SAP-Welt.

Schon das einfache Beispiel, das wir am Anfang diskutiert haben, zeigt, daß Geschäftsprozesse mehrere dieser Bereiche umfassen können. Für SAP-Systeme werden die Bereiche *Module* genannt.

	Finanzen und Controlling
FI	Finance
CO	Controlling
IM	Investment Management
	Bestellung und Produktion
LO	Logistics
MM	Materials Management
	Verkauf
SD	Sales and Distribution
CS	Customer Service
	Personal (HR)
PY	Payroll
PA	Personell Administration
PT	Personnel Time Management
	Weitere
PM	Plant Maintenance
PP	Production Planning
PS	Project System
QM	Quality Management
RE	Real Estate Management
EHS	Environment, Health, and Safety
PE	Event Management
IS	Industry Solutions

Tabelle 2: Klassifizierung der SAP Module, neuestes SAP Release.

2.2 Datenmodelle

Wir betrachten als Beispiel nochmals den Beschaffungsprozeß des Schreibwarengroßhandel Fritz Walter. Dort gibt es folgende Datentypen:

- Material
- Lieferant
- Bestellung
- Lieferung
- Rechnung

Daten, die für eine Lieferung gespeichert werden sollen, sind

- Datum, Lieferant, Material, Menge, Bestellung, Rechnung.

Die Frage ist, wie man man die Daten sinnvoll organisiert. Ziel dabei ist die Vermeidung von Redundanz, weil dadurch die Datenpflege erheblich vereinfacht wird und die Datensicherheit besser wird. Die Methode dazu ist Normalisierung. Ich folge hier der Darstellung in [1].

2.2.1 Terminologie

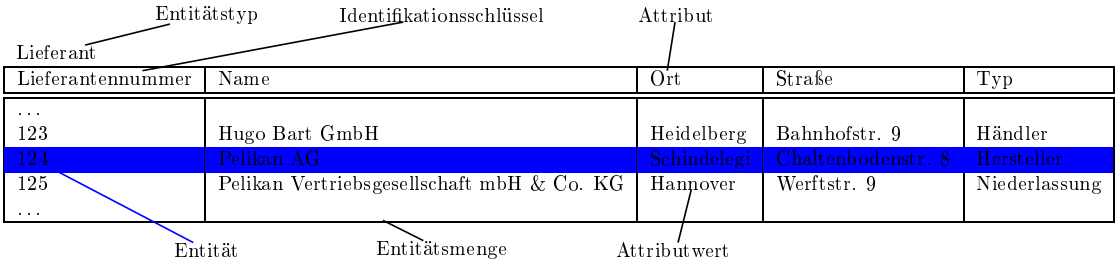

Abbildung 5: Terminologie für Datenmodell

Wir unterscheiden folgende Begriffe:

- Entität

  Eine Entität ist ein Objekt aus der realen Welt, das eindeutig von anderen Objekten, insbesondere von Objekten gleicher Art, unterschieden werden kann.

  Beispiel: Der Lieferant Pelikan ist eine Identität. Ein Material ist eine Identität. Im Fall des Lieferanten handelt es sich um ein konkretes Objekt, hier um eine Firma. Im Fall des Materials ist die Entität nicht ein konkreter Artikel (im Beispiel des Schreibwarengroßhandels ein Füllfederhalter), sondern die Menge aller konkreten Artikel (Beispiel: Füllfederhalter eines bestimmten Modells).

- Attribut

  Ein Attribut ist eine abstrakte Eigenschaft von Entitäten. Ein Attribut besitzt einen eindeutigen Namen.

  Beispiel: Firmenname oder Ort sind Attribute der Entität eines bestimmten Lieferanten.

- Attributwert

  Ein Attributwert ist eine konkrete Ausprägung eines Attributs einer Entität.

  Beispiel: Hamburg, Heidelberg, Zürich sind Attributwerte des Attributs Ort.

- Entitätstyp

  Ein Entitätstyp ist eine Klasse von Entitäten mit gleichen Attributen aber verschiedenen Attributswerten.

  Beispiel: Lieferant oder Material sind Entitätstypen.

- Entitätsmenge

  Die Menge aller Entitäten eines Entitätstyps heißt Entitätsmenge.

  Beispiele: Menge aller Lieferanten, Menge aller Materialien.

- Identifikationsschlüssel

  Der Identifikationsschlüssel erlaubt die eindeutige Identifikation einer Entität innerhalb eines Entitätstyps.

  Beispiel: Lieferantennummer. Es könnte mehrere Lieferanten Pelikan AG geben. Der Name allein ist also ein Attribut, aber erlaubt keine eindeutige Identifizierung des Lieferanten.

  Die Applikation (das SAP-System) stellt sicher, daß eine neue Entität, z.B. ein neuer Lieferant, automatisch einen neuen Identifikationsschlüssel bekommt.

Zwischen verschiedenen Entitäten gibt es Beziehungen. Beispielsweise gibt es eine Beziehung von Material zu Bestellung, von Bestellung zu Lieferant, von Material zu Lieferant. Die Beziehung zwischen Entitäten beschreibt man durch *Assoziationen*.

Eine *Assoziation* bezeichnet eine Klasse von Verweisen eines Entitätstyps auf einen anderen. Sie legt fest, wieviele Entitäten eines Entitätstyps einer Entität eines anderen Entitätstyps zugeordnet werden können.

Die Klassifikation von Assoziationen wird durch die *Kardinalitäten* 1, c, n, cn beschrieben. Jede Assoziation hat genau eine Kardinalität. Die Kardinalitäten haben folgende Bedeutungen:

1 Es wird genau eine Entität zugeordnet. Beispiel: Jede Bestellung geht an genau einen Lieferanten.

c Es wird eine oder keine Entität zugeordnet. Beispiel: Eine Bestellung wird an einen Handelsvertreter geschickt oder auch nicht.

n Es wird mindestens eine Entität zugeordnet. Beispiel: Eine Bestellung enthält mindestens ein Material.

cn Es werden eine, keine oder mehrere Entitäten zugeordnet. Beispiel: Ein Material kommt in einer, keiner oder mehreren Bestellungen vor.

Abbildung 6 zeigt eine typischen graphische Darstellung von Entitätstypen und zugehörigen Assoziationen.

Aus den Assoziationen ergibt sich, welche Identifikationsschlüssel Attribute eines anderen Entitätstyps sind. Beispielsweise erhält jede Bestellung als Attribut den Identifikationsschlüssel eines Lieferanten. Ein solches Attribut bezeichnet man als Fremdschlüssel.

Es gibt als Entitätstypen, Assoziationen und Attribute als abstrakte Klassen. Das Datenmodell eines Informaitionssystems, das diese beschreiben soll, legt genau fest

- welche Entitätstypen es gibt,
- welche Assoziationen es gibt,
- welche Attribute ein Entitätstyp besitzt,
- welche Klasse von Attributen eine Entität innerhalb einer Entitätsmenge eindeutig festlegen,

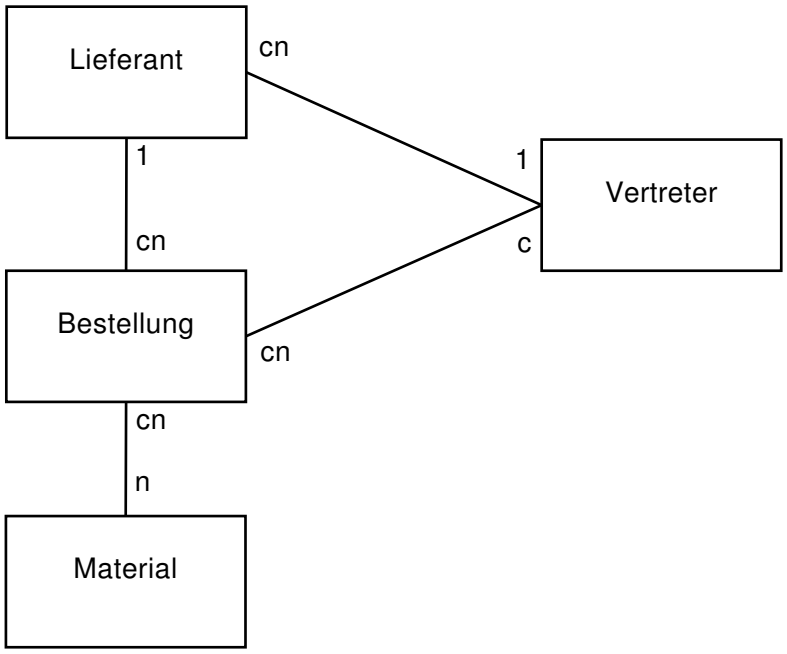

Abbildung 6: Beispiel für Assoziationen zwischen Entitäten (ER-Diagramm)

- wie Identifikationsschlüssel gebildet werden,
- welche Fremdschlüssel es für Entitätstypen gibt.

2.2.2 Normalisierung

Es ist möglich, den gleichen Sachverhalt in verschiedenen Datenmodellen zu modellieren. In der Informatik wurden Regeln entworfen, die bei dem Entwurf eines Datenmodells zwei Ziele verfolgen

1. Redundanzarmut

   Redundanz liegt vor, wenn die gleiche Information in den Daten mehrfach enthalten ist. Beispielsweise würde man die vollständige Adresse eines Lieferanten oder eines Vertreters als Attribute speichern. Es kann aber sein, daß ein Lieferant gleichzeitig Vertreter eines anderen Materials ist. Dann würden seine Adressinformationen mehrfach gespeichert.

   Redundanz führt immer dazu, daß es schwierig wird einen Datenbestand konsistent zu halten. Die Datenpflege wird teuer. Außerdem sind redundante Daten fehleranfällig.

2. Einfachheit

   Die Daten sollen einfach strukturiert sein, es sollte also möglichst wenig Entitätstypen, möglichst wenig Attribute, etc. geben.

Normalisierung ist eine Sammlung von Regeln, die dazu führen sollen, daß Datenbestände redundanzarm und einfach aufgebaut werden. Die Grundregel lautet:Dieselbe Information darf nur einmal in den Daten vorkommen. Im einzelnen werden folgende Regeln verwendet:

1. Ein Fremdschlüssel hat immer die Kardinalität c, cn oder n.

   Beispiel: Wir könnten in der Tabelle der Lieferanten bei jedem Lieferanten alle Bestellnummern von Bestellungen ablegen, die an diesen Lieferanten gegangen sind. Das würde zu einem komplexen Datenmodell führen, weil zu jedem Lieferanten unterschiedlich viele Bestellungen gehören. Also geht man umgekehrt vor und gibt jeder Bestellung als Attribut die Lieferantennummer mit. Das ist ein Fremdschlüssel.

2. Attribute mit einer inneren Struktur sollen nicht vorkommen, für diese wird ein neuer Entitätstyp eingeführt.

   Beispiel: Eine Bestellung enthält nomalerweise mehrere Bestellpositionen. Es gibt für eine Bestellung Daten, die für die gesamte Bestellung gelten (Lieferant, Bestelldatum, Besteller, etc.), und andere, die für einzelne Bestellpositionen gelten. (Artikelnummer, Menge, etc.) Man zerlegt also den Entitätstyp Bestellung in zwei Entitätstypen: Bestellkopf (enthält alle Daten für die gesamte Bestellung, Bestellnummer als Identifikationsschlüssel) und Bestellposition (enthält alle Daten für die einzelnen Positionen, die Bestellnummer als Fremdschlüssel).

3. Zwischen Entitätstypen sind nur einfache Beziehungen erlaubt (Assoziationen 1:n, 1:c, 1:cn). Andere Beziehungen werden durch Einführung eines neuen Entitätstyps aufgelöst.

   Beispiel: Die Beziehung Vertreter – Bestellkopf ist eine c:cn-Beziehung. Wenn wir in die Tabelle der Vertreter die Bestellungen aufnehmen, verstoßen wir gegen Regel 1. Also müssten wir die Vertreternummer in den Bestellkopf als Fremdschlüssel aufnehmen. Viele Bestellungen laufen aber nicht über Vertreter. Dann steht hier kein Wert; kein Wert wird in der Datenbank als NULL bezeichnet. Viele NULL-Werte in Fremdschlüsseln sind in einer Datenbank unerwünscht, sie führen zu erhöhten Anforderungen bei Abfragen. Deshalb löst man die c:cn-Beziehung auf, indem man einen neuen Entitätstyp Vertreterauftrag einführt. Dann gibt es eine 1:c-Beziehung Bestellkopf – Vertreterauftrag und eine cn:1-Beziehung Vertreterauftrag – Vertreter.

4. Direkte Rekursionen werden durch neue Entitätstypen aufgelöst.

   Beispiel: Verschiedene Materialien werden in Materialgruppen zusammengefasst. Materialgruppen können nochmals in Ober- und Untergruppen eingeteilt werden. Dadurch entstehen Rekursionen. Ein Material gehört einer Untergruppe an, diese dann einer Obergruppe. Solche Rekursionen haben mehrere Probleme: Je nach Material kann die Hierarchie mehr oder weniger Stufen haben und beim Löschen einer Untergruppe müssen die Materialien dieser Untergruppe an die Obergruppe gehängt werden. Solche Rekursionen sind deshalb zu vermeiden. Das kann durch neue Entitätstypen erreicht werden.

5. Identifikationsschlüssel müssen minimal sein.

   Identifikationsschlüssel identifizieren eine Entität eindeutig. Entitäten eines Entitätstyps unterscheiden sich durch ihre Attributswerte. Eine bestimmte Menge von Attributen er-

laubt es, zwei Entitäten zu unterscheiden. Also ist diese Menge von Attributen als Identifikationsschlüssel verwendbar. Dabei ist darauf zu achten, daß diese Menge minimal ist.

6. Vollständige Abhängigkeit vom Identifikationsschlüssel

   Alle Attribute eines Entitätstyps müssen vom gesamten Identifikationsschlüssel abhängen. Attribute, auf die das nicht zutrifft, müssen in einen anderen Entitätstyp aufgenommen werden.

   Beispiel: Man möchte für Bestellungen den zugesagten Liefertermin abspeichern. Der Liefertermin ist normalerweise Teil der Bestellposition. Der Lieferant wird aber nicht jede Bestellposition in einer Teillieferung liefern. Also führt man als neuen Entitätstyp die Teillieferung ein und der Liefertermin wird Attribut der Teillieferung.

7. Behandlung überlappender Entitätstypen

   Enthalten überlappende Entitätstypen gleiche Attribute, so werden diese in einem umfassenden Entitätstyp zusammengefasst.

   Beispiel: Lieferanten und Vertreter können überlappen. Beide haben das Attribut Adresse. Also legt man dafür einen neuen Entitätstyp an.

Diese Regeln sind nicht starr, im Einzelfall wird man davon abweichen. Sie gehen vor allem über das, was in der Theorie der Datenbanken als Normalform bezeichnet wird, hinaus.

Normalformen sind (hier nur der Vollständigkeit halber angegeben):

1. Normalform: Ein Entitätstyp ist in der ersten Normalform, wenn seine Attribute nur einfache Attributwerte ausweisen (Regel 2).
2. Normalform: Ein Entitätstyp ist in der zweiten Normalform, wenn er in der ersten Normalform ist und jedes nicht zum Identifikationsschlüssel gehörende Attribut voll vom Identifikationsschlüssel und nicht nur von einem Teil davon abhängig ist (Regel 6).
3. Normalform: Ein Entitätstyp befindet sich in der dritten Normalform, wenn er in der zweiten Normalform ist und kein Attribut über ein Nichtschlüssel-Attribut identifiziert werden kann. Entitätstypen in der dritten Normalform heißen normalisiert.

Folge der obigen sieben Regeln ist, daß es am Ende mehr Entitätstypen und mehr Assoziationen gibt, der einzelne Entitätstyp aber einfacher wird.

Neben diesen Regeln gibt es weitere, die die Erzeugung von Schlüsseln betreffen, Codetabellen, Historisierung von Daten, etc.

Änderungen im Datenbestand betreffen damit sehr häufig mehrere Entitäten unterschiedlicher Entitätstypen. Das hat zur Folge, daß Änderungen nur in konsistenter Weise vorgenommen werden dürfen. Das wird durch Transaktionen sichergestellt.

2.2.3 Andere Datenmodelle

Das Datenmodell, das in SAP verwendet wird, basiert im wesentlichen auf den oben dargestellten Ideen. Es gibt aber neue Produkte (z.B. Business by Design von SAP, dazu weiter unten),

die modernere Strukturen verwenden. Dazu gehören objektorientierte oder semantische Datenmodelle, die mit Klassen und Methoden arbeiten.

Objektorientierte Programmierung ist in SAP seit langem möglich, dazu sind aber nicht zwingend objektorierte Datenmodelle nötig.

2.3 Transaktionen

Wir haben bereits am Beispiel des einfachen Beschaffungsprozesses gesehen, daß ein Geschäftsprozeß aus einzelnen Schritten besteht. In einem einzelnen Schritt werden Daten erfasst, verändert oder gelesen, die inhaltlich zusammengehören. Einen solchen Schritt nennt man Transaktion.

Eine Transaktion ist ein computerisierter Arbeitsgang auf einer Datenbank.

In dem Beispiel des Beschaffungsprozesses (Abbildung 4) kommen folgende Transaktionen vor:

- Erzeuge Bestellung
- Erzeuge Warenbegleitschein
- Wareneingangsprüfung
- Zugang buchen
- Rechnungseingabe
- Rechnungsprüfung

In einem SAP-System gibt es sehr viele verschiedene Transaktionen. Tabelle 3 gibt zu jedem SAP-Modul an, wieviele verschiedene Transaktionen es im neuesten SAP-Release enthält. Die Zahlen gelten für ein Standard ERP-System. In speziellen Systemen gibt es andere Transaktionen.

Früher wurden Transaktionen in SAP durch ein Kombination von vier Buchstaben oder Zahlen bezeichnet. Diese Bezeichnungen kennt jeder, der mit SAP arbeitet, die meisten dieser Transaktionskürzel sind heute noch gültig. Die Nomenklatur war von der Form XXXn, wobei XXX Buchstaben oder Zahlen sind, n die Werte 1, 2 oder 3 annimmt. Dabei steht 1 für Erstellen, 2 für Ändern, 3 für Anzeigen. ME21 ist zum Beispiel 'Bestellung erzeugen', 'Create Purchase Order', ME22 und ME23 dienen zum Ändern und Ansehen von Bestellungen. Bestellungen können auch automatisch erzeugt oder geändert werden, es können Listen von Bestellungen angesehen werden, etc., für alle diese Aufgaben gibt es wieder eigene Transaktionen.

2.4 Analyse von Prozessen

2.4.1 Transaktionsanalysen

Da ein Prozeß aus einer festen Folge von Transaktionen besteht, und die Transaktionen im Rechner ablaufen, kann man im Prinzip zählen, welche Transaktion wie oft ausgeführt wird. Eine solche Analyse nennt man Transaktionsanalyse. Sie ist ein Werkzeug, mit Hilfe dessen in einem Unternehmen mögliche Probleme in Geschäftsprozessen gefunden werden können.

	Finanzen und Controlling	
FI	Finance	11.064
CO	Controlling	5.128
IM	Investment Management	4.895
	Bestellung und Produktion	
LO	Logistics	4.895
MM	Materials Management	2.709
	Verkauf	
SD	Sales and Distribution	2.576
CS	Customer Service	84
	Personal (HR)	
PY	Payroll	22.867
PA	Personell Administration	3.721
PT	Personnel Time Management	267
	Weitere	
PM	Plant Maintenance	588
PP	Production Planning	2.853
PS	Project System	592
QM	Quality Management	500
RE	Real Estate Management	1.870
EHS	Environment, Health, and Safety	1.290
PE	Event Management	53
IS	Industry Solutions	var.

Tabelle 3: SAP Module mit ihrer jeweiligen Anzahl Transaktionen, neuestes SAP Release.

Beispiel: Die Transaktion 'Rechnung erfassen' sollte genauso oft verwendet werden wie die Transaktion 'Rechnung prüfen'. Wenn 'Rechnung prüfen' weniger häufig vorkommt, werden offenbar nicht alle eingehenden Rechnungen geprüft. In diesem Fall gibt es vermutlich einen Fehler im Prozessablauf.

Beispiel: Die Transanktion 'Bestellung ändern' sollte deutlich weniger verwendet werden als die Transaktion 'Bestellung erfassen'. Wenn das nicht der Fall ist, gibt es möglicherweise Problem bei der Feststellung des Materialbedarfs oder Probleme im Einkauf.

Transaktionsanalysen erlauben auch, Klassifikationen vorzunehmen. Beispielsweise kann man zählen, wie oft für einen Lieferanten die Wareneingangskontrolle einen Fehler geliefert hat.

2.4.2 Kennzahlensysteme

Häufig werden heute für die Beurteilung von Prozessen Kennzahlen (KPIs, Key Performance Indicators) eingesetzt. Wir betrachten noch einmal das Beispiel Einkauf. Hier sind folgende Kennzahlen gebräuchlich:

Positionen pro Bestellung: Anzahl Positionen (Gesamt) / Anzahl Bestellungen

Anteil gelöschte Positionen: Anzahl gelöschte Positionen / Anzahl Positionen

Einkaufskosten pro Bestellung: Einkaufkosten / Anzahl Bestellungen

Einkaufskosten pro Position: Einkaufkosten / Anzahl Positionen

Bestellungen pro Einkäufer: Anzahl Bestellungen / FTE Einkauf

Positionen pro Einkäufer: Anzahl Positionen / FTE Einkauf

Lieferzeit: Lieferzeit dieser Positionen (Summe) / Gesamtzahl ausgewerteter Bestellpositionen (Anzahl)

Servicegrad: Bestellungen mit vollständiger Lieferung (Anzahl) / Gesamtzahl ausgewerteter Bestellpositionen (Anzahl)

Termintreue: Bestellungen mit termingerechter Lieferung (Anzahl) / Gesamtzahl ausgewerteter Bestellpositionen (Anzahl)

Erfüllungsgrad: Bestellungen mit vollständiger termingerechter Lieferung (Anzahl) / Gesamtzahl ausgewerteter Bestellpositionen (Anzahl)

Anteil elektronischer Bestellungen: Anzahl elektronischer Bestellungen / Anzahl Bestellungen gesamt

Alle diese Kennzahlen werden in Unternehmen heute standardmäßig zur Beurteilung des Einkaufsprozesses herangezogen. Dabei werden Kennzahlen intern verglichen (zwischen Teilkonzernen, Abteilungen, über die Zeit hinweg, zwischen Lieferanten, etc.) und unternehmensübergreifend (schwierig wegen uneinheitlicher Erfassung). Wichtig dabei ist, daß nicht naiv eine Kennzahl verwendet werden darf. Bei Unterschieden in einer Kennzahl können häufig andere Kennzahlen helfen, eine eventuell vorhandene Schwachstelle ausfindig zu machen.

An Kennzahlensystemen kann man (als Physiker) sehr einfach Kritikpunkte finden. Alle Kennzahlen sind einfache Quotienten. Dahinter steckt die naive Annahme, daß Größen proportional zueinander sind und deshalb ein Quotient einen direkten Vergleich ermöglicht. Das ist aber in der Regel nicht der Fall. Z.B. steigen die Einkaufkosten meist nicht proportional sondern unterproportional mit der Zahl der Bestellungen, weil sich in größeren Einkaufsabteilungen die Arbeit effizienter verteilen läßt. Ein zweites Problem ist, daß die Kennzahlen sehr häufig redundante Informationen enthalten. Bestellungen pro Einkäufer und Einkaufskosten pro Bestellung sind typischerweise invers proportional zueinander, weil Kosten proportional zur Zahl der Einkäufer ist.

Trotzdem haben sich solche Kennzahlensysteme etabliert und sie werden in Unternehmen und von Unternehmensberatungen relativ kritiklos verwendet.

3 SAP in Unternehmen

3.1 Das Unternehmen SAP.

Das Unternehmen SAP wurde 1972 gegründet. Kurzer Abriß der Geschichte:

1972 Gründung in Weinheim, Büro in Mannheim, Arbeitsplatz bei den Kunden, Arbeitszeit nachts. Ende des Jahres 9 Mitarbeiter, 620 kDM Umsatz.

1973 System RF, das erste Finanzbuchhaltungssystem. Geht später als ein Modul in R/1 auf. Plattform: Großrechner von IBM.

1974 Ende des Jahres 40 Kunden.

1975 Einkauf, Bestandsführung und die Rechnungsprüfung werden eingeführt (System RM, später Modul von R/1).

1976 Gründung der SAP GmbH Systeme, Anwendungen und Produkte in der Datenverarbeitung. 25 Mitarbeitern, Umsatz 3,81 Millionen DM.

1977 Firmensitz nach Walldorf, erste internationale Kunden.

1978 Anlagenbuchhaltung als neues Modul.

1979 Erster eigenen Rechner.
Technologiewechsel und Neukonzeption der SAP-Software: R/2

1980 Erste eigene Gebäude. Ein zweiter Rechner (4MB Hauptspeicher). Neues Modul Auftragsentwicklung RV.

1981 Erster Messeauftritt (systems München). 200 Kunden.

1982 Umsatz ca. 24 Millionen DM. Ein Gründungsmitglied der SAP scheidet aus.

1983 Expansion: Neue Gebäude, 125 Mitarbeiter, 41 Millionen DM Umsatz. Neues Modul Produktionsplanung und -steuerung (RM-PPS).

1984 48 neue Mitarbeiter.

1985 Jetzt gibt es vier Rechner mit insgesamt 64 MB Speicher.

1986 Erste Auslandsgesellschaften. 300 Mitarbeiter, damit werden Abteilungen geschaffen. Umsatz 100 Millionen. Modul Personalwirtschaft.

1987 Erste Ansätze für R/3.

1988 Umwandlung in AG. 940 Mitarbeiter. Umsatz von 245 Millionen DM. Jubiläum: Dow Chemicals ist der 1000ste Kunde.

1989 Neue bedienerfreundliche Oberfläche für das System R/2.
R/3 gewinnt Konturen: Plattform ist Unix, Client-Server-Modell. 85 Millionen (33% vom Umsatz) werden in die Entwicklung investiert.

1990 Übernahme von Steeb (50%) und CAS. 1.700 Mitarbeiter, Umsatzmarke 500 Millionen DM

1991 Sneak Preview: 1991 präsentiert die SAP erste Anwendungen des Systems R/3 auf der CeBIT in Hannover.

1992 Erste erfolgreiche Installationen von R/3 bei ausgewählten Kunden.

1993 beginnt die Zusammenarbeit der SAP mit Microsoft, dem größten Softwarehersteller der Welt. Ein Ziel der Vereinbarung ist die Portierung des Systems R/3 auf Windows NT. Freigabe erfolgt 1994. Schon vorher wurden alle Unix-Plattformen unterstützt.

1994 Umsatz 1,8 Milliarden DM, davon 34,3 Prozent aus Amerika. Am Jahresende 5.229 Mitarbeiter.

1995 Deutsche Telekom AG entscheidet sich für das System R/3; mit 30.000 SAP-R/3-Arbeitsplätzen.

1996 Erste Internet-Initiative der SAP.

1997 25 Jahre SAP. Erstmals mehr als 1 Millarde DM Gewinn: 1,6 Milliarden.

1998 Dietmar Hopp und Klaus Tschira verlassen den Vorstand. Hasso Plattner und Henning Kagermann sind Vorstandssprecher.

1999 mySAP.com verbindet E-Commerce-Lösungen mit den bestehenden ERP-Anwendungen.

2000 new economy: Internetmarktplätze und Portale. Partnerschaft mit Commerce One.

2001 Übernahme von Top Tier, Shai Agassi wird ein Jahr später zum Vorstandsmitglied. Internetblase platzt. Umsatz steigt um 17 Prozent auf 7,3 Milliarden Euro.

2002 29.000 Mitarbeiter Ende des Jahres. Vorstand verstärkt sich: Leo Apotheker wird Vertriebsvorstand.

2003 Hasso Plattner zieht sich aus dem Vorstand zurück.
mySAP.com (zwischenzeitlich mySAP Technology) heißt jetzt SAP-NetWeaver.

2004 24.000 Kunden in über 120 Ländern, rund 84.000 Installationen.

3.2 SAP Produkte

SAP R/3, die Basis der meisten SAP-Produkte, die heute auf dem Markt sind, wurde 1992 auf den Markt gebracht. Der Vorgänger war R/2, eine ERP-Software für den Einsatz auf Großrechnern. Vorgänger von R/2 war R/1, ein reines Finanzbuchhaltungssystem, das vorher RF hieß.

SAP R/3 ist eine monolythische ERP-Software. Die Module, die wir oben aufgelistet haben, sind zwar funktionell getrennte Einheiten, aber nicht architektonisch getrennt. Der Grund dafür ist einfach: Geschäftsprozesse gehen über Modulgrenzen hinweg und benötigen übergreifend einheitliche Datenbestände. Der größte Teil von R/3 ist in ABAP (Allgemeiner Berichts-Anwendungs-Prozessor, heute Advanced Business Application Programming) programmiert.

SAP R/3 wurde später in SAP ERP oder mySAP ERP umbenannt. Es wurde zu einem Teil der SAP Business Suite.

Seit einiger Zeit gibt es die SAP NetWeaver Plattform, auf der SAP ERP und andere SAP-Systeme (siehe unten) laufen. Diese Plattform erlaubt die Programmierung in Java. Allerdings ist

immer noch der überwiedgende Teil der SAP Programmierung in ABAP gehalten. Insbesondere die kundeneigenen Entwicklungen werden fast ausschließlich in ABAP entwickelt, weil bei den Firmen das know-how dazu vorhanden ist.

Neben ERP spielen die folgenden SAP-Systeme eine wichtige Rolle:

Customer Relationship Management (CRM) Ein CRM erlaubt die Steuerung aller Vorgänge, die mit Kunden und Vertrieb verbunden sind. Dazu gehören:

- Kundendatenbank
- Steuerung, Klassifizierung und Dokumentation aller Vertriebsaktivitäten.
- Datenversorgung und Kommunikation über mobile Endgeräte (Außendienstmitarbeiter).
- Customer Interaction Center (CIC), Kundenhotline, etc.

Business Information Warehouse (BI, BW) Ein BI ist ein zentrales Informationsystem. Hier werden alle Daten, die für Berichte in einem Unternehmen wichtig sind, gesammelt. Häufig haben Kunden ein zentrales BI System, fast ebenso häufig aber auch mehrere dezentrale BI Systeme zusätzlich für unterschiedliche Aufgaben. In der Regel findet dann trotzdem noch ein großer Teil des Reporting im ERP System statt.

Durch die Übernahme von Business Objects durch SAP ist dieser Bereich in einem Umbruch begriffen.

Enterprise Buyer (EBP) Der Enterprise Buyer ist ein eigenes System zur Abbildung des kompletten Beschaffungsprozesses für direkte und indirekte Materialien und Dienstleistungen über eine Internet-Lösung. Der Prozeß beginnt mit dem Anlegen des Einkaufswagens und endet mit dem Erfassen der Rechnung. Der Enterprise Buyer verkürzt den Beschaffungsprozeß und spart Unternehmen Kosten. Durch die Verlagerung der Routinearbeiten im Einkauf auf die Mitarbeiter der Fachabteilungen wird der Einkauf entlastet und kann sich strategischen Aufgaben wie Vertragsverhandlungen und Lieferantenanalysen widmen.

Enterprise Portal (EP) Heißt heute SAP NetWeaver Portal. Das EP ist die Unternehmensportal-Software der SAP AG. Ziel des Portals ist es, Mitarbeitern einen einheitlichen Zugriff auf unterschiedliche Systeme zu ermöglichen (Single Sign-On), eine personalisierte Darstellung, die Verwendung aller heute im Internet gängigen Möglichkeiten wie Knowledge Management & Collaboration (KMC), Dokumentenmanagement, Teamräume, Realtime Collaboration (z. B. Chat, Application Sharing), Integration diverser Groupware-Systeme, etc.

Advanced Planner and Optimiser (APO) Das APO ist eine Erweiterung des Supply Chain Managements (SCM). Es erlaubt die komplette Planung und Steuerung der Produktion. Wichtige Aspekte sind:

- Supply Chain Management (SCM)
- Supply Chain Monitoring
- Network Design
- Supply Network Planning
- Demand Planning
- Production Planning and Detailed Scheduling (PP/DS)
- Transportation Planning/Vehicle Scheduling

Teile dieser Funktionalität sind auch in einem ERP realisiert.

Exchange Infrastructure XI oder heute SAP Process Integration (SAP PI) zur zentralen Steuerung von Schnittstellen, siehe unten.

Weitere Systeme Daneben gibt es noch weitere Systeme.

3.3 Neuere SAP-Produkte

Neben den genannten Produkten versucht SAP seit längerem, schlankere und kostengünstigere Produkte für den Mittelstand zu entwickeln. Folgende Produkte gab und gibt es:

- Business One

  basiert auf der Software TopManage, die von der israelischen Firma TopManage Financial Solutions LTD entwickelt und vertireben wurde. Die Firma wurde 2001 von SAP übernommen, im Zuge dessen wurde Shai Agassi zeitweise Vorstandsmitglied bei SAP. Business One ist weiterhin im Einsatz, aht sich aber gegen Konkurrenzprodukte wie Sage oder Navision (jetzt Microsoft Dynamics NAV) nicht durchgesetzt.

- Business All-in-One

  Hierbei handelt es sich um ein kleines, vorkonfiguriertes ERP System für größere Mittelständler oder Tochterunternehmen großer Konzerne. Im Gegensatz zu Business One ist es kompatibel zu SAP ERP. Es hat sich aber gegenüber SAP ERP nicht durchgesetzt.

- Business ByDesign

  Business ByDesign ist eine vollständig neu entwickelte Lösung für den Mittelstand, die 2007 vorgestellt wurde. Sie verwendet ein objektorientiertes Datenmodell. Die letzte Version, die vertrieben wurde, ist aber noch recht problembehaftet. Das Hauptproblem ist, daß bei der Entwicklung kaum ein Kontakt zu Kunden gesucht wurde (im Gegensatz zu der Entwicklung von R/1, R/2 oder R/3), so daß viele Probleme erst im produktiven Einsatz deutlich wurden. Gerüchteweise wird jetzt versucht, die Probleme durch ein komplettes Redesign zu lösen.

3.4 SAP-Landschaften

In einer Landschaft sind verschiedene Systeme miteinander durch Schnittstellen verbunden. Schnittstellen gibt es auch nach außen, zu Kunden und Lieferanten (siehe Abbildung 7).

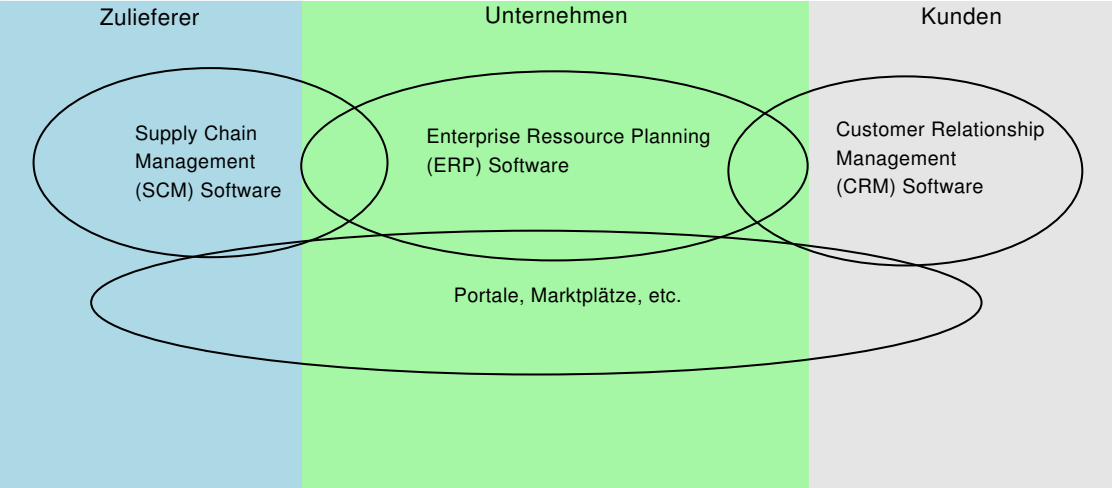

Abbildung 7: Verzahnung von Unternehmen mit Kunden und Lieferanten und die Abbildung in Software.

Dabei unterscheidet sich die Art der Schnittstellen. Wesentliche Unterschiede bestehen zwischen B2B (Business to Business, Komminikation zwischen Unternehmen) und B2C (Business to Consumer, Kommunikation zwischen Unternehmen und Endkunde). B2B Schnittstellen sind in der Regel wenige, aber komplexe und stark individualiserte Schnittstellen. B2C Schnittstellen sind viele, aber standardisierte Schnittstellen. In beiden Fällen ist neben der elektronischen Schnittstelle die persönliche Schnittstelle wichtig.

Abbildung 8 zeigt ein Beispiel für eine SAP-Landschaft in einem kleineren oder mittleren Unternehmen oder in einer Tochterfirma eines Konzerns.

Benutzer, die intensiv mit den Systemen arbeiten, melden sich über das SAP-GUI auf den Systemen an.

- Vorteile: Schnelle Anbindung, flexibel, jede Funktionalität steht zur Verfügung.
- Nachteile: Aufwendig und teuer bei Installation und Betrieb.

Benutzer, die die Systeme nur gelegentlich nutzen (e.g. Zeiterfassung, Urlaubsanträge, Raumbuchungen), verwenden das System über einen Web-Browser.

- Vorteile: Kein Installationsaufwand.
- Nachteile: Alle Ein- und Ausgabemasken müssen vom Server in HTML umgesetzt werden.

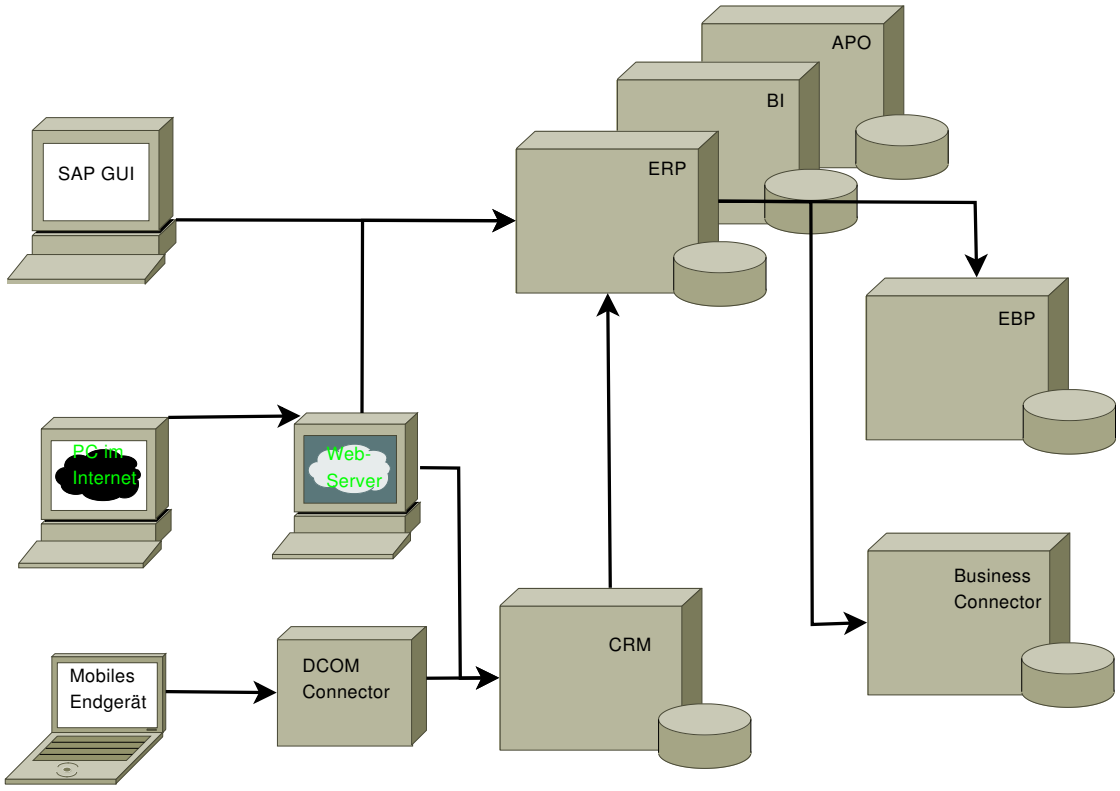

Abbildung 8: Beispiel für eine SAP-Landschaft mit mehreren Systemen.

Zusätzlich gibt es den Zugriff über mobile Endgeräte für Außendienstmitarbeiter. Dazu ist ein spezieller Server (Middleware) notwendig. Hier erfolgt der Zugriff asynchron.

Die Verknüpfung vieler Systeme über Schnittstellen widerspricht eigentlich der ursprünglichen Idee eines zentralen Systems zur Unternehmenssteuerung. Folgende Punkte sind zu beachten:

- Da die Systeme jeweils eigene Datenbanken haben, wird es extrem schwierig, redundate Datenhaltung zu vermeiden. Das sollte weiterhin das Ziel sein.
- Da Daten in verschiedenen Systemen zueinander in Bezug stehen, müssen Backupverfahren entsprechend aufgesetzt werden.
- Verschiedene Systeme lassen sich nicht auf beliebigen Plattformen betreiben. Plattformplanung wird wichtig. Man handelt sich unter Umständen das Problem ein, daß man viele verschiedene Plattformen betreiben muß. Probleme sind hierbei Know-How-Beschaffung und Kosten.
- Mit der Zahl der Systeme steigt die Zahl der Schnittstellen. Pflege von Schnittstellen kann aufwendig sein. Außerdem konkurieren Aufrufe, die über eine Schnittstelle kommen, mit dem Dialogbetrieb in einem System. Partielle Lösung: Ein XI-System.
- In hoch integrierten Landschaften sind Performanceprobleme schwerer zu lokalisieren.

- Durch die starke Spezialisierung ist es schwierig, Personal zu bekommen, das mit integrierten Landschaften kompetent umgehen kann.

3.5 Typische Zahlen zu Kosten, Nutzung und Qualität von SAP-Landschaften

Tabelle 4 stellt typische Zahlen für ein einzelnes ERP-System zusammen.

Nutzung	
Benutzer	100 - 50.000
Dialogschritte pro Tag	10.000 - 5 Mio.
Verschiedene genutzte Transaktionen pro Tag	200 - 3.000
Infrastruktur	
Nicht-produktive Systeme	1 - 5
Server	1 - 20
CPUs	4 - 100
Plattenplatz	300 GB - 10 TB
Qualität	
Antwortzeit pro Dialogschritt	0,3 - 2 s
Betriebszeit	5×12 - 7 $\times24$ h/week
Verfügbarkeit	98% - 99,8%

Tabelle 4: Typische Daten für ein SAP Produktionssystem

Systeme	
SAP Produktionssysteme	3-300
Personal	
Intern	10 - 5.000
Extern	10 - 5.000
Kosten	
Betriebskosten p.a.	10 Mio. - 1 Mia.EUR
Projektkosten p.a.	3 Mio. - 200 Mio. EUR
Veränderung	-20% - +5%

Tabelle 5: Typische Daten für eine SAP-Landschaft

Tabelle 5 stellt typische Zahlen für eine SAP-Landschaft zusammen, so wie sie in vielen ver-

schiedenen Unternehmen heute anzutreffen ist.

Sehr häufig werden solche Zahlen, oder Quotienten solcher Zahlen als Kennzahlen (KPIs) zur Steuerung der IT verwendet. Typische Kennzahlen sind z.B. SAP Kosten pro Benutzer (typisch 50 EUR/Monat bis 200 EUR/Monat), IT-Kostenanteil am Umsatz (typisch 1% bis 3%). Solche Kennzahlen liefern aber bestenfalls einen groben Überblick, eine seriöse Steuerung oder Vergleiche sind damit nicht sinnvoll, werden aber gemacht.

4 Software-Architektur

4.1 Begriffe

Rechner: Eine Maschine mit CPU, Hauptspeicher, Netzwerkkarte, etc.

Server: Ein Programm oder ein System von Programmen, das auf einem Rechner läuft und einer bestimmten Aufgabe dient. Es bedient Anfragen von anderen Programmen (Clients), die es über Schnittstellen ansprechen.

Client: Ein Programm, das auf einem Rechner läuft und Anfragen an einen Server stellt.

GUI: Graphische Schnittstelle, die von einem Benutzer bedient wird.

Datenbank: Eine Datenbasis, in der Daten in organisierter Form abgelegt sind.

Datenbanksystem: Ein Programm, das Daten in einer Datenbank ablegt, wieder ausliest, ändert, anderen Programmen zur Verfügung stellt.

Datenbankserver: Eine konkrete Installation einer Datenbank und eines Datenbanksystems. Der Begriff wird mehrdeutig verwendet und bezeichnet auch den Rechner, auf dem diese Installation läuft.

Applikationsserver: Ein System von Programmen (Applikation), die Anfragen von Clients bedienen, Daten zur Verfügung stellen, etc., aber nicht über eine eigene Datenbank verfügen, sondern die Daten als Client von einem Datenbankserver beziehen.

SAP-System: Ein SAP-System besteht aus einer Datenbank, in der Regel einem Datenbankserver, einem oder mehreren Applikationsservern, Clients mit SAP-GUI und ggf. weiteren Servern.

4.2 Client-Server

Erstes und wesentliches Architekturmerkmal eines SAP R/3 Systems ist die Client-Server-Architekur. Abbildung 9 zeigt eine Skize der einfachsten Client-Server Struktur eines SAP-Systems.

Diese Struktur wurde durch R/3 realisiert. Wichtiger Punkt einer solchen Architektur ist die Skalierbarkeit. Es kann mehrere SAP Applikationsserver geben, sie können auf einen oder mehrere Rechner verteilt werden. Das gleiche gilt natürlich auch für die Applikationsebene. Im Prinzip

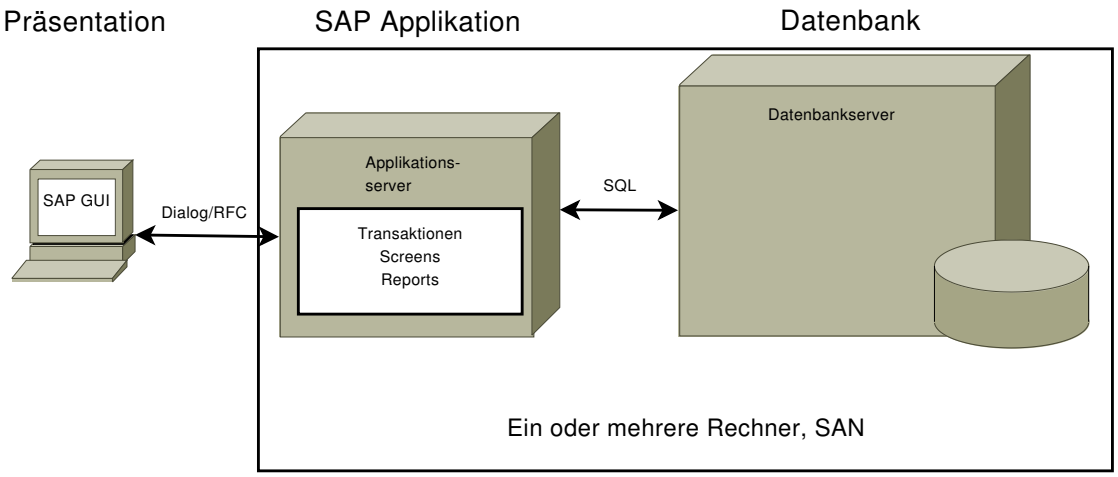

Abbildung 9: Einfache Client-Server Architektur

können bei einigen Datenbankprodukten auch mehrere Instanzen für Datenbankserver verwendet werden, diese Möglichkeit wird in der Praxis aber nur extrem selten verwendet.

Das Client-Server-Konzept hat sich heute in vielen Bereichen durchgesetzt. Daneben gibt es vermehrt Peer-to-Peer Netzwerke, in denen Rechner gleichwertige Aufgaben verrichten und es keine dedizierten Server und Clients gibt.

In modernen SAP-Systemen oder -Landschaften gibt es komplexere Client-Server-Beziehungen, ein Beispiel zeigt Abbildung 10. Auch hier gilt das Prinzip der Skalierbarkeit. Alle Komponenten bis auf den Datenbankserver kommen in großen Installationen mehrfach vor. Gewisse Abläufe, insbesondere im Datenbankserver (Sperren), können aber nicht auf mehrere Instanzen verteilt werden, dazu unten mehr.

Der Zugriff auf die Systeme durch den Benutzer ist die erste Client-Server Ebene. Für den Benutzer ist das Programm, mit dem er auf ein SAP-System zugreift (SAP-GUI, Browser) der Server, an den er als Client seine Anfragen schickt.

Die Präsentationsschicht (SAP-GUI, Browser) kommuniziert entweder direkt oder über zwischengeschaltete Server mit dem SAP Application Server.

Der SAP-GUI kommuniziert direkt mit dem SAP Application Server. Das ist der meist verwendetste Zugriff und der älteste. Er ist so konzipiert, daß der Datenaustausch zwischen Client und Server möglichst klein ist.

Der Zugriff über Browser kann auf verschiedene Arten erfolgen:

- ITS (Internet Transaction Server) und Webserver für den Webzugriff über HTML. Anwendungsgebiete: Zugriff von Gelegenheitsnutzern (Zeiterfassung, Reiseanträge und -kostenabrechnung, Urlaubsanträge, etc.), B2B-Bestellungen, Internet-Vertrieb, etc.
- BSP (Business Server Pages): Hier werden mit ABAP HTML-Seiten und Javascript dynamisch generiert. Ein gesonderter Webserver ist nicht nötig, wird aber zur Sicherheit empfohlen.

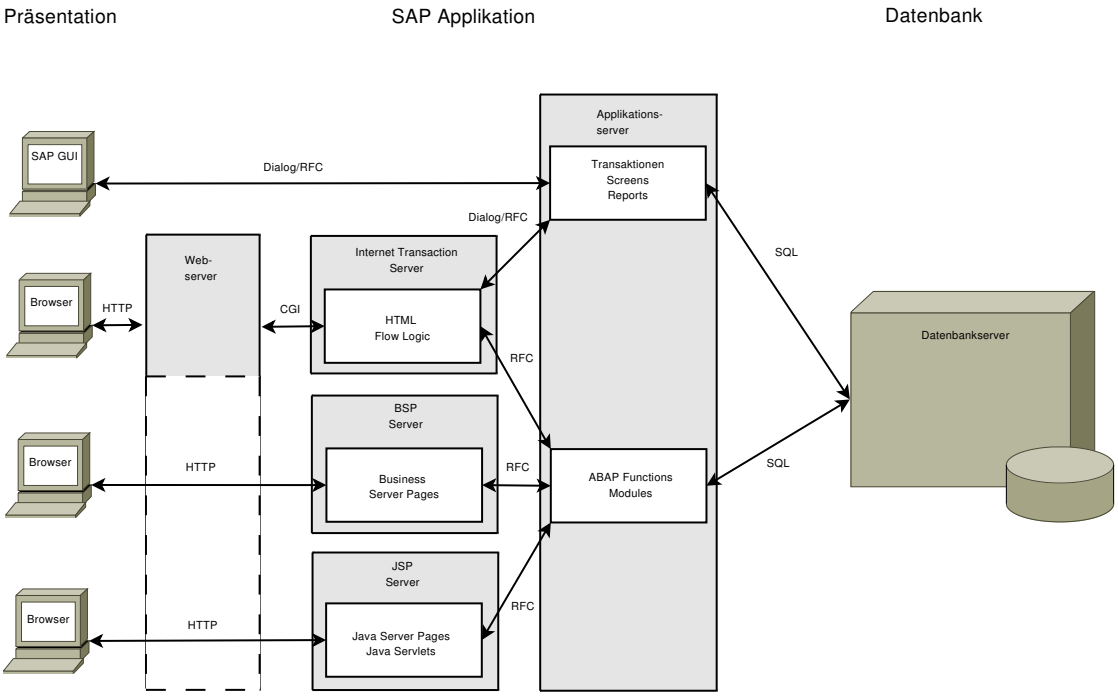

Abbildung 10: Komplexere Client-Server Architektur

- JSP (Java Server Pages): Analog zu BSP mit Java.

Der SAP Application Server stellt die Applikationslogik in Form von Transaktionen, Screens, Reports, Funktionsbausteinen zur Verfügung. Die für die Applikation nötigen Daten bekommt er als Client vom Datenbankserver. SAP hat eine eigene Datenbank, die SAP-DB und unterstützt Datenbanken aller führenden Hersteller. Marktführer in diesem Bereich ist Oracle. Zwischen dem Applikationsserver und dem Datenbankserver werden große Datenmengen ausgetauscht. Der Applikationsserver hält zusätzlich Daten in einem eigenen Datenpuffer.

4.3 Applikationsserver und Prozesse

Ein SAP-System hat mindestens einen, meist mehrere Applikationsserver. Auf ihnen laufen alle Programme und Transaktionen, hier werden alle Anfragen bedient. Dies geschieht in der Regel parallel. Damit die Anfragen performant und sicher bedient werden können, laufen auf einem Applikationsserver mehrere Prozesse oder Dienste (Services). Folgende Prozesse gibt es in einem SAP-System:

Dialog: Der Dialogprozeß dient der Bearbeitung von Anfragen von Benutzern im Dialog. In einem SAP-System gibt es im Normalfall viele Dialogprozesse. Die Anzahl der Dialogprozesse ergibt sich aus der Anforderung an das System. Sie bestimmt, wieviele Anfragen von Benutzern parallel verarbeitet werden können.

Batch: Der Batchprozeß bearbeitet Anfragen, die im Hintergrund ablaufen sollen. In einem SAP-System gibt es in der Regel viele Batchprozesse. Auch hier bestimmt die Anzahl der Batchprozesse die Anzahl der Programme, die parallel verarbeitet werden können.

Spool: Bearbeitet Spool-Aufträge, e.g. Druck. In der Regel gibt es mehrere Spoolprozesse.

Dispatcher: Dient der Verteilung der Anfragen auf die anderen Prozesse. Pro Applikationsserver gibt es einen Dispatcher.

Gateway: Dient dem Aufbau einer Verbindung zu externen Programmen und Systemen (RFC).

Verbuchung: Daten, die z.B. von einer Transaktion in die Datenbank geschrieben werden sollen, werden aus Performancegründen erst nach Beendigung des Dialogs in die Datenbank geschrieben. Diese Aufgabe übernehmen Verbuchungsprozesse. In der Regel gibt es mehrere Verbuchungsprozesse.

Message: Kontrolliert die Kommunikation zwischen den verschiedenen Applikationsservern und vergibt freie Prozeßressourcen. Es gibt pro SAP-System einen Message-Service.

Enqueue: Verwaltet Sperren. Typischerweise gibt es pro SAP-System einen Prozeß zur Verwaltung von Sperren, es kann aber auch mehrere geben. In diesem Fall müssen alle Enqueue-Services auf dem gleichen Applikationsserer laufen.

Die Rolle der einzelnen Services werden im nächsten Abschnitt genauer diskutiert.

Für die Frage, wieviele SAP-Applikationsserver ein SAP-System hat und wieviele Prozesse von welchem Typ darauf laufen, hängt von den unterschiedlichen Anforderungen an das System ab. Wichtige Kriterien sind:

- Ausfallsicherheit. Deshalb werden in der Regel mehrere Rechner verwendet, die in unterschiedliche Rechenzentren oder zumindest in unterschiedlichen Brandabschnitten stehen. Mehrere Rechner bedingen meist mehrere Applikationsserver.
- Parallelität: Die Zahl der Prozesse hängt davon ab, wieviele Anfragen parallel zu verarbeiten sind.
- Performance: Die Zahl der Prozesse hängt davon ab, wie schnell Anfragen zu bearbeiten sind.

4.4 Ablauf von Transaktionen und Transaktionsschritten

Technisch betrachtet ist eine Transaktion kein eigenes Objekt. Es gibt Programme, die im SAP System ablaufen. Diese Programme haben *Screens*, also Bildschirme mit Masken, auf denen dem Benutzer Daten angezeigt werden und wo er Eingaben machen kann. Eine Transaktion ist eigentlich nur ein Eintrag in einer Tabelle. Hinter dem Transaktionscode (dem Namen der Transaktion, z.B. ME21, Bestellung anlegen, siehe oben) ist neben Beschreibungen das Programm und der Einstiegsbildschirm hinterlegt. Der Aufruf der Transaktion startet also ein Programm, das Programm zeigt dem Benutzer einen Bildschirm, auf dem er z.B. Eingaben machen kann.

Wenn der Benutzer den Bildschirm bearbeitet hat, schickt er ihn ab, die Daten werden vom System verarbeitet, und der Benutzer bekommt ggf. einen weiteren Bildschirm gezeigt. Eine solche Verarbeitung nennt man Transaktionsschritt.

Ein einzelner Transaktionschritt besteht aus kleineren Schritten. Die Struktur eines Transaktionsschritts in in Abbildung 11 dargestellt.

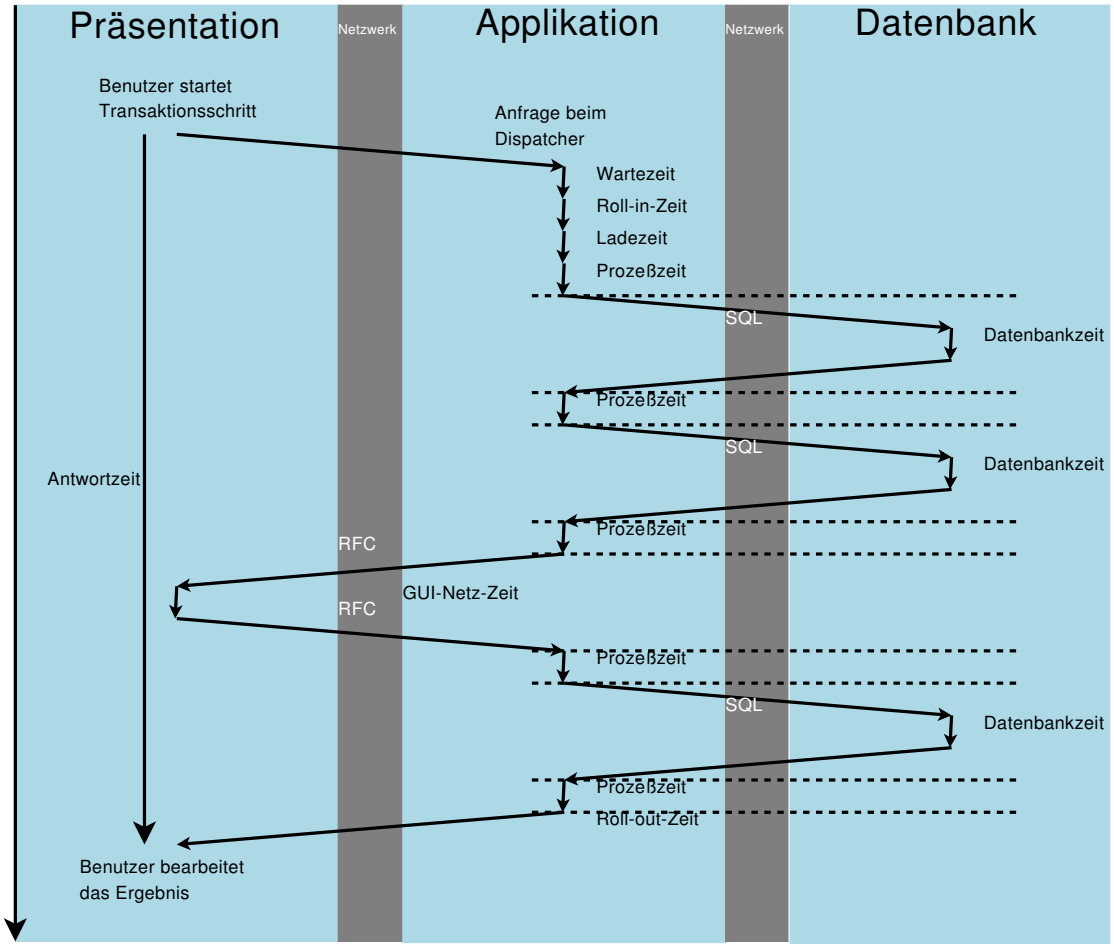

Abbildung 11: Ablauf eines Transaktionsschritts (angelehnt an [2], p. 134, 236)

Eine Transaktion besteht aus einem, meist aus mehreren Transaktionschritten, bei denen die Daten eingegeben werden. Für den Ablauf eines Transaktionsschritts wird der Dispatcher und ein Dialogprozeß verwendet. Möglicherweise werden durch die Transaktion Daten auf der Datenbank geändert. Wenn das der Fall ist, müssen die entsprechenden Tabellenfelder für die Änderung gesperrt werden. Wenn der Benutzer die Transaktion beendet hat, müssen die Daten geschrieben werden, anschließend müssen die Tabellenfelder wieder freigegeben werden. Man benötigt also eine Sperrlogik. Dazu gibt es mehrere Möglichkeiten:

1. Die Sperrlogik der Datenbank wird verwendet. Jede moderne Datenbank bietet die Mög-

lichkeit, Sperren zu setzen. Wenn die Verarbeitung der Daten innerhalb eines Transaktionschritts erfolgt, kann diese Möglichkeit verwendet werden. Meist ist das nicht der Fall.

2. Der Enqueues-Services erlaub das Sperren von logischen Objekten (z.B. Belege) über mehrere Transaktionsschritte hinweg. Beispiel: Eine Reise soll gebucht werden. Daran hängen verschiedene, im Prinzip separate Vorgänge: Buchung eines Flugs, eines Hotels, eines Mietwagens, etc.. Die Reise soll nur gebucht werden, wenn alle diese separaten Buchungen gemeinsam möglich sind. Es muß also möglich sein, über mehrere Schritte hinweg Sperren zu setzen. Im Laufe der Transaktion wird eine solche Sperre, genannt SAP-Enqueue, gesetzt. Wenn der Benutzer alle nötigen Überprüfungen vorgenommen hat und alle Daten eingegeben hat, werden die Daten geschrieben (Verbucher), anschließend wird die SAP-Enqueue freigegeben.

Wenn der Benutzer alle Daten in Transaktionsschritten eingegeben hat, müssen diese in die Datenbank geschrieben werden. Dafür gibt es mehrere Möglichkeiten:

synchron: Die Daten werden sofort während der Eingabe in die Datenbank geschrieben. Die synchrone Verbuchung kommt so gut wie nie zur Anwendung. Gründe:

- Aus logischen Gründen ist das oft nicht möglich (SAP-Enqueues).
- Die Performance wäre nicht gut.

asynchron: Die Daten werden erst dann in die Datenbank geschrieben, wenn der Benutzer den letzten Transaktionschritt beendet hat. Hier gibt es mehrere Möglichkeiten, die alle realisiert sind:

- V1-Verbuchung mit Anwendung von SAP-Enqueues.
- V2-Verbuchung ohne Sperren, aber zeitnah.
- V3-Verbuchung ohne Sperren wie V2, aber als Hintergrund-Prozeß.

In Umgebungen mit mehreren Systemen passiert es häufig, daß eine Transaktionsschritt während der Ausführung auch auf ein anderes System zugreifen muß. Das passiert über Funktionsaufrufe, sogenannte remote function calls, RFCs. In diesem Fall wird eine Verbindung zu dem anderen System hergestellt und die Funktion wird dort ausgeführt. Auch hier gibt es zwei Fälle, die man unterscheiden kann:

synchrone RFCs: Sie kommen zur Anwendung, wenn das Ergebnis des Funktionsaufrufs für die weitere Verarbeitung des Transaktionsschritts benötigt wird. In diesem Fall wird das andere System kontaktiert, dann wird die Funktion dort ausgeführt. Das geschieht in dem aufgerufenen System natürlich in einem Dialogprozeß. Im rufenden System wird während dieser Zeit der laufende Transaktionsschritt angehalten und der Dialogprozeß wird für andere Nutzer freigegeben (Roll-out). Wenn das gerufene System fertig ist, sendet es eine Meldung an das rufende System, dort wird der Transaktionsschritt wieder in einen freien Dialogprozeß geladen (Roll-in) und es wird weiter gearbeitet.

Die Zeiten für Roll-out und Roll-in sind sehr kurz typisch sind einige Millisekunden. Der Performanceverlust ist also gering im Vergleich zu dem Aufwand, den man hätte, wenn man den Dialogprozeß während der Wartezeit belegt halten würde.

Die Kommunikation zwischen verschiedenen Systemen, die auf diese Weise via RFC kommunizieren, erledigt ein Gateway-Prozeß.

asynchrone RFCs: Hier wird keine Ausgabe des Funktionsaufrufs für die weitere Verarbeitung benötigt, also läuft der Dialogprozeß nach dem RFC-Aufruf normal weiter.

Wenn der Benutzer während der Transaktion einen Druckauftrag absetzt, wird dieser Druckauftrag parallel durch den Spool-Prozeß verarbeitet.

4.5 Batchverarbeitung

Neben der Verarbeitung von Dialog-Transaktionen werden in jedem SAP-System auch viele Batchprogramme abgearbeitet. Die Batchverarbeitung läuft auf separaten Batchprozessen, damit eine sinnvolle Priorisierung möglich ist und es zu keiner Konkurrenz zwischen Dialog und Batch kommt.

Im Prinzip kann jede Dialogtransaktion auch im Batch ausgeführt werden. In der Realität wird das häufig benutzt, um große Dateneingaben (Massendatenänderungen etc.) zu verarbeiten. Man schreibt dann sogenannte Batch-Input-Mappen, in denen die auszuführenden Transaktionen der Reihe nach mit ihren jeweiligen Eingaben verzeichnet sind und die dann im Hintergrund ausgeführt werden.

Dieses Beispiel zeigt, daß die Steuerung, die wir für den Ablauf von Dialogtransaktionen beschrieben haben, in gleicher Weise auch für Batchprogramme gelten. Das führt nun aber zu folgendem Problem:

Ein Batchprogramm kann ebenfalls Funktionsaufrufe via RFC starten. Sehr häufig ist dabei das rufende System identisch mit dem gerufenen System, der Funktionsaufruf wird in dem gleichen System abgesetzt. Das ist zum Beispiel sinnvoll, um Verarbeitungen parallel ausführen zu können. Jetzt entsteht aber das Problem, daß RFCs in Dialogprozesses ausgeführt werden. Damit können also Batch-Programme in Konkurrenz zur Dialogverarbeitung stehen.

Die Konsequenz ist, daß man eine spezielle Batch-Steuerung benötigt. Jedes SAP-System hat eine eingebaute Batchsteuerung, mit der sich der Ablauf von Batchprogrammen genau planen läßt. Insbesondere ist das auch wichtig für Batchketten. Eine Batchkette liegt vor, wenn ein Batchprogramm erst dann starten darf, wenn ein anderes beendet ist. Neben einer Zeitsteuerung wird also auch eine Ereignissteuerung nötig.

Eine Bemerkung am Rande: Vielen Unternehmen erscheint die eingebaute Batchsteuerung nicht ausreichend. Sie setzen deshalb Programme zur Batchsteuerung von anderen Anwendern ein. Diese laufen entweder innerhalb eines SAP-Systems und ergänzen die vorhandene Batchsteuerung, oder außerhalb in eigenen Batchsteuerungssystemen.

5 Betrieb von SAP

Der Betrieb einer SAP-Landschaft kann auf sehr unterschiedliche Weise organisiert sein. Sehr häufig betreibt ein Unternehmen Teile selbst, andere Teile werden im Outsourcing betrieben. Auf

einem hohen Niveau läßt sich der Betrieb einer SAP-Landschaft in einem einfachen Schichtenmodell darstellen, siehe Abbildung 12.

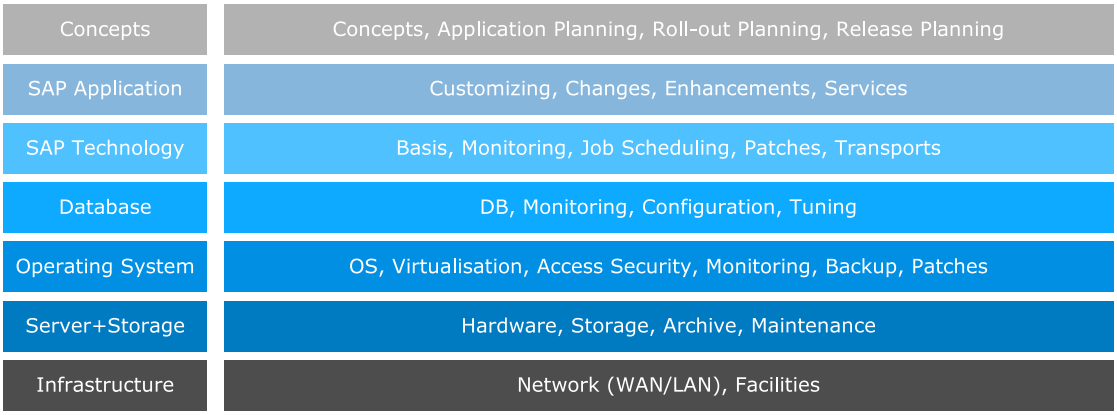

Abbildung 12: Schichtenmodell für den Betrieb einer SAP-Landschaft

Die folgenden Aufzählungen sollen einen groben Überblick über die Komplexität der Infrastruktur bieten, die nötig ist, um eine größere Menge an SAP-Systemen zu betreiben. Viele dieser Aspekte gelten genauso für andere Systeme. Die Listen sollen einen Einblick in die wichtigsten Komponenten bieten. Sie enthalten die wichtigsten Komponenten, sind aber je nach Größe der Landschaft nicht vollständig.

5.1 Infrastruktur

Netzwerke:

- Hochgeschwindigkeitsnetzwerk im RZ für die Verbindung der Rechner untereinander.
- Unternehmensinternes Netzwerk für die Kommunikation der Rechner mit den PCs.
- Anschluß an WAN.
- Sicherheitssysteme zur Abschirmung der Netze untereinander und nach außen, VPN, secureID, Firewalls, sichere Remote-Services.
- Typischer Datenverkehr zwischen einem größeren RZ und der Außenwelt liegt bei einigen PetaByte p.a.
- Telekommunikationsnetzwerke.

Raum, Strom, Klima

- In der Regel Reinraum.
- Mehrere getrennte Brandabschnitte, Brand-Früherkennungssysteme.
- Sicherheitsschleusen.

- Sicherheitssysteme für den Zugang, Videoüberwachung innen und außen.
- In großen Rechenzentren häufig Anschluß an 10kV, eigene Transformatoren, etc. Typischer Stromverbrauch: Einige Tausend 4-Personen-Haushalte.
- Überspannungsschutz.
- Batterien zur unterbrechungsfreien Stromversorgung.
- Notstromaggregate zur unterbrechungsfreien Stromversorgung. Typische Leistung einige Megawatt.
- Redundante Klimatisierung, permanente Überwachung relevanter Klimaparameter (Temperatur, Luftfeuchte, etc.)
- Redundante Kühlung.
- Heute in der Regel nach gängigen Umweltschutzrichtlinien zertifiziert.

5.2 Hardware

Plattensysteme, z.B. EMC Symmetrix, IBM System Storage:

- Schnelle Platten. Viele kleine Platten sind schneller als wenige große.
- Anbindung via Glasfaserkabel.
- Host-Bus-Adapter. Fibre Channel (FC) oder iSCSI.
- Raid-Systeme.
- Management Software dafür.

Server. Führende Hersteller IBM, HP, Sun, Dell. Betriebssysteme: Alle UNIXe (Linux, AIX, HP-UX, Solaris), Windows Server.

- In allen Teilen redundante Auslegung.
- Wartung inkl. Tausch essentieller Teile im laufenden Betrieb. Alternativ Mehr-Node-Cluster.
- Virtualisierung, z.B. IBM Power 6 mit AIX. vmware als Alternative für kleine Systeme.
- Management-Software

Backup-Systeme

- Eigene Backup-Server, eigene Plattensysteme für Backup, Bandroboter.
- Backupsoftware

Archivierungssysteme

- Eigene Archivierungsserver, eigene Plattensysteme für Backup, optisches Archiv.
- Archivsoftware.

Drucksysteme für zentralen Druck.

Steuerungs- und Überwachungshardware

Dokumentationssysteme

5.3 RZ-Betrieb

Der RZ-Betrieb, Operations-Management, ist in der Regel hochautomatisiert. Wichtige Tätigkeiten sind:

- permanente Überwachung aller System.
- Incident-Management, Problem-Management: schnelle Reaktion auf Fehler, schnelle Fehlerbehebung, Tuning auf Hardware- und Betriebssystemebene.
- Change-Management, insbesondere testen und einspielen von Patches auf der vorhandenen Hardware.
- Batch-Planung.
- Dokumentation.
- Output-Management.
- Auftragsarbeiten.

5.4 Datenbank und Basisadministration

Man unterscheidet bei dem Betrieb von SAP-Systemen zwischen dem Betrieb der SAP-Basis und dem Betrieb der Applikation. Zum Betrieb der SAP-Basis gehören alle Aufgaben, die nötig sind, damit ein SAP-System technisch einwandfrei läuft. Zum Applikationsbetrieb gehören alle Aufgaben, die nötig sind, damit die Funktion des Systems für die Benutzer und das Unternehmen in fachlich korrekter Form gewährleistet ist. Der Applikationsbetrieb ist Gegenstand des nächsten Abschnitts.

Zum Betrieb der SAP-Basis wird häufig auch der Betrieb der Datenbank gerechnet. In anderen Organisationsformen ist der Betrieb der Datenbank teil des RZ-Betriebs.

Die Sprachregelung ist in diesem Bereich zudem nicht ganz eindeutig. Aus Sicht des Rechenzentrums ist die 'Basis' die Hardware, die Betriebsysteme und ggf. die Datenbank. Die Applikation ist das SAP-System. Aus Sicht des Rechenzentrums gehört damit der SAP-Basisbetrieb zum Applikationsbetrieb.

- Tuning und technische Optimierung der Datenbank, Datenbankreorganisation, Archivierungen, Reorganisation von Indizes.
- Incident-Management, Problem-Management: schnelle Reaktion auf Fehler, schnelle Fehlerbehebung auf der Datenbank.
- Change-Management, insbesondere testen und einspielen von Datenbank-Patches.
- Tuning und Technisches Monitoring der SAP Basisfunktionalität, Early-Watch.
- Incident-Management, Problem-Management: schnelle Reaktion auf Fehler, schnelle Fehlerbehebung auf der Ebene der SAP-Basis, dazu gehört auch Kontaktieren des SAP Supports, Beachten von Support-Meldungen (OSS-Meldungen) der SAP, etc.

- Change-Management, insbesondere testen und einspielen von SAP Hotfixes und Patches.
- Einrichten neuer Benutzer, Sperren und Löschen von Benutzern.
- Monitoring von Schnittstellen, insbesondere technische Aspekte betreffend.

5.5 Applikationsbetrieb und Support

- Benutzer-Administration. Rollen und Rechte. Im Gegensatz zur Basis geht es hier um den inhaltlichen Aspekt (wer hat Zugriff auf welche Bereiche des Systems, etc.). SAP besitzt ein sehr ausgefeiltes und flexibles Rollenkonzept, daher ist dieser Bereich zum Teil sehr aufwendig.
- Lizenzmanagement.
- Benutzer-Hotline, First-Level-Support. Hier landen alle Anfragen von Benutzern, die in welcher Art auch immer Hilfe benötigen. Einfach zu lösende Anfragen werden ggf. hier direkt gelöst.
- Second-, Third-, Last-Level-Support. Behandlung komplexerer Supportanfragen. Corrective Maintenance
- Customizing / Bug fixing
- Schulung
- Monitoring der Anwendung
- Monitoring von Schnittstellen, insbesondere inhaltliche Aspekte betreffend.
- Jobplanung: Fachlicher Teil, insbesondere bei der Steuerung von Jobketten.
- Überwachung der Prozesse.
- Dokumentation.

5.6 Weiterentwicklungen, Anpassungen, etc.

Neuhochdeutsch: Change-Management. Adaptive Maintenance.

Hier geht es um jede Art von kleineren oder größeren fachlich motivierten Änderungen am System, im Gegensatz zu den technischen Änderungen (Tuning, Patches, etc.) und den Fehlerbehebungen (Corrective Maintenance).

In den meisten Unternehmen gibt es keine personelle Trennung zwischen Mitarbeitern, die Support leisten, und Mitarbeitern, die Anpassungen oder Weiterentwicklungen vornehmen. Die Trennung ist meist durch fachliche Themen festgelegt. Es gibt z.B. eine Gruppe für FI und CO, hier wird Support und Change-Management für die Module FI und CO gemacht, eine für SD, etc. Von Ausbildung und Kenntnisstand der Mitarbeiter ist eine solche Gliederung sinnvoll. Andererseits wird in den meisten Unternehmen aber finanziell strikt zwischen Support auf

der einen, Anpassungen oder Weiterentwicklungen auf der anderen Seite getrennt. Support geht kostenseitig zu Lasten der IT, für Support gibt es vereinbarte Reaktionszeiten, Fehler müssen schnell behoben werden. Anpassungen oder Weiterentwicklungen gehen kostenseitig zu Lasten der Fachabteilung, müssen ggf. genehmigt werden, etc. In den meisten Unternehmen wird diese eigentlich notwendige Trennung aber nicht konsequent durchgeführt. Oft werden kleine Änderungen mit einem gewissen Maximalaufwand (typisch 2-3 Tage) zum Support gerechnet. Das führt dann oft zu Wildwuchs. Größere Änderungen werden gestückelt, die gesetzte Grenze wird nicht eingehalten, Änderungen werden ohne Genehmigung auf dem kleinen Dienstweg durchgeführt, die Dokumentation ist nicht ausreichend, etc..

In typischen SAP-Systemen gibt es etwa einen Anteil von 20% selbst entwickelter oder modifizierter Funktionalität, die tatsächlich genutzt wird. Je nachdem, wie alt diese Systeme sind, gibt es einen großen Teil an Programmen, Transaktionen, Funktionen, etc. die früher einmal entwickelt wurden, eine Zeit lang genutzt wurden, und anschließend nicht mehr verwendet wurden. Gerade bei größeren Veränderungen, z.B. der Verschmelzung von Systemen, dem Auftrennen von Systemen oder bei Releasewechseln fallen diese Dinge negativ auf und verursachen hohe Kosten.

Im Bereich Change-Management wird oft noch zwischen zwei oder drei Klassen von Änderungen unterschieden, kleine und große oder kleine, mittlere und große Änderungen. Große Änderungen sind Projekte (z.B. Releasewechsel, Neueinführung von Funktionalität, Aufnahme neuer Geschäftseinheiten in ein System, etc.), die eine präzise Planung und Steuerung benötigen und viele Ressourcen binden. Für die verschiedenen Klassen gibt es dann unterschiedliche Genehmigungsverfahren und unterschiedliche Budgets.

Wichtige Bereiche neben Change-Management sind Qualitätssicherung und Dokumentation. Es muß sichergestellt sein, daß jede Änderung von anderen Mitarbeitern weitergepflegt werden können. Es muß sichergestellt sein, daß alle möglichen Fehler, die auftreten können, in sauberer Weise abgefangen werden.

5.7 Service Level Management

Der Betrieb eines SAP Systems oder einer SAP Systemlandschaft läßt sich in unterschiedliche Bereiche gliedern. In der Regel findet sich diese Gliederung in der Organisationsform wieder. Die verschiedenen Bereiche sind organisatorisch getrennte Einheiten, die zum Teil als Profitcenter oder als eigene Unternehmen fungieren. Teile sind evt. ausgegliedert. Dadurch entstehen Schnittstellen zwischen Organisationseinheiten. Diese Schnittstellen sollten (die Realität sieht hier oft anders aus) sauber definiert sein, mit klaren Übergabepunkten, klaren Zuständigkeiten, klaren Qualitätsmerkmalen. Regelungen für eine solche Schnittstelle nennt man Service Level Agreement (SLA). Da es mehrere dieser Schnittstellen gibt und dazwischen Abhängigkeiten bestehen, gibt es Abhängigkeiten zwischen diesen SLAs. Der Entwurf und die Überwachung dieser SLAs ist Gegenstand des Service Level Managements (siehe dazu z.B. [2], p. 42 ff). Ein typisches Szenario mit mehreren Organisationseinheiten ist in Abbildung 13 dargestellt.

Welche SLAs im Einzelfall vereinbart werden, hängt von den Bedürfnissen des Geschäfts ab und wird im Idealfall vom Geschäftsprozessinhaber festgelegt. Der Geschäftsprozessinhaber legt die

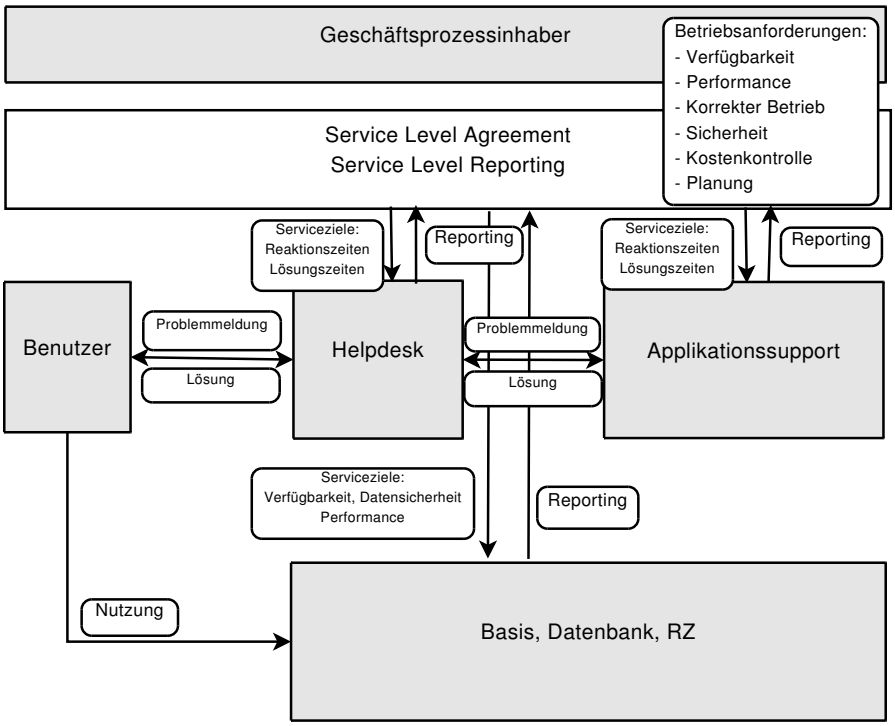

Abbildung 13: Szenario für Service Level Management

Qualität fest, sorgt so für Zufriedenheit bei den Anwendern, und versucht, diese Ziele zu niedrigen Kosten zu erreichen. In der Praxis sieht die Situation häufig anders aus: Es gibt SLAs für den Bereich Basis, Datenbank, RZ, weil dieser Bereich outgesourct ist, aber nicht für die anderen Bereiche. Die SLAs sind auch nicht mit dem Geschäftsprozessinhaber festgelegt, sondern mit der IT-Abteilung.

Typische Vereinbarungen für Basis, Datenbank, RZ:

- Festlegung von Betriebszeiten. Man unterscheidet meist
  - bedienter Betrieb: Personal ist vor Ort, führt Monitoring durch, reagiert sofort.
  - Rufbereitschaft: Personal ist nicht vor Ort, aber abrufbar.
  - unbedienter Betrieb.
  - Verfügbarkeit während der Betriebszeiten.
  - maximale Ausfallzeiten bei ungeplanten Ausfällen.
  - Wartungszeiten.
- Datensicherung und -wiederherstellung. Hier wird vereinbart:

  - Art (online, offline, Platte, Band, etc.), Umfang, Häufigkeit der Datensicherung. Meist führt man in größeren Intervallen (e.g. wöchentlich) Vollsicherungen und in kurzen Intervallen (täglich) differentielle oder inkrementelle Sicherungen durch.
  - Dauer der Datensicherung.
  - Dauer von Datenrestaurierung.
  - Wie oft wird die Datenrestaurierung getestet.
  - Verfahren im Katastrophenfall.
- Performance: Zur Performance von Dialogschritten findet man alternative Möglichkeiten:
  - Festlegen einer mittleren Antwortzeit für alle Dialogschritte oder für eine bestimmte Klasse von Transaktionen, z.B. 700ms für alle Standardtransaktionen
  - Festlegen von Quantilen für die Antwortzeit für alle Dialogschritte oder für eine bestimmte Klasse von Transaktionen, z.B. 90% aller Dialogschritte unter 1s.
  - Festlegen von Schranken oder Quantilen für bestimmte, besonders relevante Transaktionen (geschäftskritische Transaktionen, repräsentative Auswahl, etc.)
- Laufzeiten: Entsprechende Vereinbarungen kann es für die Laufzeiten relevanter Batchketten geben.
- Reaktionszeiten und Lösungszeiten bei Problemen. Typische Vereinbarungen wären:
  1. Sehr hohe Priorität: Ausfall eines zentralen Systems (viele Benutzer können nicht arbeiten). Reaktionszeit 10min, Lösungszeit 4h im bedienten Betrieb.
  2. Hohe Priorität: Ausfall oder erhebliche Beeinträchtigung einer zentralen Funktionalität für einen Anwender oder eine Gruppe von Anwendern: Reaktionszeit 2h, Lösungszeit 8h im bedienten Betrieb.
  3. Mittlere Priorität: Ausfall einer Funktionalität für einen Anwender oder eine Gruppe von Anwendern, die durch andere Funktionen umgangen werden kann: Reaktionszeit 4h, Lösungszeit 24h im bedienten Betrieb.
  4. Niedrige Priorität: Kleine Fehler. Reaktionszeit 8h, Lösungszeit 96h im bedienten Betrieb.

  Für den Fall, daß mehrere Betriebszeiten festgelegt wurden, kann es für die verschiedenen Betriebszeiten unterschiedliche Reaktionszeiten geben.
- Festlegen des Reportings: In welchem Umfang und mit welcher Frequenz wird der Prozessverantwortliche über die festgelegten Kennzahlen unterrichtet.
- Festlegen von Aktionsplänen: Was passiert, wenn eine im SLA getroffene Vereinbarung nicht eingehalten wird?
- Festlegen von Pönalen: Strafzahlungen sollten für den Betreiber motivierenden Charakter haben.

Diese Vereinbarungen betreffen die technische Seite der Systeme, nicht aber die inhaltliche Seite.

Typische Vereinbarungen für den Applikationsbetrieb:

- Festlegung von Betriebszeiten für die Benutzerhotline.
- Festlegung von Betriebszeiten für Second-, Third-, Last-Level-Support.
- Reaktionszeiten und Lösungszeiten bei Problemen. Typische Vereinbarungen sind ähnlich wie bei den Vereinbarungen zum technischen Betrieb, die Zeitfenster sind meist etwas länger.
- Klassifikation möglicher Fehler und Festlegung von Reaktionswegen und -zeiten bei den verschiedenen Fehlerklassen. Beispielsweise sind Syntaxfehler bei Programmen anders zu behandeln als abgebrochene Verbuchungen. Zusammen mit den Prioritäten ergibt sich damit eine Matrix.
- Festlegung inhaltlicher Punkte für das Monitoring der Anwendungen, z.B.
  - Abgebrochene Verbuchungen.
  - Abgebrochene Batchketten.
  - Abgebrochene Schnittstellenprozesse.
- Festlegung eines Freigabe- und Prüfverfahren für Änderungen.
- Festlegung von Transportfenstern für Änderungen.
- Format und Inhalt der Dokumentation.

Service Level Report Zusätzlich zu den oben genannten Punkten enthält ein Service Level Report oft weitere Kenngrößen, deren zeitlicher Verlauf dargestellt wird, z.B.

- Anzahl der pro Tag aktiven Benutzer.
- Anzahl Dialogschritte pro Tag oder pro Stunde.
- Mittlere Antwortzeit in der Stunde mit der höchsten Systembelastung
- CPU-Auslastung der Server.
- Hauptspeicherauslastung, Pagingraten, etc.
- Statistiken über Plattenzugriffe
- Datenbankgröße, größte Tabellen auf der Datenbank.

Diese Größen sind oft nützlich, um langfristige Trends zu erkennen und daraus Maßnahmen abzuleiten.

SAP Solution Manager Das Service Level Management ist letztlich eine Steuerung und Überwachung der IT-Prozesse. Da fast alle Unternehmen auf eine funktionierende IT angewiesen sind, sind gut funktionierende IT-Prozesse heute fast ebenso wichtig wie funktionierende Geschäftsprozesse. Diese Einsicht ist aber noch nicht überall verbreitet.

Zur Steuerung und Überwachung der IT-Prozesse und für das Service Level Management und Reporting bietet die SAP ein eigenes System an, den SAP Solution Manager (siehe dazu z.B. [2], p 64ff). Dieses System wird auch verwendet, wenn der Support der SAP in Anspruch genommen wird. Heute ist die Einführung und Verwendung eines SAP Solution Managers in den Lizenzverträgen der SAP bindend vorgeschrieben. Einige wichtige Services, die Unternehmen heute mit dem SAP Solution Manager nutzen, sind

- Solution Monitoring: Überwachung der Anwendung.
- Service Level Management und Reporting.
- SAP Early Watch Alert Service: Erkennen von mittelfristigem Optimierungspotential.
- SAP Early Watch Service: Detaillierte technische Systemanalyse.
- Customer Program Optimization: Optimierung kundeneigener Programme.
- SQL Statement Optimization: Finden und optimieren 'teurer' SQL-Abfragen.
- System Administration.
- Interface Management: Überwachung und Optimierung von Schnittstellen.
- Business Process Management and Optimization: Monitoring und Optimierung von Geschäftsprozessen.
- SAP GoingLive Check: Absicherung des Produktivstarts von Anwendungen.

5.8 Betriebsmodelle

5.8.1 Core/Context

Die Begriffe Core und Context werden in der Betriebswirtschaft verwendet, um Prozesse in Unternehmen zu unterscheiden. Die Begriffe werden nicht ganz einheitlich verwendet.

Core Mit Core werden Prozesse beschrieben, die für das Unternehmen wesentlich und charakteristisch sind, die ein Unterscheidungsmerkmal zum Wettbewerb darstellen, die Marktvorteile bringen.

Context Context bezeichnet Prozesse, die unterstützenden Charakter haben. Sie können für das Unternehmen in dem Sinne essentiell sein, daß ein Ausfall zu einem massiven Problem im Geschäft führt. Sie stellen aber kein Unterscheidungsmerkmal dar, sind nicht charakteristisch.

In der IT unterscheidet man Core und Context in dem Sinn, daß Core diejenigen Bereiche der IT umfasst, die eine Kenntnis der Geschäftsprozesse nötig machen. Context sind dagegen Bereiche, die unabhängig von den Geschäftsprozessen laufen können.

In dem Schichtenmodell (Abbildung 12) sind damit alle Bereiche unterhalb und einschließlich von SAP Technology Context, die Bereiche SAP Application und Concepts sind Core.

5.8.2 In-/Outsourcing

Hier geht es um die Frage, ob und wenn ja welche Bereiche der IT outgesourct werden können. Outsourcing bedeutet, daß ein Teil des Betriebs der IT an ein anderes Unternehmen in Auftrag gegeben wird.

Das Outsourcing von IT, insbesondere von dem Betrieb von SAP-Systemen ist heute fast der Regelfall. Wenn Bereiche in's Outsourcing gegeben werden, dann sind das meist solche, die zu Context gehören.

Typische Szenarien:

- spezielle Aufgaben (zentraler Druck, Backup)
- Rechenzentrum (Betrieb von Rechnern und Infrastruktur einschließlich Betriebssystem)
- Betrieb bis Oberkante Datenbank
- Betrieb bis Oberkante SAP-Basis
- User Helpdesk separat
- Gelegentlich: Mehrere Betreiber für unterschiedliche Bereiche.
- Selten: Volloutsourcing: Der komplette Betrieb der Systeme inkl. Applikationsbetrieb wird outgesourct. Beispiel: SRH betreibt SAP-Systeme für Krankenhäuser nach diesem Modell.

Eigener Betrieb, Vorteile:

- Man hat alles unter Kontrolle.
- Kostentransparenz
- Wenige, einfache Schnittstellen, kurze Wege.
- IT als Teil der Unternehmensausrichtung

Eigener Betrieb, Nachteile:

- Knowhow Beschaffung in vielen Bereichen.
- Hoher Aufwand, wenn die Anforderungen an SLAs hoch sind. Hohe Kosten für Sicherheit, Performance, etc.
- Kaum Skalierungseffekte, geringe Volumina gegenüber Lieferanten.
- Sehr häufig Schwächen in der Organisation, keine klare Trennung von Zuständigkeiten, insbesondere bei mittelständischen Unternehmen.

Outsourcing, Vorteile:

- Hohe Flexibilität hinsichtlich Sicherheit, Performance, etc.
- Keine Probleme bei der Beschaffung von Personal.
- Möglichkeit zur Definition klarer Schnittstellen und Übergabepunkte (SLAs).
- Planbarkeit, auch der Kosten
- Hohe Skalierungseffekte
- IT-Prozesse werden unabhängig von Personen.
- Hohe Flexibilität durch Virtualisierung, On-Demand-Lösungen, Dynamic Services.

Outsourcing, Nachteile:

- Schnittstelle zum Outsourcer muß gepflegt werden.
- Der Outsourcer will Geld verdienen, die Marge bezahlt der Kunde.
- Keine oder nur geringe Kontrolle über die fachliche Kompetenz der Mitarbeiter.
- Der Markt für Outsourcing ist intransparent. Schlechte Vergleichbarkeit unterschiedlicher Angebote. Habe ich den richtigen Outsourcer gefunden?
- Es gibt Fälle, in denen Outsourcing nicht möglich ist, zum Beispiel immer dann, wenn Geheimhaltungsvorschriften einen Eigenbetrieb erzwingen.

Mischform: Eigene IT-Tochter In großen Konzernen wird häufig die IT in eine eigene Tochterfirma ausgegliedert. Oft arbeitet diese dann auch zusätzlich als Outsourcer im Markt und betreibt die Systeme anderer Unternehmen. Gründe für diese Konstruktion sind vielfältig:

- Man hält das fachliche Knowhow in der Firma und bietet es kleineren Unternehmen der gleichen Branche an.
- Ist die IT erst einmal in eine eigene Firma ausgeliedert, kann diese leichter verkauft werden.
- Steuerliche Vorteile.
- ...

Offshoring Offshoring bezeichnet die Auslagerung von Teilen des Betriebs in das Ausland, wo Kosten, insbesondere Personal- und Raumkosten geringer sind.

Outosourcer verwenden zum Teil Offshoring. Rechenzentren oder Hot-Lines werden gelegentlich im Ausland betrieben, dann aber meist im nahen Ausland.

Vorteile sind meist Kostenvorteile.

Nachteile können sein:

- Schlechter ausgebildetes Personal.
- Hohe Personalfluktuation.
- Schlechte Sprachkenntnisse.
- Weniger gute Infrastruktur: Anbindung an Stromnetz oder Telekommunikationsnetz sind oft nicht mit deutschen Standards vergleichbar.
- Längere oder langsamere Netze führen zu längeren Antwortzeiten.
- Es gibt Fälle, in denen Offshoring nicht möglich ist, zum Beispiel immer dann, wenn Geheimhaltungsvorschriften einen Betrieb in Inland erzwingen.

5.9 ITIL

Bisher haben wir dargestellt, welche Bereiche für den vollständigen Betrieb eines SAP Systems oder eine Landschaft erforderlich sind. Als Modell dafür haben wir das Schichtenmodell (Abbildung 12) dargestellt und für jede Schicht erläutert, welche Elemente oder Tätigkeiten dort wichtig sind.

Jede Schicht innerhalb dieses Modells kann als ein IT Service verstanden werden, der seine Dienste der darüber liegenden Schicht zur Verfügung stellt. In dieser abstrakten Sicht stellt sich die Frage, wie ein IT Service generell zu organisieren ist, welche Prozesse, also IT Prozesse wichtig sind.

Während alle Unternehmen wesentlichen Wert auf die Effektivität der Geschäftsprozesse legen, dort permanente Optimierungen anstreben und bereits erreicht haben, sind die IT Prozesse häufig nicht so gut organisiert.

ITIL (IT Infrastructure Library) ist ein etabliertes System zur Beschreibung und Optimierung von IT Prozessen. Viele Unternehmen setzen dieses System ein, häufig nicht in allen Bereichen oder nicht konsequent. Dieser Abschnitt soll einen Überblick über ITIL liefern.

Die verschiedenen Prozesse von ITIL gibt es in jedem IT Service, also in jeder Schicht, die wir in dem Schichtenmodell beschrieben haben.

5.9.1 Entstehung von ITIL

ITIL wurde Ende der 80er Jahre entwickelt. Ursprünglich enstand es als Leitfaden für die Verwaltung in UK. Das System hat sich weiterentwickelt und wird heute in vielen Organisationen und Unternehmen weltweit eingesetzt.

Version 1 von ITIL waren eine Reihe von Dokumenten, die 1992 bis 1998 publiziert wurden.

ITIL 2 wurde 2001 veröffentlicht. Es besteht aus Büchern, in denen einzelne Prozesse eines IT Services behandelt werden. Die Bücher behandeln im wesentliche fünf Aspekte eines IT Services, siehe Abbildung 14.

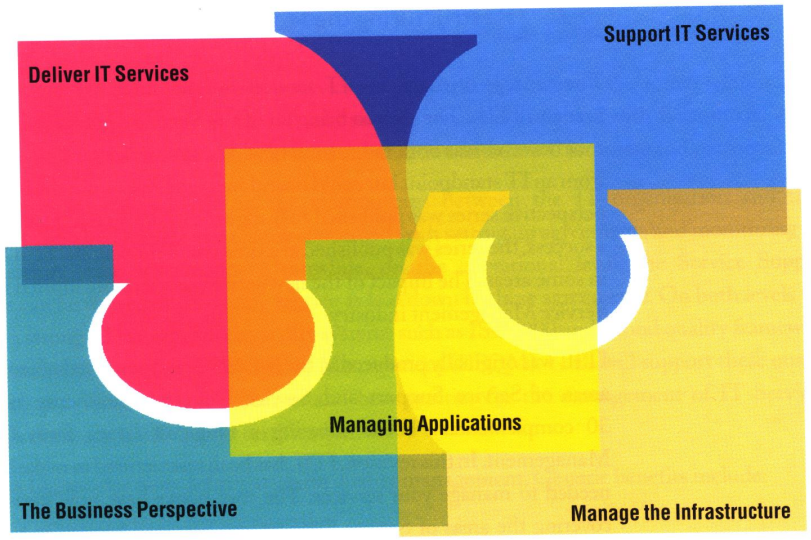

Abbildung 14: ITIL Version 2. [3]

5.9.2 Struktur von ITIL

Die einzelnen Bücher behandeln diese Themenbereiche:

1. Business Perspective

   Hier geht es um die Perspektive des Business auf die IT Prozesse. Einzelne Aspekte sind:

   - Business Continuity Management
   - Partnership and Outsourcing
   - Surviving Change
   - Transformation of Business Practice through Radical Change

2. Service Delivery

   Wie wird ein Service bereitgestellt. Wichtige Punkte sind:

   - Capacity Management.

     Welche Ressourcen werden benötigt, wieviel davon?

   - Financial Management for IT Services

     Was kostet der Service, wie werden die Kosten verrechnet?

   - Availability Management

     Wann und wo muß der Service erreichbar sein?

- Service Level Management

  Zu welcher Qualität soll der Service zur Verfügung gestellt werden?

- IT Service Continuity Management

  Wie wird die Kontinuität gewährleistet?

- Customer Relationship Management

  Wie wird die Beziehung zum Kunden organisiert?

3. Service Support

   Hier wird beschrieben, welche Prozesse für den Support wichtig sind.

   - Service Desk

     Erste Anlaufstelle für den Anwender (single point of contact).

   - Incident Management

     Wie werden Meldungen des Anwenders behandelt?

   - Problem Management

     Einige Meldungen von Anwendern deuten auf Fehler des Services hin. Wie werden diese behoben?

   - Configuration Management

     Das ist ein zentraler Bestandteil aller anderen Service Management Prozesse: Hier geht es um die Dokumentation aller Veränderungen am System. Häufig wird dieser Teil mit dem Change Management zusammen betrieben.

   - Change Management

     Wie werden Änderungen am System vorgenommen? Welche Regularien sind einzuhalten?

   - Release Management

     Größere Änderungen der Anforderungen können neue Releases nötig machen. Wie ist hier vorzugehen?

4. ICT Infrastructure

   Hier geht es um technische Aspekte eines Services.

   - Network Service Management
   - Operations Management
   - Management of Local Processors
   - Computer Instalation and Acceptance
   - Systems Management

5. Application Management

   Application management im Sinne von ITIL beschreibt Prozesse, die im Zusammenhang mit der gesamten Lebenszeit einer Anwendung stehen. Dazu gehören z.B. Punkte wie Einführung der Anwendung, Fähigkeiten von Personal, Schulungen, Ablösung der Anwendung.

6. Security Management

   Hier geht es um alle Prozesse, die für die Sicherheit einer Anwendung wesentlich sind. In diesen Bereich werden heute häufig auch Umweltfragen, Arbeitssicherheit, etc. integriert.

In Version 3 von ITIL wurde die Darstellung umstrukturiert, siehe Abbildung 15.

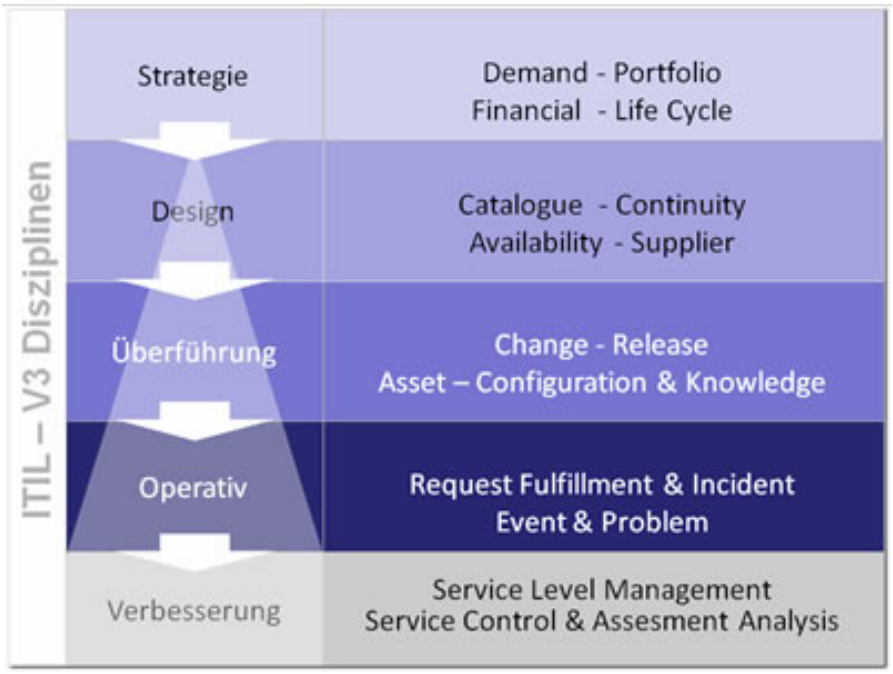

Abbildung 15: ITIL Version 3.

Zu ITIL könnte man eine komplette Vorlesung halten. Viele Begriffe, die ITIL verwendet, kamen in den vorangegangenen Abschnitten bei der Beschreibung des Schichtenmodells bereits vor. Das ist das Hauptproblem: Die Begriffe werden nicht einhaitlich verwendet und häufig werden Begriffe aus ITIL verwendet und man meint, damit hätte man ITIL realisiert.

Die einzelnen Punkte und Unterpunkte in der obigen Aufzählung zu ITIL werden jeder einzeln im Detail durch ITIL beschrieben. Es gibt präzise Empfehlungen dazu, wie diese einzelnen Prozesse zu implementieren sind.

Wichtig ist hier vor allem sich klar zu machen, daß jeder einzelne Prozeß, den ITIL beschreibt, in jedem IT Service vorkommt. Man erreicht damit eine vollständige Beschreibung des IT Services und ist sicher, daß keine Aspekte vergessen werden. Man erreicht weiter, daß Prozesse

tatsächlich auch ablaufen und nicht wegen fehlender Übergabepunkte, mangelhafter Beschreibungen, etc. hängen bleiben, weil z.B. die zuständige Person gerade in Urlaub ist und deshalb keiner daran denkt. Letzteres ist der Grundgedanke bei der Einführung von Prozessen: Prozesse sollten unabhängig von Menschen sein, die sie durchführen. Sie können dokumentiert werden und man kann in der Dokumentation verfolgen, was im Einzelfall zu tun ist. Zudem erlaubt eine saubere Prozessbeschreibung auch eine Automatisierung von Prozessen.

Tatsächlich werden die Prozesse von Menschen durchgeführt, sind daher nie unabhängig von Menschen. Deshalb ist ein weiterer wichtiger Aspekt, der durch ITIL erreicht wird, die Fehlertoleranz. Prozesse müssen so strukturiert werden, daß jeder denkbar mögliche Fehler abgefangen wird. Sie müssen zudem so strukturiert sein, daß Fehler in automatischer Weise zu Korrekturen führen.

6 Kosten für den SAP-Betrieb

6.1 TCO

Total Cost of Ownership (TCO) ist ein Begriff, der von Bill Kirwin, Research Director der Unternehmensberatung Gartner Inc., entwickelt wurde. Dahinter verbirgt sich die Idee, ein Modell zu entwickeln, mit dem die kompletten Kosten einer Anschaffung modelliert, beschrieben und erfasst werden können. Das TCO-Modell wurde für IT-Systeme entwickelt und findet hauptsächlich in der IT Anwendung. Es wird hauptsächlich von Herstellern verwendet, die damit z.B. ihren Kunden zeigen wollen, daß die Anschaffungskosten nur ein geringer Teil der gesamten Kosten sind und daß die Anschaffung des Produkts letztlich zu geringeren Kosten führt.

Wir diskutieren zunächst das TCO-Modell der SAP. Es besteht aus verschiedenen Detaillierungsgraden, genannt Level. Level 1 ist in Abbildung 16 dargestellt.

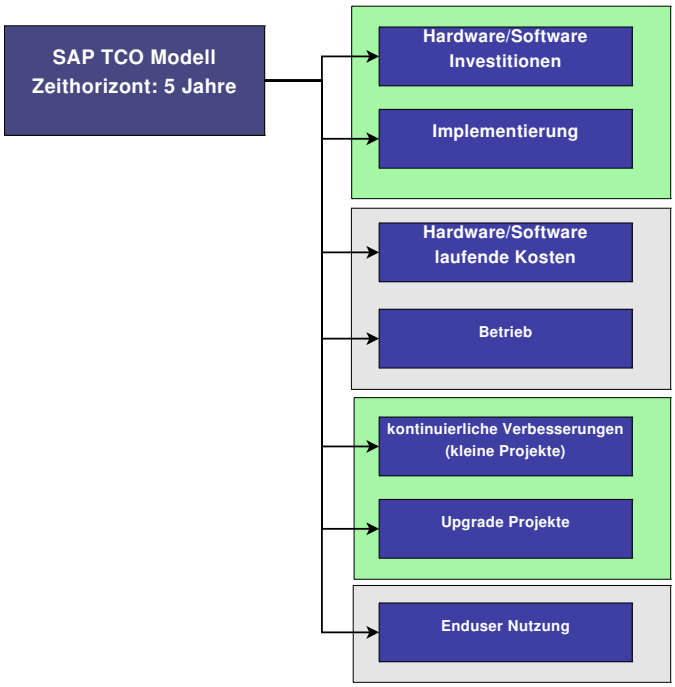

Abbildung 16: SAP TCO-Modell Level 1

Das Modell liefert auf Level 1 zunächst eine Gliederung in

1. Einführungskosten, also Anschaffungskosten für Hard- und Software und Implementierungskosten
2. Betriebskosten und laufende Kosten für Hard- und Software
3. Projektkosten für Verbesserungen und Upgrades
4. Nutzungskosten.

Diese Aufteilung zeigt schon, daß für das Kostenmodell ein Zeitraum festgelegt werden muß. Übliche Zeiträume sind die Lebensdauer des Produkts oder ein sinnvoller Produktzyklus. Für das TCO-Modell der SAP wird als Zeitraum 5 Jahre festgelegt. Das entspricht in etwa der Lebensdauer der eingesetzten Hardware und ebenfalls in etwa der Erwartung, in welchen Zeiträumen Kunden neue Releases der SAP-Software einführen.

Jeder Kostenblock in Level 1 wird in Level 2 weiter gegliedert. Die Gliederung ist in Abbildung 17 dargestellt.

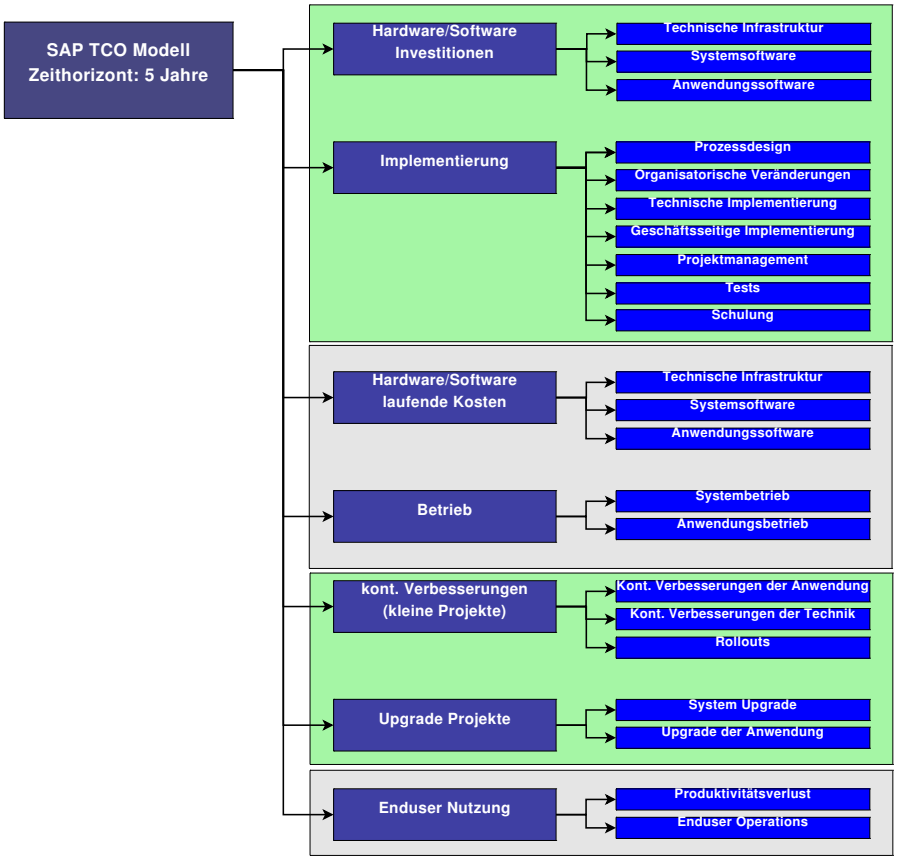

Abbildung 17: SAP TCO-Modell Level 2

Die Gliederung ist für die Einführungskosten, speziell für die Implementierung bereits recht detailliert.

Für die Hard- und Softwarekosten wird (relativ willkürlich) in technische Infrastruktur, Systemsoftware und Anwendungssoftware unterteilt. Die Anwendungssoftware ist die SAP-Software, hier stecken also die Lizenzkosten. Wir werden weiter unten das Lizenzmodell der SAP diskutieren. Dabei werden wir sehen, daß der Kunde die Datenbank, die er für den Betrieb des SAP-Systems benötigt, entweder vom Datenbankhersteller oder von der SAP direkt beziehen kann. Hier wird ein erster Problem des Modells sichtbar: Die Datenbanklizenzen stecken je nach Art

der Anschaffung in der Systemsoftware (in der Regel wird das bei DB2 für zSeries von IBM so sein) oder in der Anwendungssoftware (immer dann, wenn die Datenbank direkt von SAP lizenzisiert wird).

Bei den Betriebskosten wird nur zwischen Anwendungsbetrieb und Systembetrieb unterschieden. In unserem Schichtenmodell für den Betrieb von SAP umfasst der Systembetrieb die Betriebskosten für RZ, Datenbank und SAP-Basis, der Anwendungsbetrieb dagegen das Incident- und Problemmanagement und Teile des Change-Managements für die Applikation.

Das Modell unterscheidet zwischen kontinuierlicher Verbesserung (Continuous Improvements, das sind kleiner Änderungen) und Projekten. Kontinuierliche Verbesserungen Teile des Change-Managements. Projekte sind im Gegensatz zu kontinuierlicher Verbesserung Änderungen, die so umfangreich sind, daß sie getrennt budgetiert werden. Die Grenze zwischen kontinuierlicher Verbesserung und Projekten ist von Unternehmen zu Unternehmen unterschiedlich.

Die Kosten für die Enduser-Nutzung umfassen

- die Schulung der Endbenutzer (Education),
- den Aufwand, den der Mitarbeiter für die Organisation seiner Daten benötigt (Data Management),
- den Mehraufwand an Zeit zur Erledigung von Arbeiten am Computer, der zwischen einem Super-User und dem durchschnittlichen End-User besteht (Fuzz Factor),
- die Vor-Ort Unterstützung des Endbenutzers durch andere Mitarbeiter (Peer Support).

Diese Kosten werden sehr häufig nicht mit in die Betrachtung einbezogen.

Unterhalb von Level 2 gibt es Level 3, in dem jeder Kostenblock von Level 2 weiter untergliedert wird. Wir haben oben dargestellt, welche Hardware, Software oder Services für den Betrieb von SAP nötig sind. Für alle diese Komponenten fallen Kosten an. Jede dieser Komponenten kann in einen Bereich des Kostenmodells einsortiert werden und liefert damit einen Beitrag auf Level 3.

Wir betrachten als nächstes, wie sich die Gesamtkosten auf die einzelnen Komponenten verteilen. Diese Verteilung ist in Abbildung 18 dargestellt.

Die Verteilung der Kosten (ohne Enduser Kosten) ist für drei Szenarien dargestellt:

Global Player: Das ist ein großes, weltweit tätiges Unternehmen. Es gibt sehr viele (bis zu mehreren hundert) SAP-Systeme. Diese werden von einer konzerneigenen IT-Tochter betrieben. Die dargestellte Kostenverteilung bezieht sich auf die internen Kosten der IT-Tochter.

Local Player: Das ist ein großes deutsches Unternehmen, das hauptsächlich im deutschen Markt aktiv ist, oder ein großer, international tätiger Mittelständler. Hier gibt es ca. 10 SAP-Systeme. Hier ist angenommen, daß die Systeme von einem Outsourcer betrieben werden. Die Kostenverteilung ist die Verteilung der Kosten aus Sicht des Unternehmens. Deshalb fallen z.B. keine Kosten für Hardwareanschaffungen an, da diese vom Outsourcer getätigt werden. Statt dessen sind die Betriebskosten höher.

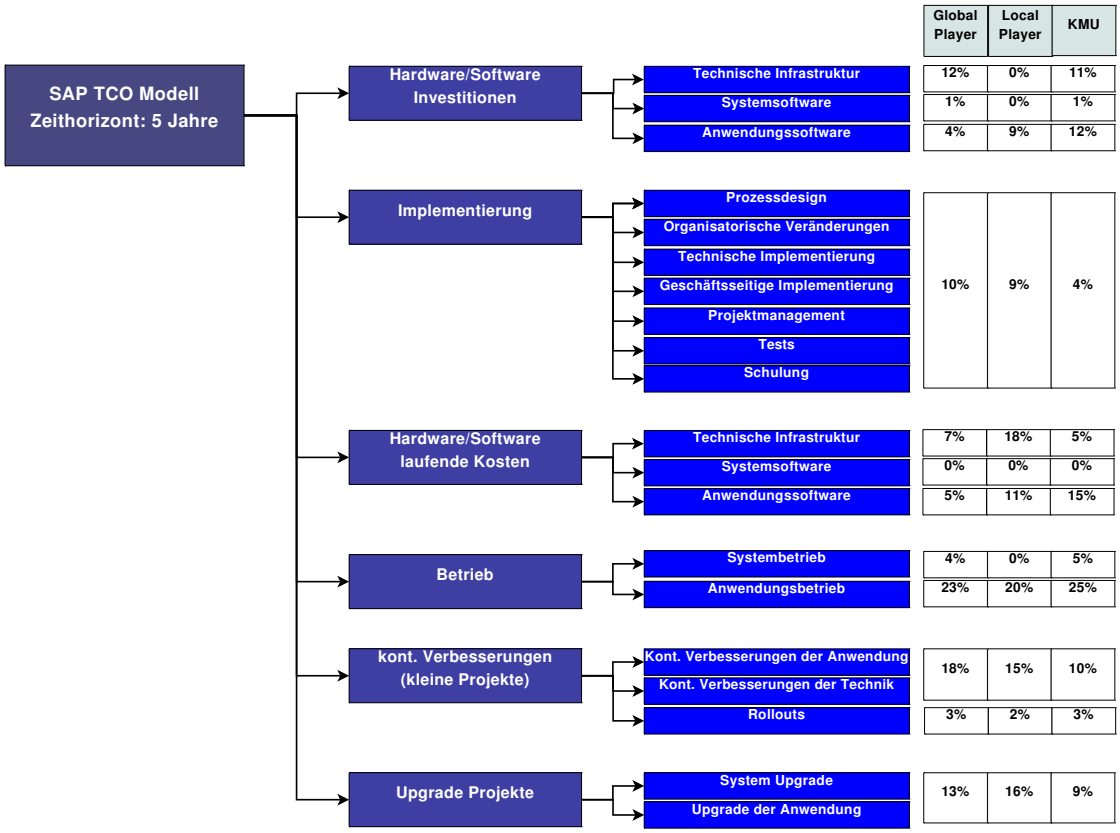

Abbildung 18: SAP TCO-Modell Level 2 mit Kostenverteilung

KMU: Ein kleines oder mittleres Unternehmen, das seine SAP-Systeme (ein bis drei) selbst betreibt.

In den Kostenverteilungen sind folgende Punkte auffällig:

- Die laufenden Kosten für Systemsoftware sind 0%, die Anschaffungskosten liegen bei nur 1%. Grund dafür ist, daß in sehr vielen Fällen die Datenbank über SAP beschafft wird und Betriebssystemsoftware zusammen mit der Hardware beschafft wird.
- Die meisten Kosten entfallen auf den Anwendungsbetrieb und die kontinuierlichen Verbesserungen, also auf die Schicht Applikationsbetrieb in dem Schichtenmodell für den SAP-Betrieb. Hier wäre eine feinere Aufgliederung (Level 3) wichtig.
- Der Anteil der SAP-Lizenzkosten liegt bei großen Unternehmen bei knapp 10%, bei kleinen über 25%. Hier macht sich die Rabattierung der SAP bemerkbar.
- Der Anteil der Hardwarekosten liegt zwischen 16% und 19%. Die Hardwarekosten sind für kleinere Unternehmen nicht günstiger, aber der Anteil ist geringer, weil der SAP-Lizenzanteil höher liegt. Für den Hardwarekostenanteil ist wichtig, daß es hier eine deutliche Zeitabhängigkeit gibt. Durch den Verfall der Hardwarepreise ist der Anteil heute

niedriger als früher. Vor 7 Jahren lag der Hardwarekostenanteil bei 25% bis 30%.

- Die Implementierungskosten sind in großen Unternehmen höher. Das liegt an verschiedenen Gründen. Häufig ist hier der organisatorische Aufwand für die SAP Einführung viel höher, die Abdeckung der Prozesse durch SAP ist höher, die Prozesse weisen eine stärkere Differenzierung und Spezialisierung auf, deshalb werden mehr Eigenentwicklungen und Modifikationen des SAP-Standards benötigt, etc.
- In die Upgrade-Kosten fließen zwei Effekte ein: Zum einen muß bei Upgrades häufig Software nachgekauft werden, hier sind große Unternehmen wegen der Rabattierung besser gestellt. Zum anderen fallen wie bei den Implementierungskosten dort höhere Aufwände für organisatorische Veränderungen an.

Kritik an TCO-Modellen

Es gibt eine Reihe von Kritikpunkten, die an TCO-Modellen angebracht werden. Die wichtigsten sind:

- Es gibt keine einheitliches TCO-Modell. Jeder Berater und jeder Analyst, damit auch jeder Hersteller hat ein eigenens Modell. Oft werden kalkulatorische Anteile für Miete, Energiekosten und Nebenkosten vergleichbarer Art nicht oder nicht vollständig berücksichtigt. Das gilt auch für Arbeitsplatzkosten, die oft nur geschätzt werden.
- Das TCO-Konzept bietet keine Methoden zur Ermittlung der indirekten Kosten durch Produktivitätsverluste.
- Es gibt keinen Zusammenhang zwischen Kosten, Nutzung und Qualität.
- Selbst bei einem vorgegebenen TCO-Modell ist die Vergleichbarkeit nicht gewährleistet, weil die Kostenstrukturen in der Realität nie dem Modell entsprechen und deshalb Kosten in unterschiedlicher Weise zugeordnet werden.
- TCO betrachtet nur Kosten. Die Wertschöpfung ist nicht Gegenstand. Dafür werden andere Modelle oder Begriffe verwendet (ROI, TBO Total Benefit of Ownership).

6.2 Lizenzkosten

SAP Lizenzkosten richten sich im wesentlichen nach zwei Größen,

- Anzahl der Benutzer,
- Volumen des Vertrags.

Die wichtigsten Nutzerklassen sind:

Professional User: Das ist der normale SAP Nutzer.

Semi-Professional User: Ein Benutzer, der das System nur eingeschränkt nutzt (in der Regel nimmt er keine Änderungen vor).

Self Service User: Verwendet nur einfache Funktionalität im Self-Service, z.B. Reisekostenabrechnung, Urlaubsantrag.

Entwickler: Entwickelt auf einem SAP-System.

Daneben gibt es weitere Nutzerklassen, die wenig vorkommen, und es gibt Nutzerklassen für spezielle Kunden.

Die wichtigen Nutzerklassen sind Professional und Semi Professional User. Der Anteil an Semi Professional darf in der Regel 20% nicht überschreiten. Ein Professional User kostet 3.800 EUR, ein Semi-Professional User 1.900 EUR. Für 10.000 Benutzer, davon 2.000 Semi-Professional wird also ein Lizenzpreis von 34 Millionen EUR fällig. Dazu kommt die Lizenz für die Datenbank, die je nach Datenbank zwischen 10% und 17% liegt. Auf den Lizenzpreis wird je nach Volumen des Vertrags ein Volumenrabatt gewährt, maximal 50%. Im vorliegenden Beispiel kommt man so auf knapp 20 Millionen EUR.

Zusätzlich zur Anschaffung wird Wartung fällig. Die Wartung liegt derzeit bei 22% des Lizenzpreises pro Jahr.

Für bestimmte Komponenten gibt es zusätzliche Kosten. Lizenzen für Personalwirtschaft (HR) berechnen sich nach der Zahl der abzurechnenden Personen, Kosten für spezielle Module für Stromkonzerne nach der Zahl der Zähler, etc.

Für Lizenzen, die im Ausland genutzt werden, gibt es Aufschläge.

6.3 Betriebskosten und Projektkosten

In der Regel wird zwischen Betriebskosten und Projektkosten unterschieden. Betriebskosten sind die Kosten, die für den laufenden Betrieb der Systeme anfallen.

Projektkosten sind dagegen Kosten, die für Projekte, also größere Veränderungen anfallen. Typische Beispiele sind Migrationen, Releasewechsel, Neueinführungen, Verschmelzen von Systemen.

Schon der Ausdruck „größere Veränderungen“ deutet an, daß „kleinere Veränderungen“ zum Betrieb gerechnet werden. Wo die Grenze liegt, ist von Unternehmen zu Unternehmen verschieden, häufig werden auch nicht nur zwei sondern mehr Größenklassen für Veränderungen verwendet. Typische Abgrenzungen orientieren sich an den Kosten, meist gemessen in Personentagen (PT). Kleine Veränderungen, die zum Betrieb gehören, haben oft einen Aufwand von 2PT. Es gibt aber auch Fälle, in denen die Grenze bei 0PT oder 180PT liegt.

Der Unterschied zwischen Betriebskosten und Projektkosten liegt in mehreren Punkten: Die Kosten werden unterschiedlich budgetiert, die entsprechenden Ausgaben werden unterschiedlich genehmigt, und die Kosten werden unterschiedlich weiterverrechnet, dazu der übernächste Unterabschnitt.

6.4 Abstrakte Kostenmodelle

Wir haben oben TCO-Modelle diskutiert. TCO-Modelle haben den Vorteil, (hoffentlich) vollständig zu sein. Ein Problem entsteht aber vor allem dann, wenn man Kosten in verschiedenen Unternehmen oder in verschiedenen Konzerneinheiten vergleichen will. In der Regel sind die Kosten unterschiedlich strukturiert und in der Regel sind sie nicht so strukturiert, wie in dem TCO-Modell vorgesehen. Damit kommt es zu Problemen bei der Erfassung und bei der Vergleichbarkeit.

Sehr häufig wird das Problem auf die Erfassung der Kosten verschoben. Die tatsächlichen Kosten werden dann so angepasst und geschnitten, daß sie in das Modell passen. Das hat verschiedene Nachteile. Der wichtigste Nachteil ist sicher das Abgrenzungsproblem: Wie wird ein Kostenblock aufgeteilt, welche Anteile sind wo zu berücksichtigen? Ein ebenso wichtiger Punkt ist aber, daß das Modell sich nach der Realität zu richten hat und nicht umgekehrt.

Dieses Problem läßt sich auf folgende Weise lösen:

Es gibt auf der einen Seite einen vollständigen Katalog von Elementen, die für den Betrieb eines SAP-Systems oder einer Landschaft nötig sind. Diese sind sowohl Hard- und Software, Raum, Strom, Klima, Verbrauchsmaterial, aber auch alle nötigen Tätigkeiten. Diesen vollständigen Katalog liefert beispielsweise ITIL. ITIL liefert hier eine sehr feine Granularität. Jedes einzelne Element kann als Kostenelement eines Kostenmodells betrachtet werden. Auch das SAP TCO-Modell aus Abschnitt 6.1 hat den Anspruch auf Vollständigkeit, liefert aber nur eine gröbere Einteilung, die im Einzelfall zu Abgrenzungsproblemen führen kann.

Auf der anderen Seite gibt es in jedem Unternehmen Ressourcen für Kosten. Dazu gehören Anschaffungen, Wartungs- und Serviceverträge, Personalkosten in vorliegender Gliederung, etc.

Das Unternehmen, das SAP betreibt, (im folgenden einfach der Kunde) gibt Kosten für Ressourcen an. Die Liste der Ressourcen K ist kundenspezifisch. Im Idealfall handelt es sich um die Kosten in genau der Granularität, in der sie dem Kunden vorliegen. In Einzelfällen kann davon abgewichen werden. Beispiele für eine sinnvolle Abweichung wären:

- Zusammenfassung mehrerer Personen zu einer Gruppe, wenn die Tätigkeiten sehr ähnlich sind.
- Aufteilung einer Person oder einer Gruppe von Personen auf Tätigkeiten, wenn eine präzise Zeiterfassung für die Tätigkeiten vorliegt.

Die Kosten der Systemlandschaft liegen in Form eines Vektors $c = (c_k)_{k \in K}$ vor. c_k sind die Kosten der jeweiligen Ressource.

Das Kostenmodell wird durch einen gerichteten Graphen $G = (V, E)$ dargestellt. Der Graph hat genau eine Quelle, die Gesamtkosten. Alle Kanten sind von der Quelle weg orientiert. Das SAP TCO-Modell ist ein Beispiel für einen solchen Graphen, es ist ein einfacher Baum.

Der Kunde ordnet Vertices des Kostengraphen seinen Ressourcen zu. Die Teilmenge der Vertices, die zugeordnet werden, bezeichne ich mit V_s. Typischerweise wird V_s Senken enthalten, es können aber auch andere Vertices in V_s enthalten sein. Es kann außerdem sein, daß bestimmte Kosten, also bestimmte Teilgraphen von G bei dem Kunden nicht anfallen. Ein typischer Fall

wäre der eines Outsourcers, der für seine Kunden Dienstleistungen nur bis Oberkante SAP-Basis (in dem Schichtenmodell) anbietet.

Um die Kosten für einen Vertex v zu berechnen, konstruiert man einen Baum $T \subset G$, $T = (V_T, E_T)$ der den Vertex v enthält und alle Senken, die von v aus erreichbar sind. Die Kosten von v ergeben sich aus der Summe der Kosten aller Kinder von v auf T und den v direkt zugeordneten Kosten. Bezeichnet man mit $\bar{c}_v$ die direkt zugeordneten Kosten und mit c_v die Kosten von v, dann gilt

$$c_v = \bar{c}_v + \sum_{u:(v,u)\in E_T} c_u \tag{1}$$

Die Berechnung der $\bar{c}_v$ beschreiben wir weiter unten.

6.4.1 Abbildung von Ressourcen auf das Kostenmodell.

Die Kosten c_k sollen auf die Kostenelemente des Kostenmodells abgebildet werden. Da Kosten additiv sind, muß diese Abbildung linear sein.

Die lineare Abbildung der Werte von K auf die Vertices in V_s beschreiben wir mit Hilfe einer Matrix $A = (a_{vk})_{v\in V_s, k\in K}$.

Es gilt $a_{vk} \geq 0$,

$$\sum_{v\in V_s} a_{vk} = 1 \,\forall k \in K \tag{2}$$

Generisch ist A reduzibel. D.h., daß $G(A)$ in unverbundene Teilgraphen $G_i(A) = (V_i(A), E_i(A))$ zerfällt. Es gilt $\bigcup_i V_i(A) \subset V_s$. Mit G_i bezeichne ich den Teilgraphen von G, der alle Vertices enthält, von denen aus die Senken $V_i(A)$ erreichbar sind. Außerdem werden in G_i suksessiv Quellen gestrichen, von denen nur eine Kante ausgeht. G_i ist ein vollständiger Teilgraph, wenn alle Senken in G, die von der Quelle von G_i aus erreichbar sind, auch Vertices von G_i sind.

Die unverbundenen Teilgraphen $G_i(A)$ sind trennbare Kostenblöcke. Wenn der zugehörige Graph G_i ein vollständiger Teilgraph von G ist, ist die Kostenzuordnung zu der Quelle von G_i sicher. Andernfalls hängen die Fehler bei der Zuordnung von den Fehlern bei der Schätzung der Kostenzuordnungen in der Matrix A ab.

Der Graph $G(A)$ wird von dem Kunden vorgegeben, d.h. der Kunde gibt an, welche Senken zu einer Ressource beitragen. Er kann auch die Matrix A ganz oder teilweise vorgeben. sofern dafür präzise Daten und nicht nur Schätzungen dafür vorliegen. In diesem Fall ist es aber besser, diese Daten für die Aufteilung der Ressource in mehrere Ressourcen zu verwenden. Beispiele für ein solches Vorgehen wären:

- Supportkosten: Jeder im Applikationssupport tätige Mitarbeiter ist eine Ressource. Die Kosten für diese Ressource sind die Vollkosten für diesen Mitarbeiter, als Gehalt, Gehaltsnebenkosten, Kosten für den Arbeitsplatz, etc. Wenn eine detallierte Zeitaufschreibung des Mitarbeiters vorliegt, kann diese verwendet werden, um die Kosten dieser Ressource verschiedenen Bereichen zuzuordenen. Ein Teil der Kosten entfällt dann auf Projektkosten, ein Teil auf kleinere Changes, ein Teil auf Incidents und Problems. So lassen sich seine

Kosten auf Incidentmanagement, Problemmanagement, Changemanagement und Projekte aufteilen.

- Serverkosten: Die Anschaffungs- und Wartungskosten für einen Server sind Ressourcen. Wenn auf dem Server verschiedene Anwendungen, z.B. verschiedene SAP-Systeme laufen, kann es sinnvoll sein, einen Schlüssel für die Verteilung der Kosten auf diese Systeme festzulegen. Ein möglicher, in diesem Fall sinnvoller Verteilungsschlüssel kann die CPU-Last sein, die die einzelnen Systeme auf diesem Server verursacht haben.

Die Matrix A wird also vorgegeben oder berechnet. Grundlage für die Matrix A ist die Verteilung der Kosten auf die Vertices V_s in den typischen, ähnlichen Unternehmen. Die Gewichte werden so berechnet, daß sich ebendiese Verteilungen innerhalb der trennbaren Kostenblöcke ergeben. Auf diese Weise kommt man z.B. auf die Verteilung der Kosten innerhalb des SAP TCO-Modells in Abschnitt 6.1. Die dort angegebenen drei verschiedenen Modelle für die Kostenverteilung, die sich an der Größe des Unternehmens orientieren, zeigen, daß für die Berechnung der Matrix A wichtig ist, wie gut die SAP-Landschaft, von der gerade die Kosten erfasst werden, mit anderen vergleichbar ist. Die Einträge der Matrix A sind also mit statistischen Verfahren geschätzt und haben entsprechend Fehler.

6.4.2 Berechnung der Kosten und der Fehler

Die direkt zugeordneten Kosten $\bar{c}_v$ für die Kostenelemente $v \in V_s$ sind

$$\bar{c} = Ac \tag{3}$$

Aufgrund von Fehlern in den Matrixelementen von A ergeben sich Fehler in den Elementen von K. Die Fehler in den Matrixelementen von A sind aber korreliert. Sei δA die Matrix der Fehler, dann ist

$$\sum_{v \in V_i} \delta a_{vk} = 0 \forall k \in K, \forall i \tag{4}$$

Wenn A irreduzibel ist (das ist nicht der generische Fall, kann aber vorkommen), dann gilt diese Bedingung nur für V_s.

Der Fehler von $\bar{c}$ ist

$$\delta\bar{c} = \delta A c \tag{5}$$

Interessant sind die mittleren Fehler

$$\sqrt{\langle \delta \bar{c}_v^2 \rangle} \tag{6}$$

oder allgemeiner die Korrelationen

$$\langle \delta\bar{c}_v \delta\bar{c}_u \rangle \tag{7}$$

Sie lassen sich aus den Korrelationen

$$\langle \delta a_{vk} \delta a_{ul} \rangle \tag{8}$$

berechnen.

Zur Abschätzung der Fehler kann man für die Wahrscheinlichkeitsverteilung von A ein Produkt unabhängiger log-normaler Verteilungen wählen, das mit δ- Funktionen für die Nebenbedingungen multipliziert wird,

$$p(A) = \mathcal{N} \prod_{v,k} a_{v,k}^{-1} \exp(-\sum_{v,k} g_{v,k}(\log a_{v,k} - \mu_{v,k})^2) \prod_{i,k} \delta(\sum_{v \in V_i} a_{v,k} - 1) \tag{9}$$

Mit diesem Verfahren erhält man aus den Kostenangaben c_k des Kunden Schätzungen für die Kosten $\bar{c}_v \pm \sqrt{\langle \delta \bar{c}_v^2 \rangle}$ der Senken des Kostenmodells und über die Aggregation der Kosten auf G auch Kosten für alle Vertices des Kostenmodells. Diese Kosten können als Grundlage für Vergleiche von Kosten unterschiedlicher SAP-Landschaften verwendet werden. Wie das genau funktioniert, wird später diskutiert.

Die Verwendung von log-normalen Verteilungen in (9) ist hier rein heuristisch begründet. Kosten müssen nicht-negativ sein, daher sind auch die Matrixelemente $a_{v,k}$nicht-negativ. Für nichtnegative Zufallsgrößen bieten sich verschiedene Verteilungsfunktionen als Ansatz an, die sich im wesentlichen durch ihr Verhalten für große Werte unterscheiden. In unserem Fall sind die $a_{v,k}$ nach oben durch 1 beschränkte. Das wird durch die Nebenbedingungen (2) oder (4) sichergestellt, die in (9) durch die δ-Funktion berücksichtigt sind. Daher spielt das Verhalten für große Werte von $a_{v,k}$ hier keine Rolle. Man kann deshalb eine möglichst einfache Verteilung verwenden. Die log-normale Verteilung hat zwei Parameter, die den Erwartungswert und das zweite Moment festlegen.

Schwierig in (9) ist die Schätzung der freien Parameter. Diese werden so gewählt, daß die Verteilung der Kosten auf die c_v den erwarteten Verteilungen innerhalb der trennbaren Kostenblöcke V_i entsprechen.

6.5 Kostenverrechnung

Ein weiteres wichtiges Thema in dem Bereich Kosten ist die Weiterverrechnung der IT-Kosten auf die Anwender. Das ist heute in jedem Unternehmen gängige Praxis. Es ist selbstverständlich, wenn die IT ein Profitcenter ist, in ein eigenes Unternehmen ausgegliedert wurde oder wenn es sich um einen Dienstleister handelt.

In der Praxis kommen unterschiedliche Verrechnungsmodelle vor. Für die Frage, wie in einem Unternehmen eine geeignete Kostenverrechnung aufgesetzt wird, sind folgende Kriterien wichtig:

- Gerechtigkeit. Die Kostenverrechnung soll verursachungsgerecht sein. Es gibt sogar gesetzliche Rahmenbedingungen, die hier einzuhalten sind.
- Einfachheit. Die Kostenverrechnung soll einfach sein. Einfachheit bedeutet zweierlei: Die Kostenverrechnung selbst muß kostengünstig sein und sie muß für den Rechnungsempfänger nachvollziehbar sein.
- Steuerung. Jede Weiterverrechnung von Kosten führt beim Rechnungsempfänger (unter Umständen) zu Veränderungen im Verhalten. Das Verhalten wird so geändert, daß die

Kosten sinken. Gerade heute herrscht in Unternehmen ein hoher Kostendruck, der sich hier sofort auswirkt. Man muß sich also bei jeder Kostenverrechnung Gedanken machen, welche Auswirkungen diese Kostenverrechnung haben wird.

Einige Beispiele sollen diese Punkte anschaulich machen:

Nutzerkosten Kosten werden auf Benutzer umgelegt. Jeder Benutzer erhält monatlich genau den gleichen Betrag in Rechnung gestellt. Dieses Modell erfüllt das Kriterium Einfachheit in hohem Maß. Es ist leicht zu implementieren und leicht zu vermitteln. Häufig wird hier argumentiert, daß ja viele Kosten, die für den Betrieb von SAP-Systemen anfallen, von der Zahl der Benutzer abhängen. Als Beispiel werden die Lizenzkosten genannt.

Gegen diese Kostenverrechnung sprechen viele Punkte:

- Die meisten Kosten, die für den SAP-Betrieb anfallen, sind nicht einfach proportional zu r Zahl der Benutzer.
- Selbst die Lizenzkosten sind für unterschiedliche Nutzertypen unterschiedlich.
- Diese Kostenverrechnung ist nicht verursachungsgerecht.

Die letzten beiden Punkte kann man teilweise dadurch berücksichtigen, daß man verschiedenen, unterschiedlich teure Nutzerklassen einführt.

Die Verteilung der Kosten nach Nutzern wird häufig als Modell verwendet, weil man annimmt, daß damit negative Steuerungseffekte nahezu ausgeschlossen sind. Das ist falsch. Beispielsweise wurden in einem Krankenhaus Kosten so verrechnet. Das Krankenhaus hat daraufhin in jeder Abteilung nur einen SAP-Nutzer eingerichtet, um Kosten zu sparen. Anfallende Eingaben wurden diesem Nutzer auf Zetteln zum Abarbeiten auf den Schreibtisch gelegt. Auf diese Weise entstehen natürlich beliebig viele Fehler. Auch die Einführung unterschiedlicher Nutzertypen führt dazu, daß versucht wird, möglichst viele Benutzer in günstige Gruppen zu sortieren. Das steht aber fast immer im Gegensatz zum Ablauf von Geschäftsprozessen.

Kostenverrechnung nach Lastparametern Hier wird als Verteilungsschlüssel ein Lastparameter oder auch eine Kombination verschiedener Lastparameter verwendet, beispielsweise die verbrauchte CPU-Last gemessen in CPU-Sekunden, die Datenbankzugriffe, da übertragene Datenbankvolumen.

Auf den ersten Blick sieht diese Form der Kostenverteilung einfach und verursachungsgerecht aus. Aber auch hier gibt es Probleme:

- Lediglich für die Serverkosten oder für die Kosten des Plattenspeichers oder der Datenbank sind diese Lastparameter als Verteilungsschlüssel verursachungsgerecht. Das ist aber ein geringer Teil der gesamten Betriebskosten von SAP Systemen, siehe Abbildung 18. Andere Kosten, z.B. der Support, sind von Lastparametern gänzlich unabhängig.
- Die Verrechnung nach Lastparametern ist für den Benutzer gänzlich intransparent. Er hat weder Einfluß auf Lastparameter, noch kann er sie beurteilen.

- Eine Steuerungswirkung ist in diesem Fall entweder ausgeschlossen oder sie führt zu ungewolltem Verhalten. Beispielweise kann es sein, daß ein Benutzer merkt, daß die Nutzung eines bestimmten Programms regelmäßig zu hohen Kosten führt. Er vermeidet dann die Nutzung dieses Programms und arbeitet an dem SAP-System vorbei. Das ist aber nicht gewollt und nicht im Sinne eines integrierten Systems zur Unterstützung der Geschäftsprozesse.

Verschiedene Verteilungsschlüssel für unterschiedliche Kosten Um die oben genannten Vor- und Nachteile zu vermeiden, könnte man unterschiedliche Kosten unterschiedlich verteilen, also z.B. Serverkosten nach CPU-Sekunden, Datenbankkosten nach Datenbankzugriffen, Supportkosten nach Benutzern. Das führt aber zu komplexen und für den Empfänger der Rechnungen intransparenten Kosten.

Eine Analogie mag das anschaulich machen: Eine solche Kostenverteilung entspräche einer Telefonrechnung, auf der separat verwendete Leitungskilometer, Stromkosten in Relais, etc. ausgewiesen sind.

Stückkosten für Transaktionen Eine alternative Möglichkeit besteht darin, für Gruppen von Transaktionen oder auch für einzelne Transaktionen oder Programme Preise für die Nutzung festzulegen. Die Festlegung kann nach verschiedenen Kriterien erfolgen. In der Regel wird man dazu ähnlich wie bei den Verteilungsschlüsseln oben geeignete Last- oder Nutzungsparameter verwenden. Der Benutzer erhält dann eine Preisliste, der er entnimmt, welche Transaktion welche Kosten verursacht. Vorteile eines solchen Modells sind

- Die Kosten sind für den Benutzer transparent. Er weiß genau, wie oft er welche Transaktion oder welches Programm aufruft.
- Die Preisgestaltung kann verwendet werden, um das Verhalten der Benutzer zu steuern.
- Die Preise können so gestaltet werden, daß die Kosten verursachungsgerecht umgelegt werden.
- Wenn die Preisliste steht, ist die Verrechnung einfach zu machen.

Nachteile sind

- Der Initialaufwand, also das Erstellen der Preisliste, ist aufwendig.
- Die Preisliste muß in regelmäßigen Intervallen angepasst werden.
- Wenn sich das Verhalten der Benutzer durch die Preisgestaltung (in gewünschter Weise) ändert, ist die Abrechnung ggf. nicht kostendeckend. Für Profitcenter kann man in diesem Fall eine Marge einkalkulieren, für andere Fälle kann meine eine Abschlagsrechnung vereinbaren.

7 Nutzung von SAP-Systemen

In diesem Abschnitt geht es darum zu verstehen, was wichtige Nutzungsparameter von SAP-Systemen sind. Dabei muß man unterscheiden zwischen der Nutzung aus Sicht der Anwender und der Nutzung der Ressourcen durch das SAP-System.

7.1 Nutzer

7.1.1 Verhalten des einzelnen Nutzers

Für einen einzelnen Nutzer ergibt sich die Nutzung in erster Linien aus den Aufgaben, die er hat. Wichtige Kriterien sind:

- Funktionalität: Einzelne Nutzer verwenden sehr häufig Transaktionen oder Programme aus einem speziellen Modul.
- Variabilität: Wie viele verschiedene Transaktionen verwendet ein Benutzer? Nutzt er im wesentlichen wenige Transaktionen aus einem Modul oder viele unterschiedliche Transaktionen aus einem Modul, oder verwendet er verschiedene Module?
- Intensität: Wie intensiv nutzt der Benutzer das System? Man unterscheidet grob Gelegenheitsnutzer (bis zu 500 Dialogschritte pro Woche), Sachbearbeiter (500 bis 5.000 Dialogschritte pro Woche) und Poweruser (mehr als 5.000 Dialogschritte pro Woche). Die Bedeutung dieser Grenzen kann man sich folgendermaßen deutlich machen: 5.000 Dialogschritte pro Woche entsprechen etwa 1.000 Dialogschritten pro Tag (etwas mehr, da Feiertage, Urlaub, etc. zu berücksichtigen sind). Das entspricht bei einer Antwortzeit von 1s einer Zeit von etwa 20min. Man rechnet typischerweise mit einem Verhältnis von Rechenzeit zu Arbeitszeit von 1:10 bis 1:15. Damit entspricht dies einer Arbeitszeit von 3,5h bis 5h. Ein Poweruser ist also ein Benutzer, der mindestens 3,5 bis 5 Stunden pro Tag mit dem System arbeitet.
- Dynamik: Hier kann man zwischen kontinuierlicher, homogener Nutzung, periodischer Nutzung und stochastischer Nutzung unterscheiden. Häufig sind diese Kriterien mit der Intensität korreliert. Gelegenheitsnutzer haben häufig eine stochastische Nutzung, manchmal eine periodische Nutzung. Poweruser haben fast immer eine kontinuierlich hohe, homogene Nutzung.

Andere wichtige Unterscheidungsmerkmale betreffen die Lizensierung:

- Professional User: Das ist ein normaler SAP-Nutzer.
- Semi-Professional User: Ein Nutzer, der im wesentlichen lesend auf das System zugreift.
- Self-Service Nutzer: Ein Self-Service Nutzer verwendet das System nur für eigene Zwecke, d.h. Reisekostenabrechnung, Urlaubsantrag, etc.

7.1.2 Verhalten der Gesamtheit aller Benutzer

Betrachtet man das Verhalten aller Benutzer, so erhält man im wesentlichen zu den Kriterien für einzelne Benutzer Verteilungen und Korrelationen. Dazu kommt noch die Dynamik der Gesamtheit. Je nach Situation kann es ein homogenes Verhalten, ein periodisches, saisonales Verhalten, ein kontinuierliches oder sprunghaftes Wachstum oder einen entsprechenden Rückgang geben.

7.1.3 Rollen und Rechte

In einem SAP-System werden Berechtigungen typischerweise über Rollen verteilt. Einer Rolle werden bestimmte Rechte zugeordnet. Die Rechte können sehr detailliert vergeben werden. Sehr häufig werden z.B. auch Rechte für bestimmte Inhalte einer Tabelle vergeben.

Nutzer haben meist mehrere Rollen. Welche Rechte ein Benutzer hat, entscheidet sich daraus, welche Rollen er zugeordnet bekommt.

Typischerweise liegt in einem System das Verhältnis der Zahl der Rollen zur Zahl der Benutzer zwischen 0,2 und 0,8. Einem Benutzer werden typischerweise zwischen 2 und 40 Rollen zugeordnet, in einzelnen Fällen aber auch deutlich mehr.

7.2 Dialog und Batchnutzung

7.2.1 Inhaltlich

- Welche Module werden verwendet?
- Innerhalb der Module: Wie hoch ist die Variabilität?
- Werden spezielle Module (z.B. Industrielösungen) verwendet?
- Bei einzelnen Transaktionen: Verhältnis zwischen Anlegen, Ändern, Ansehen von bestimmten Dingen.

7.2.2 Eigenentwicklungen

- Wieviele unterschiedliche Eigenentwicklungen gibt es?
- Wieviele unterschiedliche Eigenentwicklungen werden verwendet? Wieviele pro Tag? Eine typische Zahl sind 300 unterschiedliche Eigenentwicklungen pro Tag in einem variabel genutzten ERP System.
- Wie ist die Dynamik, gibt es Maxima in bestimmten Zeitintervallen, z.B. bei Monats- oder Quartalswechsel?

7.2.3 Dynamik

- Dialogschritte pro Tag.
- Dialogschritte pro Stunde, Verteilungen innerhalb eines Tages, einer Woche.

7.2.4 Batchverarbeitung

- Wieviele Batchsteps werden pro Tag oder pro Stunde ausgeführt?
- Wieviel Zeit wird für Batchsteps benötigt?

7.3 Schnittstellen

Schnittstellen können aus zwei Sichtweisen betrachtet werden: Zum einen gibt es die inhaltliche Seite, zum anderen die technische Seite. Wir betrachten hier zunächst die inhaltliche Seite.

Inhaltlich ist wichtig, zu welchen Systemen es Schnittstellen gibt, wie intensiv diese Schnittstellen verwendet werden, wie hoch die Fehlerraten sind, wie Fehler behandelt werden.

7.4 Nutzung von Ressourcen

7.4.1 CPU

Die CPU-Nutzung hat verschiedene Aspekte:

- Wie hoch ist die CPU-Last auf einzelnen Servern?
- Wofür wird Rechenzeit benötigt?
- Gibt es bestimmte Zeiten, in denen die Last besonders hoch ist?

7.4.2 Datenbank

Für die Datenbanklast ist zu unterscheiden zwischen

- direkten Zugriffen, das sind vollqualifizierte Zugriffe auf einen Datensatz unter Verwendug eines Index,
- sequentiellen Zugriffen, das sind Zugriffe, bei denen mehr als ein Datensatz gelesen wird,
- Schreibzugriffen, also Zugriffen, bei denen Änderungen vorgenommen werden.

Für jede dieser Arten kann man folgende Aspekte untersuchen:

- Quantität
- Dynamik
- Inhaltliche Aspekte

Daneben ist zusätzlich das Verhältnis der verschiedenen Zugriffsarten wichtig.

7.4.3 Schnittstellen

Typische Schnittstellenprotokolle

- Batch Input
- ALE
- RFC
- Dateischnittstellen

Für jedes Schnittstellenprotokoll sind die Menge der Aufrufe, das übertragenen Datenvolumen (in beide Richtungen), die Rolle des SAP-Systems (Client oder Server), die Dynamik wichtig.

8 Performance von SAP-Systemen

Einen großen Teilbereich der Qualität von SAP-Systemen hatten wir bereits im Abschnitt 5.7 besprochen. Da ging es zum einen um Vereinbarungen zur Qualität, besonders zu Datensicherheit, Verfügbarkeit, etc. Ein weiterer wichtiger Punkt war das entsprechende Reporting.

In diesem Abschnitt geht es um einen wichtigen Qualitätsaspekt: Die Performance von Systemen. Auch dazu werden in SLAs Vereinbarungen getroffen. Hier geht es darum, wie die Performance gemessen wird, wie sie beurteilt wird, wie sie verbessert werden kann.

Die Performanceanalyse von SAP Systemen ist ein komplexes Gebiet mit vielen Facetten. Dieser Abschnitt kann nur einen kleinen Einblick liefern. Für Details verweise ich auf [2].

8.1 Messung von Performance

Für die Messung von Performance steht in einem SAP-System eine eigene Gruppe von Transaktionen und Programmen zu Verfügung. Sie bilden das CCMS (Computer Center Management System). Es ist Teil der SAP Basis, also des Moduls BC.

Ein wichtiger Teil des CCMS ist der Workloadmonitor (Transaktion ST03 oder ST03N).

8.1.1 Der Workloadmonitor

Für jeden Transaktionsschritt werden in einem SAP-System Daten aufgezeichnet. Einen Teil dieser Daten haben wir im Abschnitt 7, Nutzung bereits gesehen. Es sind Daten zu Antwortzeiten, zur Datenbank- und CPU-Nutzung, zu Schnittstellen, etc. Diese Daten werden für einen einzelnen Transaktionsschritt nur für einen kurzen Zeitraum gespeichert. Anschließend werden die Daten auf Stunden- oder Tagesbasis verdichtet. Diese verdichteten Daten werden nach einer festgelegten Zeit, meist nach einer Woche oder einem Monat, weiter auf Wochen und Monatswerte verdichtet. Diese Werte werden dann nach einem festgelegten Zeitraum, drei bis zwölf Monate, gelöscht.

Der Workloadmonitor ist ein Programm, mit dem ein Administrator oder ein Experte in der Lage ist, diese Daten anzusehen und in eingeschränkter Weise auszuwerten. Die Auswertung kann unter mehreren Gesichtspunkten erfolgen. Interessant sind

- Auswertung von speziellen Performanceproblemen einzelner Nutzer, die an den Support gemeldet wurden.
- Auswertung genereller Performanceprobleme, die das ganze System betreffen.
- Aufspüren von Tendenzen.

Wir betrachten diese Punkte im folgenden genauer.

8.1.2 Betriebssystemmonitor

Der Betriebssystemmonitor bildet eine Schnittstelle zum Betriebssystem. Dahinter verbergen sich eine Reihe von Programmen, die Daten aus dem Betriebssystem sammeln und zur Analyse im CCMS bereitstellen. Die wichtigste Transaktion ist ST06. Damit kann man für einzelne Server verschieden Lastparameter ansehen. Diese Daten werden vom Betriebssystem im 10ms Takt zur Verfügung gestellt. Die Daten werden im CCMS gemittelt und sind dann auf Stundenbasis auch in der Rückschau verfügbar.

Die Interpretation dieser Daten ist je nach Konfiguration mit Vorsicht zu genießen und kann nur von Fachleuten richtig interpretiert werden. Ein besonderes Problem stellen Konfigurationen dar, die mit Virtualisierung arbeiten. In einer modernen, virtualisierten Umgebung werden die einem System zur Verfügung stehenden Ressourcen im 10 ms Takt an die Erfordernisse des Systems angepasst. So kann es sein, daß im Betriebssystemmonitor immer eine CPU-Auslastung von nahe 100% angezeigt wird, das aber nur deshalb, weil die Virtualisierungsschicht dem virtuellen Rechner, auf dem das System läuft, eben immer nur genau so viel Ressourcen zuweist, wie gerade benötigt werden.

Der Betriebsystemmonitor ist geeignet, Hardwareengpässe oder I/O-Probleme zu analysieren.

8.1.3 Datenbankmonitor

Die Datenbank liegt als Schicht zwischen Hardware und Betriebssystem auf der einen und der SAP-Basis auf der anderen Seite.

Der Datenbankmonitor erlaubt die Performanceanalyse auf Datenbankebene. Hier können einzelne Datenbankzugriffe, einzelnen Tabellen oder Indizes, etc. untersucht werden. Hier ist es auch möglich, 'teure' SQL-Aufrufe zu identifizieren und zu analysieren.

8.1.4 Weitere Punkte

Es gibt eine Reihe weiterer Aspekte, die für die Performance eines SAP-Systems entscheidend sein können, die aber nur selten in Erscheinung treten. Dazu gehören:

- Sperren
- Pufferung
- Netzanbindung

Auch hierfür gibt es jeweils entsprechende Werkzeuge.

8.2 Performanceanalyse einzelner Dialogschritte

Wir haben schon weiter oben in Abschnitt 4.4 gezeigt, wie ein Transaktionsschritt abläuft. In Abbildung 11 wurde dieser Ablauf im Detail dargestellt. Der Ablauf wird hier noch einmal aus Performancesicht diskutiert.

Dispatcher-Wartezeit Zuerst wird über den Dispatcher dem Aufruf ein freier Workprozess zugeordnet. Dies führt zu einer Wartezeit. Diese Wartezeit sollte < 50ms sein und sollte nicht länger als 10% der Antwortzeit betragen.

Ladezeit Die Transaktion wird geladen. Das sollte schnell gehen, typisch < 50ms. Probleme können z.B. durch einen zu kleinen Programmpuffer oder durch einen CPU-Engpass verursacht werden.

Roll-in-Zeit, Roll-out-Zeit Auch das sollte schnell gehen, typisch < 20ms.

Roll-Wartezeit Die Rollwartezeit ist die Zeit, die ein Prozess nach dem Roll-out wartet, wenn ein synchroner RFC aufgerufen wird. Die Dauer hängt von der Komplexität des externen Aufrufs ab. In den Fällen, wo es sich um einen externen RFC zu Frontend des Benutzers handelt, wenn also Daten zum GUI übertragen werden, sollte die Roll-Wartezeit in Summe für einen Dialogschritt < 150ms sein.

GUI-Zeit Die GUI-Zeit ist die Zeit, die während eines Dialogschritts für die RFC-Kommunikation mit dem Frontend benötigt wird. Typischerweise sollte sie <150ms sein.

Enqueue-Zeit <5ms

CPU-Zeit Die CPU-Zeit hängt von der konkreten Transaktion ab. Typisch ist ein Anteil von etwa 20% bis 25% an der Antwortzeit. Bei einer typischen Antwortzeit von 700ms liegt sie also für einen Transaktionschritt in Summe bei 150 bis 200ms.

Wichtig ist, daß die CPU-Zeit kein additiver Bestandteil der Antwortzeit ist. Das hat zwei Gründe:

- Teile der Roll-in-Zeit, Roll-out-Zeit, Ladezeit und insbesondere der Prozesszeit benötigen CPU-Zeit.
- In der Regel teilen sich mehrere Prozesse auf Betriebssystemebene eine CPU. Die Zeit, in der der Workprozess nicht von der CPU bearbeitet wird, weil gerade ein anderer Prozess bearbeitet ist, ist Bestandteil der Antwortzeit aber nicht der CPU-Zeit.

Prozesszeit Die Prozesszeit ist die Zeit, die ein Dialogschritte für die Datenverarbeitung benötigt. Hierin ist ein großer Teil der CPU-Zeit enthalten. Die Prozesszeit sollte kleiner als die doppelte CPU-Zeit sein. Ist sie größer, kommt es offenbar zu langen Wartezeiten auf Betriebssystemebene.

Datenbankzeit Die Datenbankzeit ist die Zeit, die ein Dialogschritt in Summe für alle Datenbankoperationen, also Lesen und Schreiben von Daten, benötigt. Sie liegt typischerweise bei 60% der Antwortzeit, also bei etwa 400 ms. Ist das nicht der Fall, liegt z.B. ein Datenbankproblem oder ein Problem in der Speicherkonfiguration vor.

Zeit für direkte Lesezugriffe Ein einzelner, direkter Lesezugriff sollte im Schnitt schneller als 0,5ms sein. Ein guter Wert wäre 0,3ms.

Zeit für sequentielle Lesezugriffe Ein einzelner, sequentieller Lesezugriff sollte im Schnitt schneller als 5ms sein. Ein guter Wert wäre 3ms.

Zeit für Schreibzugriffe Ein einzelnerSchreibzugriff sollte im Schnitt schneller als 10ms sein. Ein guter Wert wäre 3ms.

Diese Zeiten sind Richtwerte. Für einzelne, besonders aufwendige Transaktionen können die Zeiten durchaus höher liegen. Bei deutlich höheren Werten für einzelne Transaktionen sollte geprüft werden, ob ein Performanceengpass vorliegt.

Typischerweise wird die Analyse einzelner Dialogschritte durchgeführt, wenn einzelne Benutzer über Performanceprobleme klagen. Dabei ist zu prüfen, ob es sich um ein singuläres Problem des Benutzers handelt oder um ein generelles Problem, und ob möglicherweise der Benutzer durch sein Verhalten, also z.B. seine Eingaben das Problem verursacht hat.

8.3 Performanceanalyse allgemeiner Performanceprobleme

Wenn der Verdacht besteht, daß ein allgemeines Performanceproblem vorliegt, muß die Ursache schnell gefunden werden. Eine einfache Rechnung macht dies deutlich: Wenn in einem System im Schnitt jeder Dialogschritt statt 700ms 1,5s dauert, erhöht sich die Wartezeit des Benutzers pro Dialogschritt um 800ms. In einem typischen System arbeiten etwa 1.000 Benutzer, von denen jeder im Schnitt 800 Dialogschritte am Tag ausführt. Damit kommt es in Summe zu einer Wartezeit von knapp 180 Stunden oder etwa einem Monat Arbeitszeit. Die Kosten dafür liegen bei etwa 10.000 EUR. Nicht enthalten sind Kosten, die dadurch entstehen, daß durch kleine Verzögerungen sehr schnell sehr große Verzögerungen entstehen können. Wenn beispielsweise sich der Ausdruck von Lieferscheinen um eine Stunde verzögert, kann das bedeuten, das Ware erst einen Tag später ausgeliefert werden kann. Dadurch können sehr hohe zusätzliche Kosten entstehen.

Bei der Analyse von Performanceproblemen ist es wichtig, auf Vollständigkeit zu achten. In den allermeisten Fällen gibt es mehrere Ursachen, die sich gegenseitig verstärken. Behebt man eine der Ursachen, reicht das oft nicht.

Die oben angegebenen Richtwerte können auch für die Analyse allgemeiner Performanceprobleme verwendet werden. Zunächst muß aber geprüft werden, ob das Problem

- immer,
- periodisch, oder
- häufig, aber stochastisch

auftritt. Weiter muß geprüft werden, ob eine bestimmte Gruppe von Benutzern, eine bestimmte Funktionalität, ein bestimmter Service betroffen ist oder das ganze System. Die Ursachenforschung muß alle Bereiche, die betroffenm sein könnten, einschließen. Ein Beispiel mag das verdeutlichen:

Es stellt sich heraus, daß Datenbankzeiten lange dauern. Mögliche Ursachen können sein:

- Ein Engpass der Datenbank selbst, z.B. durch eine ungünstige Konfiguration der Datenbanksoftware. Hier gibt es viele Parameter, die eingestellt werden können.
- Ein Engpass in auf speziellen Bereichen der Datenbank, z.B. auf bestimmte Tabellen, Indices oder LOBs, auf Tablespaces, etc.
- Ein Engpass in der Plattensystem, auf dem die Daten liegen. Auch hier gibt es viele Parameter, von denen einige ungünstig eingestellt sein könnten.
- Ein Engpass auf einzelnen Platten, auf RAID-Strukturen.
- Probleme mit dem Spiegel, auf den die Daten gespiegelt werden.
- Probleme mit Plattencontrollern.
- Probleme mit dem Dateisystem.
- Probleme in der Anbindung des Plattensystems an die Server.
- Probleme auf dem Datenbankserver selbst, z.B. zu hohe Auslastung, Speicherprobleme, häufiges Paging.
- Ungünstige Konfiguration der Datenpuffer auf den Applkationsservern.
- Ungünstige Datenabfrage durch den Applikationseerver.

Konkret bedeutet eine lange Datenbankzeit einfach, daß der Weg der Daten von der Platte bis in den Speicher des Servers, wo sie verarbeitet werden, lange dauert. Die Ursache dafür kann in jedem Stück Hardware oder jedem Stück Software liegen, daß auf diesem Weg verwendet wird, und sie kann in der Art- und Weise liegen, wie der Server auf die Daten zugreift.

Lange Datenbankzeiten sind ein häufiges Performanceproblem. Da der Datenbankanteil an der Antwortzeit einzelner Dialogschritte mit 60% den größten Anteil ausmacht, wirkt sich ein Performanceproblem hier unmittelbar aus.

Performanceprobleme können aber in jedem der oben diskutierten Beiträge zur Antwortzeit auftreten und es kann in jedem Fall ein große Zahl von möglichen Ursachen geben.

8.4 Analyse spezieller Performanceprobleme

8.4.1 Performancenanalyse einzelner Programme

Sehr häufig sind einzelne, vielfach selbstentwickelte Programme die Ursache für Performanceprobleme. Die Programme können z.B. so viele Ressourcen, CPU oder Datenbank, verbrauchen, daß sie andere Transaktionen behindern. Oder sie können durch die eigene lange Laufzeit zur Verzögerung in Prozessen führen, von denen sie ein Teil sind.

In diesem Fall muß das einzelne Programm im Detail analysiert werden. Dazu gibt es eine Reihe von Analysewerkzeugen:

- Performance-Trace
- SQL-Trace
- RFC-Trace
- Enqueue-Trace
- ABAP-Trace
- ABAP-Debugger

Mit diesen Werkzeugen kann im Detail geprüft werden, welche Teile eines Programms zu langen Laufzeiten oder zu einer hohen Ressourcenbelastung führen und es kann im Einzelfall eine Optimierung angestrebt werden.

8.4.2 Performance einzelner Services

Es kann sein, daß einzelne SAP-Service langsam sind, daß einzelne Server langsam sind oder daß in anderen Bereichen ein Engpass besteht. In solchen Fällen liegt häufig das Problem in einer ungünstigen Verteilung der Benutzer oder der Workprozesse auf einzelne Ressourcen.

In diesem Fall liefert der Workload-Monitor die Möglichkeit, die Performance einzelner Ressourcen zu untersuchen.

Ein sehr häufiges Problem ist z.B. eine ungünstige Verteilung der Benutzer auf einzelne Applikationsserver. Wenn sich ein Benutzer auf dem System anmeldet, wird, wenn nichts anderes konfiguriert ist, geprüft, auf welchem Server am meisten Reserven zur Verfügung stehen. Der Benutzer wird dann auf diesem Server angemeldet.

In der Praxis melden sich die meisten Benutzer auf dem System an, wenn sie morgens mit ihrer Arbeit beginnen. Die so entstehende Lastverteilung auf die Server kann aber ungünstig sein. Man kann deshalb in bestimmten Fällen versuchen, durch eine geeignete andere Konfiguration eine bessere Verteilung zu erreichen. Häufig wird z.B. empfohlen, Nutzer, die gleiche oder verwandte Transaktionen verwenden, auf dem gleichen Server anzumelden. Hintergrund ist, daß dann die Pufferung der Daten auf diesem Server besser ist. Andererseits kann dadurch leicht ein Engpass erzeugt werden, weil alle Benutzer dieses Servers z.B. besonders CPU-intensive oder besonders Datenbank-intensive Transaktionen durchführen.

Im Grunde genommen müssen solche Fragen wenn immer möglich bereits bei der Planung einer Systemlandschaft berücksichtigt werden.

8.4.3 Schnittstellen

Je nach Typ der Schnittstelle stellen sich unterschiedliche Fragen hinsichtlich der Performance. Die einzelnen Typen von Schnittstellen haben wir in Abschnitt 7.4.3 aufgelistet.

Für die Dialogperformance eines Systems ist die RFC-Performance wichtig. Einzelne RFC-Steps werden wie Dialogschritte behandelt.

Für andere Schnittstellen stehen zum Teil spezielle Werkzeuge zur Verfügung, zum Teil können die Programme, die die Schnittstelle bedienen, mit den Werkzeugen für die Analyse einzelner Programme untersucht werden.

8.4.4 Netzwerk

Die Anbindung von Frontends an die Server und die Verbindung von Servern untereinander läuft über Netzwerke. Über diese Netzwerke läuft in der Regel nicht nur der Datenverkehr der SAP-Systeme. In der Regel findet hier auch der Datenverkehr statt, der von Mail-, Workflow oder Office-Systemen verursacht wird. Dieser Verkehr überwiegt häufig. Engpässe im Netzwerk sollten deshalb mit speziellen, dafür vorgesehenen Werkzeugen durchgeführt werden. Häufig ist es auch sinnvoll, die Netzwerkanalyse vom Client auch durchzuführen.

8.4.5 Sperren

Sperren sollten in der Regel keine Performanceengpässe verursachen. Wenn es hier zu Performanceproblemen kommt, dann sind diese in der Regel für einige Benutzer besonders gravierend. Für die Analyse von Sperren existieren ebenfalls spezielle Werkzeuge.

8.5 Analyse von Tendenzen

Um schleichend problematischer werdende Engpässe frühzeitig zu erkennen, ist es wichtig, das Zeitverhalten der verschiedenen, oben genannten Parameter und ihre Zusammenhänge, Korrelationen etc. mit geeigneten statistischen Werkzeugen zu untersuchen.

In diesem Bereich gibt es in Unternehmen heute den größten Engpass. Dafür gibt es mehrere Gründe:

- Die Daten werden im CCMS über längere Zeiträume nur aggregiert vorgehalten. Das schränkt die Analysemöglichkeiten erheblich ein. Das CCMS ist ursprünglich für die Analyse spezieller Performanceprobleme entworfen und hat für die Analyse von Trends keine gute Datenbasis parat.

- In der Regel ist das Personal in der IT mit den täglichen Problemen mehr als ausgelastet, es hat keine Zeit, sich um solche Analysen zu kümmern.
- In der Regel ist das Personal in der IT für die statistische Datenanalyse nicht ausgebildet. Dieses Nichtwissen drückt sich meist in dem Vorurteil „Glaube einer Statistik nur, wenn Du sie selbst gefälscht hast.“ aus. Die präzise Analyse einer großen Menge an Daten mit statistischen Verfahren benötigt spezielle Kenntnisse und spezielle Werkzeuge, die in der Regel nicht vorhanden sind.
- Selbst das SAP-BW, also das Standardwerkzeug der SAP für die Analyse von Daten in Unternehmen bietet keine besonderen Werkzeuge für statistische Analysen.

Die Analyse von Trends ist dehalb in aller Regel das Geschäft von Spezialanbietern oder Beratern. Das Hauptproblem ist dann aber meist die Datenbeschaffung.

8.6 Modelle für die Antwortzeit

Eine Transaktion läuft in einem ERP-System in einer stochastischen Umgebung. Da das System viele Transaktionen zur gleichen Zeit ausführt, greifen mehrere Transaktionen (genauer mehrere Workprozesse, die diese Transaktionen ausführen) auf Ressourcen zu. In einem detaillierten Modell einer einzelnen Transaktion könnte man einen Transaktionsschritt durch eine Zustandsvariable $\xi(t)$ und das dynamische Verhalten der Zustandsvariablen durch eine stochastische Differentialgleichung. Man kann oBdA $\xi(t)$ so wählen, daß der Transaktionsschritt bei $\xi(t) = 0$ startet und bei $\xi(t) = 1$ endet. Wir nehmen an, daß $\xi(t)$ ein monoton wachsender stochastischer Markovprozess ist.

Die statistischen Eigenschaften von $\xi(t)$ lassen sich durch eine Wahrscheinlichkeitsdichte $\rho(x,t)$ beschreiben.

$$p(x_0 \leq \xi(t) \leq x_1) = \int_{x_0}^{x_1} \rho(x,t)dx \tag{10}$$

ist die Wahrscheinlichkeit dafür, daß $\xi(t)$ zum Zeitpunkt t einen Wert zwischen x_0 und x_1 annimmt. den Wert x annimmt Wenn $\xi(t)$ ein Markovprozess ist, dann kann man das Verhalten von $\rho(x,t)$ durch eine Fokker-Planck-Gleichung

$$\frac{\partial \rho}{\partial t} = L_{FP}\rho \tag{11}$$

beschreiben [4]. Der Erwartungswert für die Antwortzeit des Transaktionsschritts τ ist dann die *first passage time* für $\xi(t)$ mit Startwert 0 und Endwert 1, also die Zeit, die $\xi(t)$ benötigt, um von 0 startend den Wert 1 zu erreichen. Die Antwortzeitverteilung $p(\tau)$ ist durch $p(\tau) = -\frac{d}{d\tau}\int_0^1 dx\,\rho(x,\tau)$ gegeben [4].

Ein solches, detailliertes Modell gibt es für Transaktionschritte in ERP-Systemen nicht. Zudem ist es extrem schwierig, ein solches Modell empirisch zu verifizeiren, weil es praktisch unmöglich ist, den Fortschritt eines Transaktionsschritts zu messen. Das wäre im Prinzip mit geeigneten Traces (siehe Abschnitt 8.4.1) möglich. Allerdings werden dieses Traces den Ablauf des Transaktoinsschritts so stark beeinflussen, daß er nicht mehr dem ursprünglichen Verhalten entspricht.

Um die Situation zu vereinfachen, machen wir einige plausible Modellannahmen:

1. Alle Transaktionen können jederzeit ohne Verzögerung ausgeführt werden. Zu jeder Zeit stehen ausreichend Ressourcen zur Verfügung. Damit werden Performanceengpässe ausgeschlossen
2. Periphäre Prozesse (Wartezeiten, Roll-in-Zeiten, etc.) sind vernachlässigbar.
3. Der Zustand des Systems kann durch eine Menge externer Parameter beschrieben werden. Solche Parameter können z.B. die CPU-Last, die Zahl von Datenbankzugriffen oder Plattenzugriffen, die Netzwerkbelastung, etc. sein. Eine Variation dieser Parameter verändert jede laufende Transaktion in ähnlicher Weise. Das bedeutet, daß Transaktionen unabhängig voneinander betrachtet werden können.
4. Jeder Transaktionsschritt kann durch einen Satz interner Parameter beschrieben werden, zum Beispiel durch die Menge der zu lesenden Daten.
5. Die Zeitskala, auf der Systemparameter sich verändern, ist lang verglichen mit der Antwortzeit eines Transaktionsschritts.

Die ersten beiden Annahmen sind idealisierte Konsequenzen aus der Architektur eines SAP R/3-Systems. Die wesentlichen Beiträge zur Antwortzeit sind die Prozesszeit, die Datenbankzeit und die GUI-Zeit, alle anderen Parameter spielen nur dann eine Rolle, wenn Performanceprobleme vorliegen.

Die letzte Bedingung bedeutet, daß ein Transaktionsschritt eine statische Umgebung sieht. Veränderungen dieser Umgebung sind langsam. Das Modell, das wir aufgrund dieser Annahmen konstruieren werden, wird deshalb ein statisches Modell sein. Die Antwortzeit wird eine Funktion der internen und externen Parameter sein.

Wir bezeichnen die Parameter (externe und interne) mit$\mathbf{a} = (a_1, \ldots, a_n)$. Die Antwortzeit ist dann eine Funktion $\tau = \tau(\mathbf{a})$. Wenn man die Verteilungsfunktion der Parameter $P(\mathbf{a})$ kennt, kann man die Antwortzeitverteilung ausrechnen

$$p(\tau) = \int dP(\mathbf{a})\delta(\tau - \tau(\mathbf{a})) \tag{12}$$

Eine solche Beschreibung kann man als Mean-Field Approximation des detaillierten dynamischen Modells interpretieren, das wir oben kurz angerissen haben. Die Parameter $\mathbf{a} = (a_1, \ldots, a_n)$ beschreiben das mittlere Feld.

In einem nächsten Schritt werden wir plausible Skalenannahmen für $\tau(\mathbf{a})$ vornehmen. Wir nehmen an, daß für den Fall, daß man einen Parameter mit einem Faktor λ multipliziert, sich für τ ergibt

$$\tau(a_0 \ldots a_{i-1}, \lambda a_i, a_{i+1} \ldots a_n) = \lambda^{s_i}\tau(\mathbf{a}) \tag{13}$$

wobei s_i nicht-negative Exponenten sind. Dieses Verhalten kann man leicht motivieren. Wenn z.B. die CPU-Last auf dem Server oder die Zahl der Plattenzugriffe verdoppelt wird, wird sich die Antwortzeit ebenfalls um einen Faktor vergrößern. Wenn der Transaktionsschritt doppelt so viele Daten liest und verarbeitet, wird er ungefähr doppelt so lange dafür benötigen, etc.

Diese Skalenhypothese ist die wesentliche Annahme in unserem Modell und sie hat weitreichende Konsequenzen. Für diese Skalenhypothese ist wichtig, daß es keine kleinen additiven Beiträge zur Antwortzeit gibt, daß also z.B. die initiale Wartezeit oder die Roll-in-Zeit verschwinden. Das war eine der Annahmen, die wir oben getroffen haben. Andernfalls wäre (13) falsch. (13) ist sicher auch dann falsch, wenn das System Performanceengpässe hat, die die Antwortzeit dominieren.

Eine direkte Konsequenz von (13) ist

$$\tau(\mathbf{a}) = \tau_0 \prod_i \left(\frac{a_i}{a_{i0}} \right)^{s_i} \tag{14}$$

und damit

$$\ln(\frac{\tau(\mathbf{a})}{\tau_0}) = \sum_i s_i \ln(\frac{a_i}{a_{i0}}) \tag{15}$$

Wir nehmen weiter an, daß Korrelationen zwischen den Parametern a_i unwichtig sind. Eine wichtige Voraussaetzung dafür ist, daß es keinen Ressourcenengpass gibt. Wenn diese Annahme stimmt, ist die rechte Seite (15) eine Summe von vielen unabhängigen Zufallsvariablen Damit kann für diese Summe der zentrale Grenzwertsatz angewendet werden. Das bedeutet, daß $\ln \tau$ in guter Näherung normalverteilt ist, also

$$p(\tau)d\tau = \frac{1}{\tau\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2} (\ln(\frac{\tau}{\tau_m}))^2 \right) d\tau \tag{16}$$

Hier ist τ_m der Median, $\tau_{av} = \tau_m \exp(\sigma/2)$ ist der Mittelwert und $\tau_{av}^2(\exp(\sigma^2) - 1)$ ist die Varianz.

Zusammengefasst ist jeder einzelne Parameter durch eine multiplikative stochastische Größe beschrieben und der Effekt vieler multiplikativer stochastischer Größen führt hier zu einer Log-Normal-Verteilung.

8.6.1 Abweichungen vom idealen Modell

Die Annahmen, die wir für unser ideales Modell getroffen haben, sind häufig verletzt. Hier ein paar wichtige Punkte

- Die Skalenhypothese für τ in (13) schließt additive Beiträge zur Antwortzeit τ aus. In realen Systemen wird es kleine additive Beiträge geben, die dann zu kleinen Korrekturen von (16). Große additive Beiträge würden dagegen ein Performanceproblem bedeuten.
- In realen Systemen kann ein einzelner Beitrag in (15) die Summe dominieren. Das passiert zum Beispiel dann, wenn es einen generellen Performanceengpass gibt.
- In realen Systemen werden die Parameter a_i korreliert sein. Zum Teil kann man diesen Effekt eliminieren, in dem man geeignete Kombination dieser Parameter verwendet, für die die Korrelationen klein sind. Tatsächlich sind wir in der Wahl der parameter relativ frei.

Man muß nur sicherstellen, daß man einen (im wesentlichen) vollständigen Satz von Parametern verwendet. Wir können nicht ausschließen, daß kleine Korrelationen übrig bleiben. Generell werden schwache Korrelationen die Form der Verteilung aber nicht wesentlich beeinflussen. Große Korrelationen dagegen können ein Hinweis auf ein Performanceproblem sein.

Die Argumente zeigen, daß jede Abweichung von der idealen Situation ein Hinweis auf ein Performanceproblem sein kann. Wenn dieses Modell in Ordnung ist, bedutet das, daß wir für einzelnen Transaktionen Antwortzeitverteilungen messen können und aus der Verteilung auf Performanceprobleme schließen können. Dazu werden im folgenden einige Beispiele diskutiert.

8.6.2 Beispiele mit guter Performance

Wir betrachten zunächst einige Beispiel mit guter Performance.

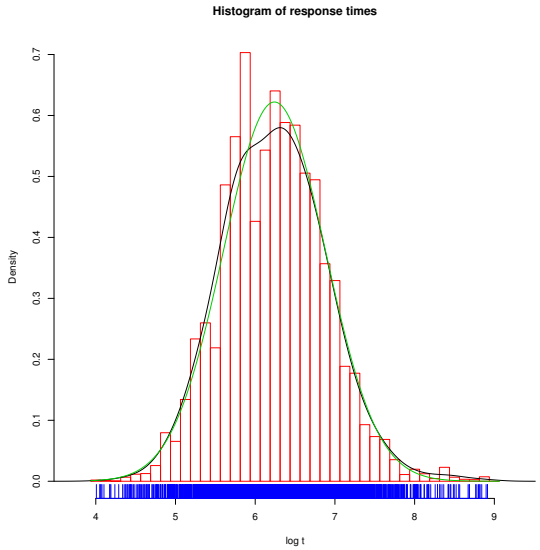

Abbildung 19: Transaktion VA01, System 1: Die Abbildung zeigt Datenpunkte (Logarithmus der Antwortzeit in Einheiten von 1ms, blau), ein Histogramm (rot), eine Dichteverteilung (schwart) und eine Normalverteilung mit gleichem Mittelwert und gleicher Varianz. $\ln t = 6$ entspricht eine Antwortzeit $t \approx 400$ms, $\ln t = 8$ entspricht $t \approx 3000$ms. Die Basis sind 324,000 Datenpunkte.

Abbildung 19 zeigt die Auswertung von Antwortzeiten von VA01 in System 1. Die Verteilung entspricht sehr gut einer Log-Normal-Verteilung. Es gibt kleine Abweichungen für $\tau \approx \tau_m$. Die Log-Normal-Verteilung wurde direkt aus dem Mittelwert und der Varianz berechnet, es wurden keine Parameter angepasst. Ähnliche Resultate erhält man für VA02 und SESSION_MANAGER in System 1 und für SESSION_MANAGER in System 2, siehe die Abbildungen 19 bis 22. Die

Beispiele zeigen eine erstaunliche Übereinstimmung der gemessenen Verteilungen mit einer Log-Normal-Verteilung.

Die Übereinstimmung zwischen den Meßdaten und der Log-Normal-Verteilung gilt, soweit die Datenlage eine Beurteilung zuläßt, mit hoher Präzision auch für die Schwänze der Verteilung, also für große Antwortzeiten. Das ist wichtig für die Beurteilung von Antwortzeiten allgemein. Lange Antwortzeiten kommen vor und ihre Häufigkeit wird durch eine Log-Normal-Verteilung beschrieben.

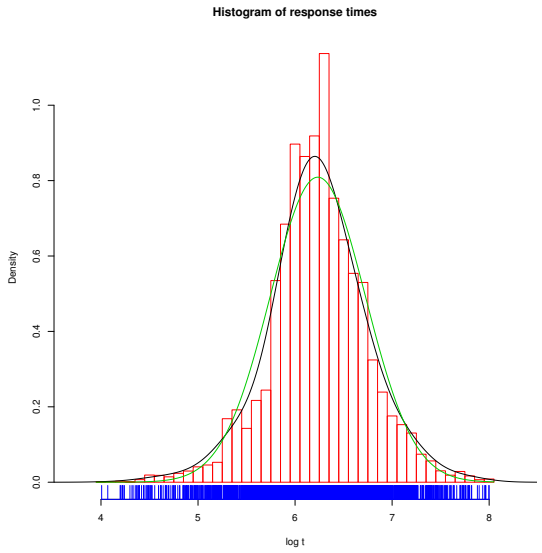

Abbildung 20: Transaktion VA02, System 1: Details siehe Abb. 19. Basis: 419,000 Datenpunkte.

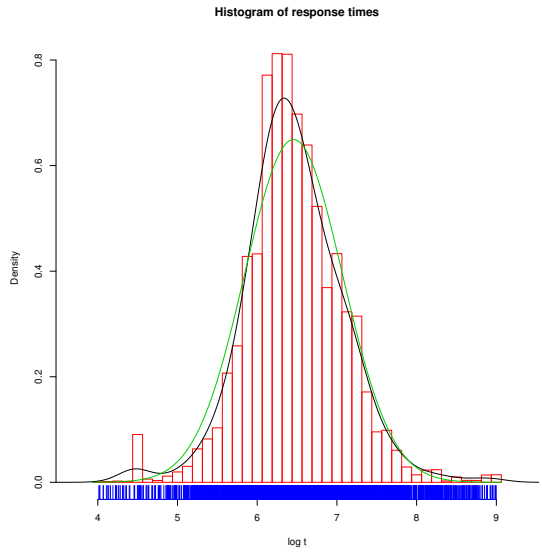

Abbildung 21: SESSION_MANAGER, System 1: Details siehe Abb. 19. Basis: 296,000 Datenpunkte.

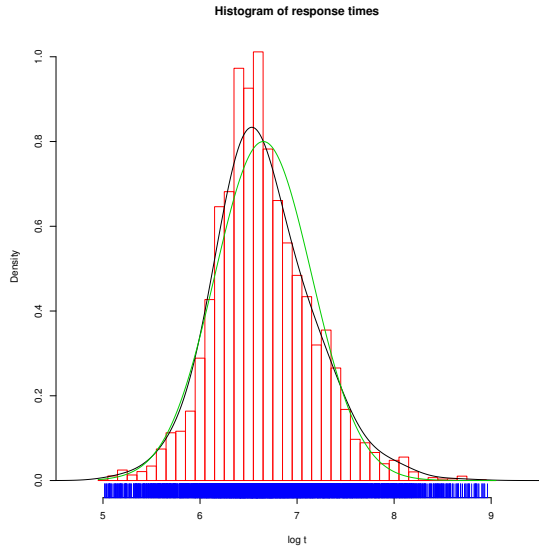

Abbildung 22: SESSION_MANAGER, System 2: Details siehe Abb. 19. Basis: 1,095,000 Datenpunkte.

8.6.3 Beispiele mit Performanceproblemen

Im folgenden betrachten wir einige Beispiele mit Performanceproblemen, bei denen eine Abweichung von der Log-Normal-Form auftritt.

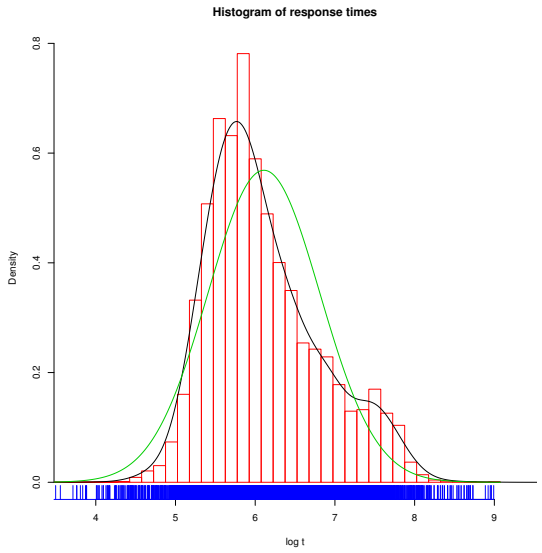

Abbildung 23: VA01, System 2: Zu den Details siehe Abb. 19. Basis 143,000 Datenpunkte.

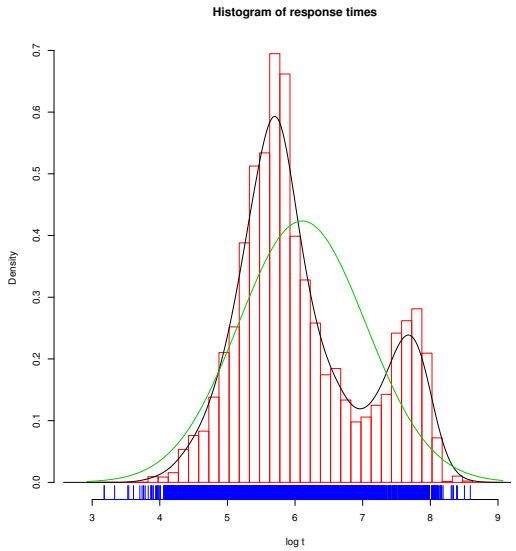

Abbildung 24: VA02, System 3: Für Details siehe Abb. 19. Basis: 281,000 Datenpunkte.

Die beiden Beispiele, VA01 in System 2 (Abb. 23) und VA02 in System 3 (Abb. 24) zeigen deutliche Abweichungen von der Log-Normal-Form. Insbesondere Abb. 24 zeigt deutliche Abweichungen. Die Kurve hat eine Form mit zwei Maxima. So ein Fall tritt unter anderem dann auf, wenn die Transaktion in einem System mit zwei sehr verschiedenen Parametersätzen läuft. Dafür kann es verschiedene Gründe geben:

- Unterschiedliche Nutzergruppen, die die Transaktion unterschiedlich verwenden.
- Unterschiedliche Aufgabenstellungen, die mit der gleichen Transaktion ausgeführt werden.
- Periodisch unterschiedliche Nutzung der Transaktion.
- Ein temporärer Performanceengpass während der Messung.
- Periodische Performanceengpässe während der Messung.
- Unterschiedlich starke Server, ungünstige Lastverteilung auf gleiche Server.

Es kann auch zu Kombinationen mehrerer dieser Gründe kommen. Beispielsweise kann es sein, daß eine Nutzergruppe das System nutzt, während regelmäßige laufende Batchprogramme eine hohe Last verursachen. Dann treffen der erste und der letzte Punkt zu.

In Abb. 23 liegt ein ähnlicher, wenn auch weniger ausgeprägter Fall vor.

In beiden Fällen lagen temporäre Performanceprobleme vor, die zu der Abweichung geführt haben.

Aus der Abweichung von der Log-Normal-Form kann man auf ein Performanceproblem schließen, es bleibt aber unklar, von welcher Art das Problem ist. In einem Fall wie für VA01 in system 2 (Abb. 23) kann man das Performanceproblem nur durch eine Detailanalyse der CCMS-Daten eingrenzen. In diesem Spezialfall konnte das Problem auf langsame sequentielle Lesezugriffe auf einer Tabelle zurückgeführt werden. Nachdem das Problem behoben war, entsprach die Verteilung wieder einer Log-Normal-Verteilung.

Bisher haben wir nur Beispiele für intensiv genutzte Standardtransaktionen gezeigt. Die gleiche Analyse läßt sich auch für Eigenentwicklungen durchführen. In Abb. 25) ist die Verteilung für eine Eigenentwicklung in System 1 gezeigt. Die Verteilung ist nahezu log-normal. Da es weniger Datenpunkte gibt, ist die Streuung größer. Außerdem sind die Antwortzeiten generell länger, der Median liegt bei $\tau_m \approx$3000 ms.

8.6.4 Allgemeine Diskussion der Log-Normal-Form.

Die oben gezeigten Verteilungen sind Beispiele für Verteilungen von $\ln \tau$. Bei der Auswahl wurde auf typische Beispiele geachtet. Ähnliche Messungen und Auswertungen kann mann für viele Transaktionen in vielen Systemen durchführen. Sofern keine Performanceprobleme vorliegen, sehen die Resultate immer ähnlich aus.

Für eine automatische Auswertung solcher Daten ist wichtig, ein quantitatives Maß für die Abweichung von einer Normalverteilung zu haben. Dann kann man automatisch feststellen, für

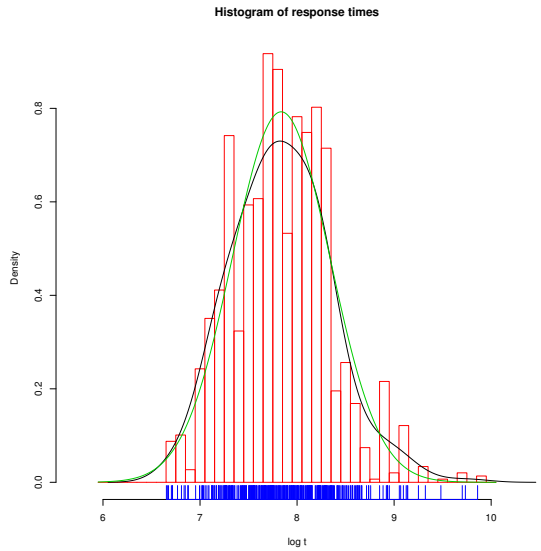

Abbildung 25: Transaktion ZXXX, System 1: Für Details siehe Abb. 19. Basis: 1,722 Datenpunkte.

welche Transaktionen in einem System die Verteilung von $\ln \tau$ von einer Normalverteilung abweicht.

Ein solches Maß sind die höheren Kummulaten κ_n, $n \geq 3$ der Verteilung. Sie lassen sich leicht definieren. Für eine Verteilung $p(x)$ ist

$$\phi(t) = \int dx\, p(x) \exp(itx) \tag{17}$$

die charakteristische Funktion. Den Logarithmus der charakterischen Funktion kann man in einer Reihe entwickeln

$$\ln \phi(t) = \sum_{n=0}^{\infty} \kappa_n \frac{(it)^n}{n!} \tag{18}$$

deren Koeffizienten die Kummulanten der Verteilung sind (siehe [5], Nummer 26.1.12). Die dritte und alle höheren Kummulaten verschwinden für eine Normalverteilung. Die Größe der dritten und aller höheren Kummulanten sind also direkt ein Maß für die Abweichung einer Verteilung von der Normalform. Die Fouriertransformierte einer Gaußfunktion ist wieder eine Gaußfunktion. Deren Logarithmus ist ein Polynom zweiten Grades.

Statt der Kummulanten betrachtet man oft die normierten Kummulanten $c_n = \kappa_n / \kappa_2^{n/2}$. Sie sind mit der Breite der Verteilung normiert und daher dimensionslos. c_3 heißt Schiefe, c_4 heißt Kurtosis der Verteilung. In Abb. 26 zeigen wir für eine große Menge von Transaktionen aus verschiedenen Systemen einen Plot von c_3 vs. c_4 für die gemessenen Verteilungen von $x = \ln \tau$. Für die meisten Verteilungen sind die Parameter klein, d.h $|c_3| \lesssim 1$ und $0 \lesssim c_4 \lesssim 1.5$. Es kommen aber auch größere Abweichungen vor. Negative Werte von c_4 treten meist dann auf, wenn die

Verteilung zwei oder mehrere Maxima hat, wie z.B. in Abb. 24. Es gibt außerdem eine klare Tendenz zu einer positiven Schiefe. Das bedeutet, daß in vielen Fällen das statistische Gewicht für längere Antwortzeiten größer ist als es bei einer Log-Normal-Verteilung zu erwarten wäre. In den meisten Fällen in 26 konnte die Abweichung von der Log-Normal-Form auf ein Performanceproblem zurückgeführt werden.

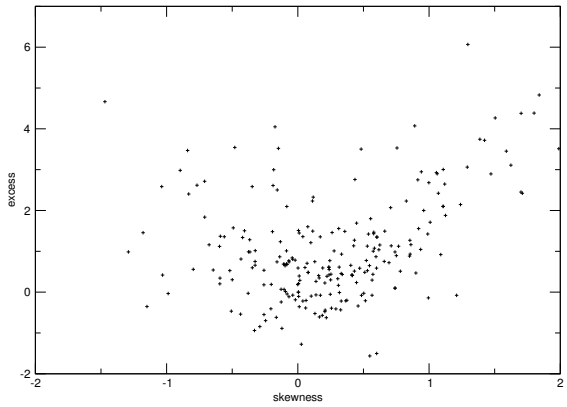

Abbildung 26: Normierte dritte und vierte Kummulante (skewness vs. excess) für 232 Antwortzeitverteilungen aus 21 verschiedenen SAP R/3 Systemen.

Man kann versuchen, die Daten statt mit einer Log-Normal-Verteilung mit einer anderen Verteilung zu fitten, um zu prüfen, ob die Resultate besser werden. Das ist immer möglich, wenn man mehr Fitparameter zuläßt. Versuche in diese Richtung wurden durchgeführt, bisher aber mit wenig Erkenntnisgewinn.

8.6.5 Diskussion des Modells

Ein wesentlicher Punkt für die Motivation unseres Modells war die Vernachlässigung von Korrelationen der Parameter, von denen $\ln \tau$ abhängt. Ein zweiter wichtiger Punkt war, daß es keine additiven Beiträge gibt.

Für die Antwortzeit selber lassen sich diese Punkte mit guten Argumenten begründen. Wenn man dagegen nur die CPU-Zeit oder nur die Datenbankzeit betrachtet (als Beispiele), dann gelten die Argumente nicht mehr. Für solche Zeiten ist die Zahl der Parameter, die darauf Einfluß haben, viel kleiner, und diese Parameter werden voneinander abhängen. Der zentrale Grenzwertsatz ist dann nicht mehr anwendbar. Das kann man testen. In Abb. 27 zeigen wir die CPU-Zeit, in Abb. 28 die Datenbankzeit für die Transaktion VA01 in System 1. Für diese Transaktion war die Verteilung der Antwortzeit in guter Übereinstimmung mit einer Log-Normal-Verteilung. Die Abbildungen zeigen deutlich, daß das für die CPU- und die Datenbanbkzeit nicht gilt.

Außerdem sind Datenbankzeit und CPU-Zeit korreliert. Um das zu sehen, zeigen wir einen Scatterplot, in dem die Datenbankzeit gegen die CPU-Zeit aufgetragen ist, siehe Abb. 29.

Abb. 29 zeigt deutlich zwei Regionen mit unterschiedlichem Verhalten, die auch in Abb. 28 als zwei Maxima deutlich zu sehen waren.

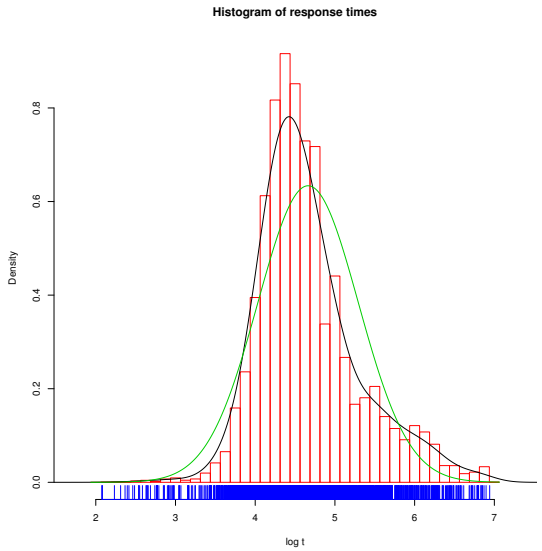

Abbildung 27: Transaktion VA01, System 1: Die Abbildung zeigt Datenpunkte für die CPU-Zeit. Im übrigen entspricht sie Abbildung 19.

9 Modellierung von SAP-Systemen

Wenn man sich mit Modellen beschäftigt, sollte man folgende Punkte im Hinterkopf haben:

- Alle Modelle sind falsch.
- Manche Modelle sind besser als andere.
- Man kann nie wissen, ob ein Modell korrekt ist.
- Je einfacher ein Modell ist, desto besser.

Wir haben bereits zweimal in der Vorlesung mathematische Modelle kennengelernt, die Teilaspekte eines SAP-Systems oder einer Landschaft modellieren:

- Kostenmodelle in Abschnitt 6.4,
- Modelle zur Performance einzelner Transaktionen in Abschnitt 8.6.

In beiden Fällen ging es darum, Aspekte eines einzelnen Systems quantitativ besser zu verstehen.

Andere Modelle, die wir diskutiert haben, waren Prozess- und Datenmodelle (nur sehr rudimentär in Abschnitt 2) sowie Modelle für den Betrieb von Systemen in Abschnitt 5. Ziel dieser Modelle war es, bestimmte Aspekte anschaulich darzustellen. Hier ging es also nicht um das quantitative Verständnis, sondern um das qualitative Verständnis bestimmter Aspekte.

In diesem Abschnitt geht es darum, weitere Versuche der mathematischen oder statistischen Modellierung kennenzulernen, Anwendungen dieser Modelle zu verstehen, aber auch die Probleme, mit denen solche Modelle behaftet sind. Ziel ist dieser Modelle ist eine quantitative Beschreibung verschiedener Aspekte von SAP-Systemen.

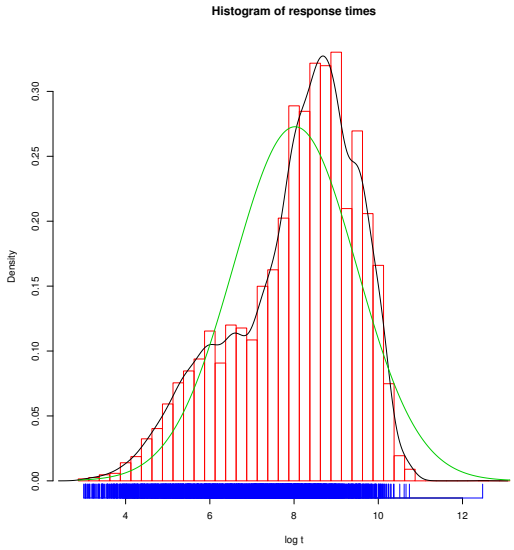

Abbildung 28: Transaktion VA01, System 1: Die Abbildung zeigt die Datenbankzeit, im übrigen entspricht sie Abbildung 19

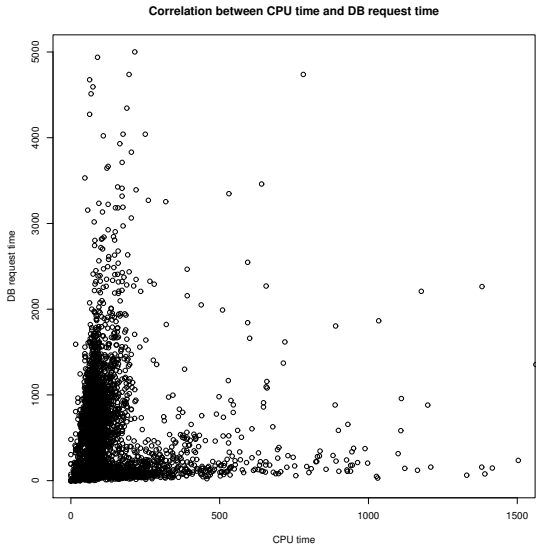

Abbildung 29: Transaktion VA01, System 1: CPU-Zeit vs Datenbankzeit

9.1 Dynamische Modellierung

Ein SAP-System ist ein dynamisches System. Da das zeitliche Verhalten nicht deterministisch abläuft, handelt es sich um ein stochastisches, dynamisches System. Ein weiterer Aspekt sind die unterschiedlichen Zeitskalen, die in diesem System eine Rolle spielen. Das ist in Abb. 30 veranschaulicht.

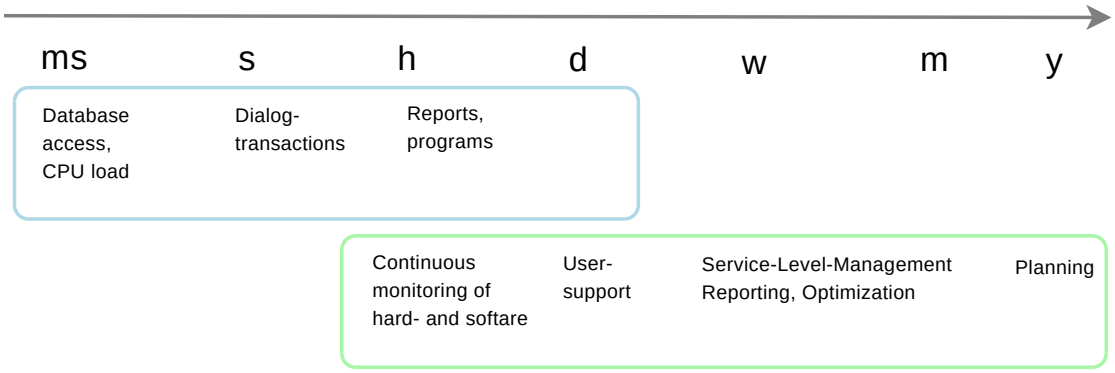

Abbildung 30: Verschiedene Zeitskalen, die in einem SAP-System eine Rolle spielen. Der blaue Kasten enthält Aspekte, die die Hard- und Softwarekomponenten betreffen, der grüne Kasten die Komponenten, die Services betreffen.

Ein vollständiges mathematisches Modell eines SAP-Systems müßte alles diese Zeitskalen umfassen. Es müsste also die Dynamik verschiedener Aspekte des Systems auf unterschiedlichen Zeitskalen beschreiben können. Dabei ist jeder einzelne Aspekt bereits sehr komplex. Beispielsweise haben wir gesehen, daß bereits die Modellierung der Kosten einen mathematisch anspruchsvollen Ansatz benötigt, wenn man ein realistisches Modell haben möchte, daß alle relevanten Kostenstrukturen abbilden kann. Auch die Modelle für Servicestrukturen (ITIL) sind komplex.

Eine vollständige Modellierung aller Aspekte eines Systems ist deshalb unrealistisch. Trotzdem ist es sinnvoll, sich der Zusammenhänge klar zu sein. Bestimmte Aspekte oder Resultate von Modellen einzelner Teilaspekte werden einen Einfluß auf die Modellierung anderer Teilaspekte haben.

9.2 Ziele der Modellierung

Ein ganz wichtiger Punkt, der beachtet werden muß, bevor man ein Modell entwickelt, ist das Ziel, das man damit verbindet: Was ist die konkrete Fragenstellung, die untersucht werden soll?

Wir haben in Abschnitt 6.4 Kostenmodelle diskutiert. Ziel der Kostenmodelle war, eine vollständige Beschreibung der Kosten eines Systems oder einer Landschaft zu bekommen, um damit Transparenz für den Inhaber des Systems zu erreichen.

Das in Abschnitt 8.6 diskutierte Modell zur Antwortzeitverteilung von Transaktionsschritten einzelner Transaktionen hatte zum Ziel, Performanceprobleme einzelner Transaktionen zu identifizieren.

Weitere wichtige Ziele für Modelle von SAP-Systemen betreffen das Verständnis der Entwicklung dieser Systeme. Typische Fragestellungen sind:

- Mit welcher Nutzung des Systems habe ich in Zukunft zu rechnen?
- Wie wird sich die Systemlast und die Performance entwickeln?
- Welche Investitionen sind daher in Zukunft zu tätigen, um die Qualität des Systems, beschrieben durch SLAs, zu halten?

Diese drei Fragen machen einen wichtigen Aspekt deutlich: Die drei Punkte Nutzung, Qualität und Kosten lassen sich nicht einfach trennen, wenn man die Dynamik auf längeren Zeitskalen betrachtet.

9.3 Statistische Modelle zur Nutzung von Systemen und Ressourcen

In den letzten beiden Abschnitten haben wir gelernt, daß sich die Nutzung und die Performance von Systemen im Detail vermessen läßt und daß sich diese Meßdaten über längere Zeiträume zumindest in aggregierter Form aufzeichnen lassen. In diesem Abschnitt wird es darum gehen, diese Daten für statistische Modelle zu nutzen.

9.3.1 Einfache statistische Modellierung

Ziel einer statistischen Modellierung ist es, zu einer guten Beschreibung der vorliegenden Daten mit statistischen Methoden zu kommen. Ziel ist es, zu einem minimalen, adequaten Modell zu kommen. Minimal bedeutet, daß in das Modell möglichst wenig Annahmen und Parameter eingehen. Adequat bedeutet, daß das Modell die Daten mit einer hinreichenden Genauigkeit beschreibt.

Man unterscheidet verschiedene Arten von statistischen Modellen. Für das Verständnis genügen vier Typen:

- Das gesättigte Modell. Es ist ein Modell, bei dem man für jeden Datenpunkt einen Parameter hat. Es beschreibt die Daten optimal, hat aber keinen Erklärungswert. Es wird in der Regel nicht betrachtet.
- Das Nullmodell. Es beschreibt die Daten durch einen Parameter, den Mittelwert. Es hat keinen Erklärungswert. Jedes andere Modell soll die Daten signifikant besser beschreiben.
- Das maximale Modell. Es enthält p Faktoren. Wechselwirkungen und Kovarianten können wichtig sein. Hier gehen alle Faktoren ein, die für die Erklärung der Daten in Frage kommen. Welche das sind, hängt von den Daten selbst ab. Es können z.B. nur solche Daten verwendet werden, die vorher erfasst wurden.

- Das minimale, adequate Modell. Es enthält $p' \leq p$ Faktoren und typischerweise weniger Wechselwirkungen und Kovarianten als das maximale Modell, aber so viele, daß das Weglassen weiterer Parameter zu einer signifikant schlechteren Beschreibung der Daten führt.

Es gibt keine festgelegte Vorgehensweise, wie man zu einem minimalen, adequaten Modell kommt. Es ist sogar so, daß es im Einzelfall nicht nur ein minimales, adequates Modell geben wird.

Ein typische Weg zu einem minimalen, adequaten Modell in der Statistik ist Ockhams Rasiermesser (occams razor) oder das Sparsamkeitsprinzip (parsimony). Es geht auf den Englischen Philosophen William von Ockham (1285–1347) zurück besagt, daß von zwei Theorien, die einen Sachverhalt erklären, die einfachere zu wählen ist. Umberto Ecco hat Wilhelm von Ockham in der Gestalt des William von Baskerville ein Denkmal in seinem Roman 'Der Name der Rose' gesetzt. Wilhelm von Ockham hat das Prinzip selbst nicht explizit formuliert, aber in seinen Schriften immer wieder verwendet. Es wurde ihm später zugeschrieben.

Für die Statistik bedeutet dieses Prinzip:

- Modelle sollen so wenig Parameter wie möglich haben.
- Modelle sollten so wenig Faktoren wie möglich haben.
- Lineare Modelle werden gegenüber nicht-linearen Modellen bevorzugt. (Randbemerkung: Ein lineares Modell bedeutet nicht, daß es einen linearen Zusammenhang zwischen zwei Größen geben muß.)
- Modelle ohne Extremwerte sind solchen mit Extremwerten zu bevorzugen.
- Modelle ohne Wechselwirkungen sind solchen mit Wechselwirkungen zu bevorzugen.

Die Vorgehensweise ist die folgende: Man startet mit einem maximalen Modell. In diesem Modell werden die Parameter mit der niedrigsten Signifikanz sukzessive weggelassen, solange, bis nur noch signifikante Parameter übrig bleiben. Dabei gibt es unter Umständen eine Reihe von Fallstricken, die zu beachten sind. Für Details verweise ich auf die einschlägige Literatur zur Statistik.

Es gibt eine Reihe sehr guter Werkzeuge zur statistischen Modellierung. Bekannte und sehr umfangreiche kommerzielle Programme sind SAS oder SPSS, neben vielen anderen zum Teil weniger umfangreichen oder besonders spezialisierten. Ein sehr gutes quelloffenes Programm ist R, das in vielem den kommerziellen Programmen in nichts nachsteht oder ihnen sogar überlegen ist. R ist gleichzeitig ein Statistikprogramm und eine funktionale Programmiersprache. Für R gibt es eine nahezu unüberschaubare Menge von Modulen, die die Sprache erweitern und für bestimmte Einsatzgebiete verwendbar machen. Ich verwende im folgenden zum Teil die Syntax von R zur Notation. Eine gute Einführung in R ist neben vielen anderen [6], eine detaillierte Beschreibung von R als funktionaler Programmiersprache, nur geeignet für Fortgeschrittene, liefert [7].

9.3.2 Dialognutzung

Ein ERP-System wird in erster Linie dazu eingesetzt, die Geschäftprozesse eines Unternehmens zu unterstützen. Dazu werden die Geschäftsprozesse in kleine Einheiten, Transaktionen, zerlegt, und diese Transaktionen werden in dem System elektronisch unterstützt und im Dialog ausgeführt. Deshalb liegt es nahe, zunächst die Dialognutzung eines Systems zu untersuchen.

Als erstes Beipiel betrachten wir deshalb die Dynamik der Dialognutzung eines Systems. Ziel ist also eine Beschreibung der Zahl der Dialogschritte $n(t)$ als Funktion der Zeit. Dabei ist es wichtig, eine der Fragestellung entsprechend vernünftige Diskretisierung der Zeitskala vorzunehmen. Wir haben gesehen, daß die Dialognutzung eines Systems innerhalb eines Tages variiert. Daher liegt es nahe, eine Diskretisierung auf Stundenbasis vorzunehmen.

Der Einfachheit halber nehmen wir an, daß sich die Dialoglast eines Systems allein aus den Geschäftsprozessen des Unternehmens bestimmt und nicht von dem System abhängt. Das ist dann der Fall, wenn es keine Restriktionen des Systems gibt, die auf die Dialognutzung zurückwirken, z.B. Performanceengpässe. Wir werden also die Zahl der Dialogschritte pro Stunde über einen ausreichend langen Zeitraum messen. Das Resultat $n(t)$ werden wir versuchen, so sinnvoll wie möglich mit Hilfe eines statistischen Modells zu beschreiben. Ein maximales Modell für $n(t)$ enthält nach diesem Ansatz also nur einen Faktor, t, aber noch beliebig viele Parameter.

Wir schränken weiter ein, indem wir von folgenden Annahmen ausgehen:

- Es gibt eine typische mittlere Nutzung.
- Die Nutzung ist an Werktagen und an Wochenenden und Feiertagen unterschiedlich.
- Es kann eine Periodizität mit spezieller Nutzung an Monatswechseln geben.
- Es kann eine Periodizität mit spezieller Nutzung an Quartalswechseln geben.
- Es kann eine Periodizität mit einem speziellen Verlauf über ein Jahr hin geben.
- Es kann innerhalb der Werktage einer Woche eine unterschiedliche Nutzung geben (Wochenprofil).
- Es kann innerhalb der Stunden eines Tages eine unterschiedliche Nutzung geben (Tagesprofil).
- Es kann eine generelle Tendenz (Anstieg, Abfall) geben.

In einer Formel ausgedrückt, bedeutet das einen Ansatz

$$n(t) = n_0 + n_{wd} i_{wd}(t) + n_m i_m(t) + n_q i_q(t) + n_y i_y(t) + n_{wp} i_{wp}(t) + n_{dp} i_{dp}(t) + n_t i_t(t) \qquad (19)$$

Das Modell enthält acht Parameter und sieben Funktionen von t, die vorgegeben sind. Sie werden folgendermaßen festgelegt:

- $i_{wd}(t)$ ist 1 an Werktagen, 0 sonst.
- $i_m(t)$ und $i_q(t)$ sind Funktionen, die 1 in der Nähe eines Monats- oder Quartalswechsels sind, 0 sonst. Typischerweise werden diese Funktionen eine Breite von zwei oder drei Werktagen haben.

- $i_y(t)$ ist eine charakteristische Funktion, die entweder besondere Aktivitäten am Jahreswechsel berücksichtigt oder eine generelle Tendenz über das Jahr hinweg festlegt.
- $i_{wp}(t)$ ist ein Wochenprofil, $i_{dp}(t)$ ist ein Tagesprofil. Die Tagesprofile und Wochenprofile kann man aus den Daten bestimmen, indem man über den kompletten Meßzeitraum die mittlere Zahl von Dialogschritten pro Stunde oder pro Wochentag bestimmt und mit der Gesamtzahl gewichtet.
- $i_t(t)$ beschreibt ein Wachstum, also $i_t(t) = t$ oder $i_t(t) = \exp(t)$.

In R würde man dieses Modell durch die Formel

$$lm(n \sim i_{wd} + i_m + i_q + i_y + i_{wp} + i_{dp} + i_t) \tag{20}$$

notieren. Ob $i_q(t)$ und $i_y(t)$ überhaupt auftreten, hängt von der Länge des Meßzeitraums ab.

Wenn man so vorgeht und von einem Meßzeitraum von drei Monaten ausgeht, enthält dieses Modell 6 Parameter. Es gibt pro Stunde einen Meßwert, also 2.160 Meßwerte. Die sechs Parameter lassen sich also sehr gut bestimmen.

Die Statistik liefert für jeden Koeffizienten zusätzlich Informationen darüber, wie signifikant der jeweilige Koeffizient ist. Dazu werden in der Statistik verschiedene Test verwendet: F-Test und F-Verteilung, t-Test, etc . R (ebenso andere Programme) liefern automatisch diese Informationen mit. In unserem Fall ist (19), (20) das maximale Modell. Werden sukzessive nicht-signifikante Koeffizienten weggelassen, so erhält man schließlich das minimale adequate Modell.

Sei $\delta(t)$ die Differenz des Modells von den Meßwerten. Für diese Größe können Korrelationen $\langle \delta(t)\delta(t')\rangle$ bestimmt werden. In den meisten Fällen werden die Korrelationen zu verschiedenen Zeiten klein sein. Ist das nicht der Fall, fehlt möglicherweise im Modell ein relevanter Term. Die mittlere Streuung $n_s = \sqrt{\langle \delta(t)\delta(t)\rangle}$ berücksichtigt man in dem Modell durch einen Rauschterm

$$n(t) = n_0 + n_{wd} i_{wd}(t) + n_m i_m(t) + n_q i_q(t) + n_y i_y(t) + n_{wp} i_{wp}(t) + n_{dp} i_{dp}(t) + n_t i_t(t) + n_s \xi(t) \tag{21}$$

Dabei ist $\langle \xi(t)\rangle = 0$, $\langle \xi(t)\xi(t')\rangle = \delta_{t,t'}$. (21) ist das gesuchte Modell für die Zahl der Dialogschritte.

In ähnlicher Weise kann man z.B. auch die Zahl der Benutzer modellieren. Bezeichnet man mit $B(t)$ die Zahl der Benutzer, dann stellt sich aber die Frage, ob

$$lm(B(t) \sim n(t) + \ldots)$$

als Ansatz für ein Modell taugt und welche Form die anderen Terme in diesem Modell haben werden. In vielen Fällen ist dieser erste Term bereits ausreichend.

Entsprechende Modelle kann man auch für bestimmte Klassen von Benutzern oder von Transaktionen aufstellen. Welche Klasse wichtig ist, hängt vom Einzelfall ab.

9.3.3 Nutzung von Ressourcen

Eine Frage ist häufig, wie Ressourcen, z.B. CPUs oder Platten genutzt werden. Diese Informationen sind wichtig, wenn man für Systeme neue Hardware beschaffen zu müssen. Besonders interessant sind Situationen, in denen durch Virtualisierung, also dadurch, daß man mehrere Systeme auf virtuellen Servern auf einem großen Rechner laufen läßt, Investitionen gespart werden können.

In einer virtualisierten Umgebung werden den virtuellen Servern die Ressourcen nach Bedarf zugeordnet. Die Zuordnung erfolgt mit einer Taktrate von 10ms. Schwankungen der CPU-Last mit einer Zeitskala von 10ms treten auch in Systemen ohne Virtualisierung auf.

Das bedeutet aber nicht, daß es in dem zugrunde liegenden statistischen Modelle Terme gibt, die von dieser Zeitskala abhängen. Typischerweise wird es so sein, daß lediglich der stochastische Term in den Modellen auf dieser Zeitskala variiert, während sich alle anderen Terme nur auf deutlich längeren Zeitskalen verändern.

Für die Nutzung von Ressourcen wird man also einen zu (21) ähnlichen Modellansatz machen. Wichtige Unterschiede zur Dialognutzung bestehen aber trotzdem, weil ein großer Teil (typisch mehr als 50%) der Nutzung von Ressourcen nicht vom Dialog stammt.

Ein Ansatz besteht hier darin, die Nutzung der Ressourcen in zwei Anteile zu zerlegen, die vom Dialog verursachte Last und die von anderen Prozessen verursachte Last (Schnittstellen, Batch). Der Dialoganteil folgt im wesentlichen $n(t)$, der zweite Anteil ist davon meist unabhängig. Auch eine Zerlegung in mehr Anteile kann sinnvoll sein.

Auf diese Weise kann man zu sehr präzisen Modellen für die Nutzung von Ressourcen kommen.

9.4 Nutzung und Performance

Im vorangegangenen Abschnitt haben wir angedeutet, wie man den Zusammenhang zwischen der Dialognutzung eines Systems, also dem Verhalten der Benutzer, und der Ressourcennutzung modellieren kann. Wesentlich komplexer ist es, den Zusammenhang von Nutzung und Performance zu modellieren.

Ausgangspunkt dafür kann die Modellierung in Abschnitt 8.6 sein. Wir können zunächst vereinfachend annehmen, daß das dort für einzelne Transaktionen vorgestellte Modell in gleicher Form auch für komplette Systeme gilt.

Wir haben dort gesehen, daß in performanten Systemen die ersten beiden Kummulaten der Verteilung für den Logarithmus der Antwortzeit ausreichend sein können. Aus (17), (18) erhält man, wenn $\kappa_n = 0$ für $n > 2$,

$$p(\ln\tau)d\ln\tau = \frac{1}{\sqrt{2\pi\kappa_2}}\exp(-\frac{1}{2\kappa_2}(\ln\tau-\kappa_1)^2)d\ln\tau$$

Für nicht-performante Systeme kann man die ersten vier Kummulaten betrachten.

In der Argumentation in Abschnitt 8.6 war wichtig, daß die Parameter, die die Antwortzeit bestimmen, langsam veränderlich sind. In der Tat ist es in den meisten Fällen ausreichend, eine Modellierung von Nutzung und Performance so durchzuführen, daß man die Kummulanten langsam zeitveränderlich wählt, also im einfachen Fall eine zeitabhängige Verteilung bekommt

$$p(\ln\tau,t)d\ln\tau = \frac{1}{\sqrt{2\pi\kappa_2(t)}}\exp(-\frac{1}{2\kappa_2(t)}(\ln\tau-\kappa_1(t))^2)d\ln\tau$$

Für die Zeitabhängigkeit der Parameter genügt in der Regel eine Zeitskala von Stunden oder Tagen. Wir können diese Parameter für jede Stunde aus den Vermessungen der Systeme bestimmen und anschließend über eine statistische Modellierung diese Parameter mit den Nutzungsparametern zu verknüpfen. Auf diese Weise erhält man schließlich einen Zusammenhang zwischen Nutzung und Performance.

Diese Modelle haben natürlich erhebliche Probleme. Der wichtigste Aspekt ist, daß diese Modelle überhaupt nur dann gut anwendbar sind, wenn die Performance des Systems nicht bereits schlecht ist. In die Modellbildung sind eine Reihe von Annahmen eingeflossen. Die wichtigste war, daß genügend Ressourcen zur Verfügung stehen. Das ist aber nicht notwenigerweise gegeben, insbesondere dann nicht, wenn die Nutzung der Systeme wächst. Selbst wenn die Nutzung nicht wächst, gibt es immer einige Parameter, die wachsen, z.B. das Volumen der Datenbank. Es besteht somit immer latent die Gefahr, daß das System bei mindestens einer Ressource an eine Grenze stößt.

Wichtig ist in diesem Zusammenhang, daß SAP Systeme, wenn man sie dynamisch beschreibt, nicht-lineare Systeme sind. Es gibt Wechselwirkungen. Diese Wechselwirkungen sind zum Teil kurzreichweitig, so z.B. Sperren, die auf der Datenbank gesetzt werden, zum Teil sind sie langreichweitig, so z.B. die Beschränkung der Ressourcen des Systems.

Systeme mit Wechselwirkungen und beschränkten Ressourcen zeigen häufig nahe ihrer Ressourcengrenze ein Umkippen. Das kann man an einem ganz einfachen Modell veranschaulichen. Wir betrachten ein Gittermodell mit drei Gitterplätzen, 1 , 2 und 3. Durch das Gitter diffundieren Teilchen. Die Teilchen werden bei 1 in das Gitter eingespeist. Sie bewegen sich möglicherweise mehrfach zwischen 1 und zwei hin und her. Wenn ein Teilchen sich auf den Gitterplatz 3 bewegt, wird es wieder aus dem Sysem entfernt. Die Teilchen modellieren Transaktionen, die Gitterplätze sind verwendete Ressourcen. Die Antwortzeit ist die Aufenthaltsdauer des Teilchens in dem Gitter. Die beschränkte Ressource besteht darin, daß sich nur eine begrenzte Anzahl von Teilchen auf dem Gitterplatz 2 befinden darf, z.B nur eines. Solange die Rate, mit der Teilchen in das System eingespeist wird, klein ist, läuft jedes Teilchen nahezu ungehindert durch das System, verhält sich also wie ein freies Teilchen. Die Antwortzeit hängt nur von den Hüpfraten zwischen den Gitterplätzen ab. Erhöht man die Rate, so kommt es wegen der Wechselwirkung auf dem Gitterplatz 2 zu längeren Antwortzeiten. Die Rate, mit der Teilchen das System verlassen, hängt von der Rate ab, mit der sie eingespeist werden, und von der Antwortzeit. Ab einer bestimmten Rate kippt das System: Die Rate, mit der Teilchen das System verlassen, wird dann kleiner als die Rate, mit der sie in das System eingespeist werden, und es kommt zu einem Stau. Die Antwortzeit wird unendlich lang.

Natürlich ist dieses einfache Modell für ein Transaktionssystem mit einer beschränkten Ressource nur eine Karrikatur eines realen Systems. Die wesentliche Eigenschaft, nämlich das Umkippen wegen Ressourcenbeschränkung, tritt aber in großen Systemen genauso auf. Hier gibt es nur verschiedene Formen von Beschränkungen, die sich gegenseitig beeinflußen und die zum Teil regulierenden Charakter haben:

Hardware: Jede Ressource ist vielfach vorhanden, außerdem erlaubt das Betriebssystem, Ressourcen quasi mehrfach zu belegen. Das kann aber auch Nachteile haben, weil sich dadurch die mittleren Laufzeiten verlängern.

Konfiguration: Die Zahl der Workprozesse und damit die Zahl der parallel ausführbaren Transaktionen ist beschränkt. Dadurch kann es zwar zu Wartezeiten kommen, dadurch wird aber eine Überbelegung der Hardwareressourcen verhindert. Natürlich entsteht dadurch aber auch eine neue Beschränkung.

Software: In der Software sind etliche Mechanismen zur Steuerung implementiert, die verhindern sollen, daß sich Beschränkungen negativ auswirken. Diese Regelungsmechansimen führen aber zu neuen Rückkopplungen, die sich ihrerseits wieder negativ auswirken können. Aus Gründen der Datensicherheit sind einige Ressourcen softwareseitig nur einfach ausgelegt, damit entstehen durch die Software neue Beschränkungen. Prinzipiell sind aber alle Softwareressourcen beschränkt, weil alle Zähler, Indizes, etc. beschränkt sind. Meist sind sie so groß gewählt, daß sich diese Beschränkungen nicht in der Praxis auswirken.

Beschränkungen jeder Art sind letztlich Nicht-Linearitäten. Nicht-lineare dynamische Systeme können Bifurkationen oder auch chaotisches Verhalten zeigen. Durch die Stochastizität auf der einen Seite, durch Dämpfungsmechanismen (siehe oben im Bereich Software und Konfiguration) auf der anderen Seite wird das System stabilisiert. Trotzdem können Probleme, die zu einem 'Umkippen' des Systems führen, nicht vollständig ausgeschlossen werden.

Fälle eines 'Umkippens' sind selten, kommen aber vor. Für Informatiker ist z.B. das klassische *dead lock* ein solches Phänomen. Deshalb gibt es viele Mechanismen, die dieses verhindern sollen. Es gibt aber andere Beispiele:

Ein langlaufender, Batchprozess (Eigenentwicklung) setzt eine Sperre in einem Bereich der Datenbank. Nutzer wollen auf diesen Bereich zugreifen. Die durch die Dialogtransaktionen ausgelösten Verbucher-Schritte können wegen der Sperre nicht ausgeführt werden, es kommt zu einem Stau in der Verbucherwarteschlange. Diese läuft über, daraufhin ist das System für Dialogbenutzer nicht mehr verwendbar. In einer solchen Situation ist die Hardware nur sehr schwach ausgelastet, daß System ist technisch gesehen erreichbar, aber für viele Benutzer praktisch nicht verwendbar.

Ziel einer Modellierung kann sein, solche Engpässe frühzeitig zu identifizieren und so Probleme zu verhindern.

9.5 Kosten, Nutzung und Performance

Schwieriger ist es, Modelle zu entwickeln, in denen Kosten, Nutzung und Performance modelliert werden. So etwas ist beispielsweise interessant, wenn man

- IT-Kosten verrechnen will,
- Vergleiche zwischen Systemen anstellen will (nächster Abschnitt) oder
- wenn Systeme zusammengelegt oder aufgetrennt werden sollen.

Ein wichtiger Aspekt ist die Frage, wie Kosten von Nutzung und Performance abhängen. Für diese Frage spielt die Dynamik der Kosten in der Regel keine Rolle, wohl aber das dynamische Verhalten der Nutzung und der Performance.

9.5.1 Einfache Modelle

Einfache Modelle haben wir bereits im Abschnitt 6.5 diskutiert. Sie arbeiten in der Regel mit Stückkosten. Dabei werden für unterschiedlichen Ressourcen unterschiedliche Stückkosten verwendet.

Für Hardwareressourcen sind folgende Stückkosten üblich

Kosten pro SAPS. SAPS ist eine im SAP-Umfeld übliche Einheit zur Beschreibung der Stärke eines Rechners. Wieviel SAPS ein Rechner hat, hängt von der Stärke der CPU, der Anzahl der CPUs und einigen weiteren Parametern ab. Die SAPS-Zahl wird von Hardwareherstellern für bestimmte Konfigurationen experimentell bestimmt, dazwischen wird interpoliert. Serverkosten werden häufig als Kosten pro SAPS verrechnet.

Kosten pro MB Hauptspeicher. In dem Preis pro SAPS ist häufig der Hauptspeicher enthalten. In etwas differnzierteren Modellen ist darin ein fester Speicher enthalten. Benötigt ein System mehr Hauptspeicher, entstehen dafür zusätzliche Kosten.

Kosten pro GB Plattenplatz. Diese Kenngröße ist einfach. Häufig wird hier aber noch differenziert nach

- RAID-Level
- gespiegelt/nicht gespiegelt

Kosten pro GB Backup. Hier wird noch differenziert nach

- Art und Häufigkeit des Backups
- Länge der Lagerzeit.

Mit diesen Preisen hat man den großen Teil der Hardwarekosten abgedeckt. Es verbleiben Kosten für Strom, Klima, Netzwerke, etc. Diese werden in der Regel pauschal pro Server oder pro System abgerechnet. Es gibt also

Kosten pro Server. Hier sind in der Regel nicht nur Kosten für Strom, etc., sondern vor allem Betreuungs- und Administrationskosten auf RZ-Seite enthalten.

Kosten pro System. Hier sind in der Regel nicht nur Kosten für Strom, etc., sondern vor allem Betreuungs- und Administrationskosten auf SAP-Basis-Seite enthalten.

Mit den bisher genannten Preisen sind alle Kosten bis Oberkante SAP-Basis verrechenbar. Für die Kosten für den Applikationsbetrieb gibt es

Kosten pro Ticket. Hier wird ggf. noch nach der Art des Tickets unterschieden.

Kosten pro Benutzer. Hier können verschiedene Klassen von Benutzern je nach Art und Umfang der Nutzung unterschieden werden. Typisch sind zwei oder drei Klassen, die einfach nach der Nutzungsintensität unterschiedlich sind. In den Kosten pro Benutzer werden auch Lizenzkosten enthalten sein.

Diese einfachen Kostenmodelle sind linear. Dahinter steht die Vorstellungen, daß Kosten additiv sind. Das ist aber nicht richtig, wenn man an Vergleiche zwischen Systemen denkt. Hier gibt es fast immer den Effekt, daß Stückkosten für kleine Systeme höher sind als für große Systeme. Ein weiteres Problem bei Preismodellen ist, daß immer wieder unterschiedliche Komponenten zu einem Paket zusammengestellt sind, für das dann ein Preis festgelegt wird. So sind Kosten pro SAPS je nach Unternehmen unterschiedlich, weil unterschiedliche Komponenten verrechnet werden. Damit hat man keine Vergleichbarkeit mehr.

9.5.2 Statistische Kostenmodelle

Man kann das in Abschnitt 6.4 beschriebene Verfahren verwenden, um zu Kosten für Kostenelemente zu kommen, die für jedes System oder jede Systemlandschaft identisch zusammengesetzt sind. Hat man die Systeme vermessen, kann man schließlich mit Hilfe statistischer Modellierung versuchen, Abhängigkeiten von Kosten zu Nutzungs- und Performancedaten zu gewinnen. Dabei wird man in der Regel auf aggregierte Nutzungs- und Performancedaten zurückgehen, die man bereits aus Modellen gewonnen hat. Zusätzlich sind die dieser Vorgehensweise weitere Qualitätsmerkmale zu berücksichtigen, die z.B. unterschiedliche Verfügbarkeiten, Ansprüche an Datensicherheit, etc. berücksichtgen. Hat man Daten von einer hinreichend großen Zahl von Systemen, so kann man daraus zu guten, statistischen Modellen kommen. Wichtig dabei ist aber, daß in der Regel in unterschiedlichen Unternehmen unterschiedlich effektiv gearbeitet wird, daß es also immer Streuungen gibt, die daher rühren, daß Einsparungspotentiale bestehen.

Es gibt eine Reihe von Problemen, die bei der statistischen Modellierung von Kosten auftreten:

- Die Zahl der Faktoren ist sehr groß. Man wird in der Regeln nicht annehmen, daß Kosten von allen Nutzungsparametern abhängen, aber selbst Untermengen, die sinnvoll erscheinen, sind groß.
- Die Zahl der Systeme, für die Kostenangaben vorliegen, ist begrenzt.
- Für einige Kostenelemente erhält man eine deutliche Abhängigkeit der Kosten von dem speziellen Geschäft des Unternehmens.
- Es gibt qualitative Faktoren, von denen die Kosten abhängen.

In der Regel wird man die Modellierung nicht für die Gesamtkosten durchführen, sondern für einzelne Komponenten des Kostenmodells. Je weiter die Komponenten in dem Schichtenmodell in Abb. 12, Seite 35 von der Anwendung entfernt sind, desto einfacher wird die Modellierung, weil die Faktoren weniger werden und die Abhängigkeiten von der Anwendung sinken.

Statistische Modelle können für spezielle Kostenelemente mit Preismodellen verglichen werden und können insbesondere wichtige Preisinformationen liefern.

Wie genau die Modellierung durchgeführt wird und wie die oben genannten Punkte berücksichtigt werden, hängt von der Fragestellung ab. Wir betrachten im nächsten Abschnitt das Beispiel der Modellierung für Vergleiche etwas genauer.

Ein wichtiger Punkt, der im folgenden eine Rolle spielt, ist, daß ein minimales statistisches Modell die relevanten Faktoren liefert, die für den Vergleich wichtig sind.

9.6 Vergleiche: Benchmarking

Modelle für Kosten, Nutzung und Performance werden vor allem für Vergleiche verwendet. Ziel ist zu beurteilen, ob Kosten für ein System in verschiedenen Bereichen adequat oder zu hoch sind im Vergleich zu anderen.

Die wesentliche Anforderung an der Vergleichsverfahren ist, Vergleichbarkeit zu garantieren. Salopp gesagt: Wir wollen nicht Äpfel mit Birnen vergleichen. Für SAP-Systeme sind dabei zwei rundlegende Fragen zu beantworten:

- Ist die betrachtete Größe für einen Vergleich geeignet?
- Sind die betrachteten Systeme gut vergleichbar?

Die zweite Anforderung hängt damit zusammen: Wir benötigen Vergleiche von unterschiedlichen Strukturen und auf unterschiedlichen Skalen in gleicher Präzision.

Konkret bedeutet das, daß wir nicht eine einzige Größe vergleichen, sondern eine bestimmte Menge verschiedener Größen, und daß wir nicht ein Kriterium für die Vergleichbarkeit verwenden können, sondern mehrere. Außerdem ist die Frage, ob sich zwei Systeme vergleichen lassen, nicht einfach mit ja oder nein zu beantworten. Wir benötigen ein Kriterium, das die Vergleichbarkeit von zwei Systemen quantifizierbar macht.

Oben haben wir darauf hingewiesen, daß Kosten von dem speziellen Geschäft des Kunden abhängen können, also von dem spezifischen Nutzungsprofil. Das kann berücksichtigt werden, indem man Systeme gewichtet, die für den Vergleich zur Verfügung stehen. Die Gewichte werden für unterschiedliche Komponenten des Kostenmodells unterschiedlich gewählt werden. Gewichte liegen z.B. zwischen 0 und 1. Das Gewicht hängt von zwei Systemen ab und beschreibt die Ähnlichkeit der beiden Systeme. Ein Gewicht von 0 bedeutet dann, daß beide Systeme nicht vergleichbar sind, Gewicht 1 bedeutet, daß die Systeme identisch sind.

Das Hauptproblem besteht darin, die Gewichte geeignet zu definieren. Dabei ist die konkrete Fragestellung zu berücksichtigen. Interessiert man sich z.B. für die Backupkosten, wird man andere Gewichte verwenden als für applikationsbezogene Kosten wie den Second-Level-Support. Die Gewichte werden in der Regel auch nicht nur von einer Größe, sondern von mehreren abhängen. Auch die unterschiedliche Qualität fließt in die Gewichte ein.

Gewichte berücksichtigen, daß verschiedene Systeme hinsichtlich verschiedener Kostenkomponenten unterschiedlich gut vergleichbar sind. Die Gewichte werden in der Modellierung berück-

sichtigt. Die Gewichte werden aber für jedes neue System, für das man Vergleiche bekommen will, neu berechnet.

Verschiedene Systeme unterscheiden sich durch unterschiedliche Nutzungs- und Performancedaten. Die Nutzungs- und Performancedaten bezeichnen wir im folgenden mit einem N-dimensionalen Vektor $X = (x_i)_{i=1...N}$. Die Dimension N des Vektorraums kann sehr groß sein, da hier alle Daten enthalten sind, die über die Vermessung der Systeme erfasst werden.

Der Ansatz für die Berechnung der Gewichte basiert auf zwei Ideen

- Für ein Kostenelement v sind nur einige Nutzungsdaten relevant. Die Relevanz beschreibe ich durch einen Projektionsoperator P_v auf dem Vektorraum X. Die relevanten Nutzungsdaten sind dann $P_v X$.
- Je ähnlicher die relevanten Nutzungsdaten $P_v X_1$ und $P_v X_2$ von zwei Systemen sind, desto höher ist das Gewicht. Das Gewicht ist 1, wenn die Nutzungsdaten identisch sind, es ist 0, wenn sie orthogonal aufeinander stehen.

Damit bietet sich für die Berechnung des Gewichts für den Vergleich der Systeme 1 und 2 für die Kostengröße v ein Ansatz der Form

$$g_{12}(v) = f(|\langle P_v X_1, P_v X_2\rangle| / \sqrt{\langle P_v X_1, P_v X_1\rangle \langle P_v X_2, P_v X_2\rangle}) \tag{22}$$

an. Hier ist $\langle .,. \rangle$ ein Skalarprodukt in dem Vektorraum der Nutzungsdaten und f eine monotone Funktion mit $f(0) = 0$, $f(1) = 1$, z.B. eine Potenzfunktion.

Die Projektionsoperatoren P_v bestimmen sich aus minimalen statistischen Modellen. Sie projizieren auf die relevanten Faktoren.

Hat man die Gewichte berechnet, die zu berücksichtigenden Faktoren identifiziert, die Modellierung durchgeführt, dann erhält man ein statistisches Modell für dieses spezielle System, daß angibt, wie die Kosten von Nutzungs- und Performancedaten abhängen. Dieses Modell kann man mit der Realität vergleichen. Bei Abweichungen zwischen Modell und Realität kann man die Ursachen dafür versuchen festzustellen:

- Sie können im Modell begründet sein, z.B. wenn es keine guten Vergleichsdaten gibt. Das ist der Fall, wenn die Gewichte durchgängig klein sind oder wenn die Streuungen groß sind. Teilweise läßt sich die Qualität der Vergleichsdaten also mit statistischen Methoden bestimmen.
- Sie können in der Realität begründet sein. Das ist dann der Fall, wenn die Kosten für das betrachtete Kostenelement für das betrachtete System zu hoch sind. Wenn das der Fall ist, ist zu untersuchen, welche Gründe in dem entsprechenden Einzelfall zu höheren Kosten führen und welche Maßnahmen die Kosten reduzieren können.

Die beschriebene Modellierung liefert mittlere Kosten für den Vergleich. Interessant für Vergleiche ist aber eher, wenn man an Optimierungen interessiert ist, der Vergleich mt Systemen, die besonders kostengünstig sind. Das ist ebenfalls mit statistischen Verfahren möglich. Eine Methode dafür ist die Data Envelopment Analyse (DEA). Da in unserem Fall Gewichte verwendet

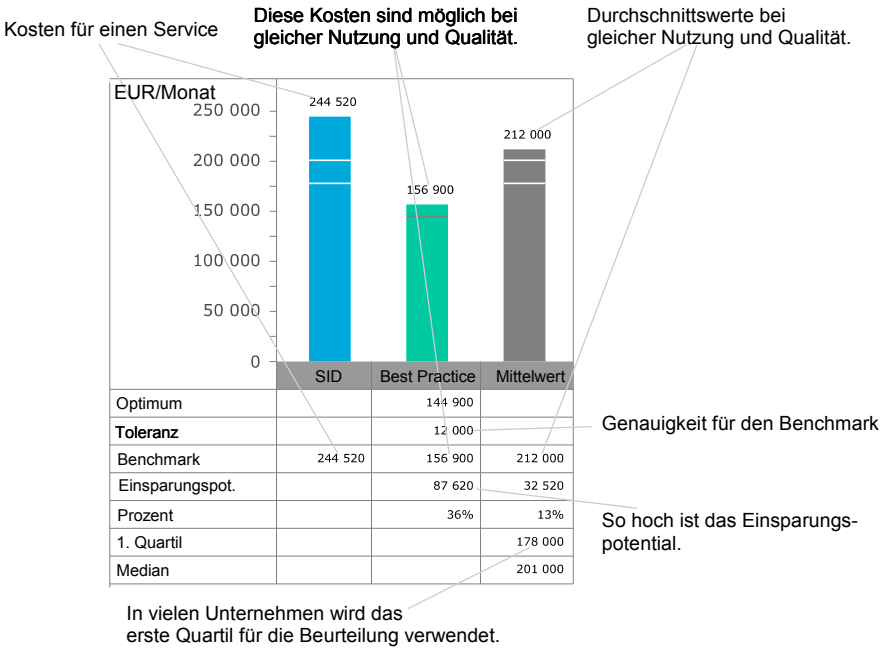

Abbildung 31: Kommentiertes Beispiel für einen Kostenbenchmark

werden und Streuungen vorkommen, muß man die klassische DEA modifizieren. Die Vorgehensweise ist sonst gleich. Sie läßt sich am einfachsten als monotone Regression beschreiben.

Das Beispiel zeigt, daß man in der Anwendung mit unterschiedlichen Vergleichszahlen arbeitet. Die in der Spalte *Best Practice* angegebenen Zahlen stammen aus der DEA. Hier ist zusätzlich eine Toleranz angegeben, die letztlich angibt, mit welcher Präzision der Vergleich möglich ist. Die in der Spalte Mittelwert angegebenen Zahlen stammen aus der normalen statistischen Modellierung. Zusätzlich zum Mittelwert sind das erste Quartil und der Median angegeben. Beide werden häufig in Zielvorgaben in Unternehmen als unternehmensinterne Vergleichsmaßstäbe verwendet.

In dem Beispiel in Abb. 31 sieht man eine deutliche Abweichung der tatsächlichen Kosten von allen Vergleichswerte. In diesem Fall wird man im nächsten Schritt nach den Ursachen für die Abweichung suchen. Die Abweichung ist so hoch, daß davon auszugehen ist, daß die Ursachen in den Kosten dieses Systems selbst zu suchen sind.

Hat man nur ein Resultat dieser Art, ist die Analyse schwierig. In der Regel hat man aber eine Fülle von Resultaten dieser Art. Jedes Resultat entspricht einem anderen Blickwinkel auf die Kosten des Systems. Aus den unterschiedlichen Blickwinkeln heraus ist die Analyse dann leichter möglich.

Ein Kostenbenchmark ist lediglich ein Werkzeug und nie das Ziel einer Analyse. Das Ziel ist, konkrete Maßnahmen festzulegen, die zu einer Reduktion der Kosten führen. Ursachen für zu hohe Kosten können in ganz unterschiedlichen Bereichen liegen, entsprechend sind die Maßnahmen immer wieder andere und sind schwer zu verallgemeinern.

Literatur

[1] Hubert Österle: Business Engineering. Springer, Berlin, Heidelberg, New York, 1995.

[2] Thomas Schneider: SAP Performanceoptimierung. Galileo Press GmbH, Bonn, 2002.

[3] ITIL The key to Managing IT Services. The Stationery Office, Norwich, 2000ff

[4] H. Risken: The Fokker-Planck Equation. Springer, Berlin, Heidelberg, New York 1989[2].

[5] M. Abramowitz, I.A. Stegun: Handbook of Mathematical Functions. Dover Publications, New York 1970[9].

[6] Michael J. Crawley: Statistics. An Introdution using R. John Wiley & Sons Ltd., Chichester 2007.

[7] John M. Chambers: Software for Data Analysis. Programming with R. Springer, New York 2008.

Tabellenverzeichnis

Abbildungsverzeichnis