Lernziele

Anwendungssysteme sind maschinelle Aufgabenträger, die automatisierte Teilaufgaben eines Informationssystems übernehmen. In dieser Kurseinheit beschreiben wir betriebswirtschaftliche Standardsoftwaresysteme als spezielle Anwendungssysteme. Die Auswahl, die Einführung und die Nutzung betriebswirtschaftlicher Standardsoftware werden behandelt. Außerdem gehen wir auf betriebswirtschaftliche Individualsoftware ein. Standard- und Individualsoftware werden gegenübergestellt. Anschließend werden objektorientierte bzw. komponentenbasierte Rahmenwerke als Kompromiss zwischen Individual- und Standardsoftware behandelt.

Nach dem Studium von Abschnitt 5.1 sollen Sie in der Lage sein, wichtige Begriffe im Umfeld von betrieblichen Anwendungssystemen erläutern zu können.

Nachdem Sie Abschnitt 5.2 durchgearbeitet haben, können Sie den Begriff der betrieblichen Individualsoftware erläutern. Außerdem kennen Sie die wesentlichen Kategorien von Softwarekomponenten.

Die Beschäftigung mit den Inhalten von Abschnitt 5.3 soll Ihnen zu Kenntnissen in betriebswirtschaftlicher Standardsoftware verhelfen. Sie können die wesentlichen Merkmale betriebswirtschaftlicher Standardsoftware beschreiben. Sie können Branchenlösungen charakterisieren. Außerdem können Sie Vor- und Nachteile betriebswirtschaftlicher Standardsoftwarelösungen diskutieren. Sie sollen in der Lage sein, sowohl Architektur als auch wesentliche betriebswirtschaftliche Komponenten von ERP-Systemen zu beschreiben. Die unterschiedlichen Phasen der Auswahl einer konkreten betriebswirtschaftlichen Standardsoftware können Sie charakterisieren. Die Tätigkeiten in den einzelnen Phasen sind Ihnen bekannt. Nachdem Sie den Abschnitt studiert haben, sollen Sie weiterhin in der Lage sein, die Phasen der Einführung betriebswirtschaftlicher Standardsoftware beschreiben zu können. Sie kennen wesentliche Arbeitsschritte, die im Rahmen der ständigen Verbesserung der betriebswirtschaftlichen Standardsoftware durchzuführen sind. Außerdem sollen Ihnen wesentliche Schnittstellentechnologien des ERP-Systems mySAP ERP vertraut sein.

Das Studium von Abschnitt 5.4 soll Sie in die Lage versetzen, objektorientierte und agentenbasierte Rahmenwerke als Kompromiss zwischen betriebswirtschaftlicher Individual- und Standardsoftware zu verstehen. Dazu lernen Sie Eigenschaften von Rahmenwerken sowie verschiedene Arten von Rahmenwerken kennen. Sie sollen wesentliche Konstruktionsprinzipien für verschiedene Arten von Rahmenwerken verstanden haben. Sie sind in der Lage, die Umsetzung allgemeiner Entwurfsprinzipien in speziellen Rahmenwerken für die Produktionsdomäne zu verstehen.

Die Übungsaufgaben innerhalb der Kurseinheit dienen der Überprüfung des erreichten Kenntnisstandes. Eine selbständige Bearbeitung der Aufgaben ist für das Erreichen eines tieferen Verständnisses des Stoffes unerlässlich. Einige der in dieser Kurseinheit behandelten Konzepte werden für Sie leichter verständlich

sein, wenn Sie die praktischen Übungen am mySAP ERP-System im Rahmen der Einsendeaufgaben durchführen. Ein Kurs über betriebliche Informationssysteme ohne praktische Übungen an einem ERP-System ist mit einem Tanzkurs ohne Tanzübungen vergleichbar. Im diesem Sinne möchten wir Sie dazu auffordern, sich an der Bearbeitung der Einsendeaufgaben zu beteiligen.

5.1 Begriffsbestimmungen

Wie in Kurseinheit 1 gezeigt, zielen Anwendungssysteme auf die automatisierte Erfüllung von Teilaufgaben eines Informationssystems ab. Somit sind Anwendungssysteme maschinelle Aufgabenträger [43].

Nutzermaschine

Wir stellen zunächst wichtige Struktur- und Verhaltensmerkmale von Anwendungssystemen zusammen. Wie in Kurseinheit 2 beschrieben, kann man zwischen **Außen- und Innensicht** eines Informationssystems unterscheiden. Ein Anwendungssystem stellt in der Außensicht eine **Nutzermaschine** bereit. Die Nutzermaschine verfügt über eine Schnittstelle, die diejenigen Datenobjekte und Operatoren bereitstellt, die isomorph zu den Aufgabenobjekten und Verrichtungen der durch das Anwendungssystem automatisierten Aufgaben sind.

Die Nutzermaschine eines Anwendungssystems lebt in einer Verfahrensumgebung, die von den Nutzermaschinen benachbarter Anwendungssysteme für die Verrichtung automatisierter Aufgaben sowie von den personellen Aufgabenträgern nicht-automatisierter Aufgaben gebildet wird.

Basismaschine

Die Innensicht eines Anwendungssystems wird durch eine **Basismaschine** und ein **Programmiersystem** gebildet [43]. Durch die Basismaschine werden Datenobjekte und Operatoren bereitgestellt, die zur Realisierung der Datenobjekte und Operatoren der Nutzermaschine dienen. Datenbankmanagementsysteme und Betriebssysteme sowie die Hardware der Computersysteme sind Bestandteil einer Basismaschine. Außerdem gehören Datenobjekte und Operatoren von Programmiersprachen ebenfalls zur Basismaschine. Das Programmiersystem ist für die Abbildung der Datenobjekte und Operatoren der Nutzermaschine auf die Datenobjekte und Operatoren der Basismaschine verantwortlich. Häufig werden mehrere Zwischenmaschinen eingesetzt, die unter Verwendung wohldefinierter Schnittstellen die gewünschte Transformation vornehmen.

Die Basismaschine ist ähnlich wie die Nutzermaschine in eine Systemumgebung eingebettet. Diese Systemumgebung besteht aus Basismaschinen benachbarter Anwendungssysteme. Zwischen den unterschiedlichen Basismaschinen sind Kommunikationsschnittstellen implementiert.

Die beschriebene Situation ist in Abbildung 5.1 veranschaulicht. Die Abbildung zeigt die Interaktion zwischen zwei Nutzer- und Basismaschinen.

Die Realisierung der Innensicht eines Anwendungssystems wird typischerweise objektorientiert vorgenommen. Der objektorientierte Ansatz ermöglicht wechselseitig vernetzte Client- und Serverobjekte und somit die Möglichkeit der fast beliebigen Verteilung des Anwendungssystems auf Rechnersysteme.

Wie in Kurseinheit 4 bei der Behandlung von Middlewareansätzen dargestellt, agieren Objekte über Rechnergrenzen hinweg und können andere Objekte veranlassen, ihre Fähigkeiten (Methoden) auszuführen. Auf welchem Rechner im Netzwerk sich das Objekt dabei befindet, ist relativ unbedeutend.

Anwendungssysteme können unter Verwendung von Individual- oder Standardsoftware realisiert werden. Wir gehen in den nachfolgenden Abschnitten auf

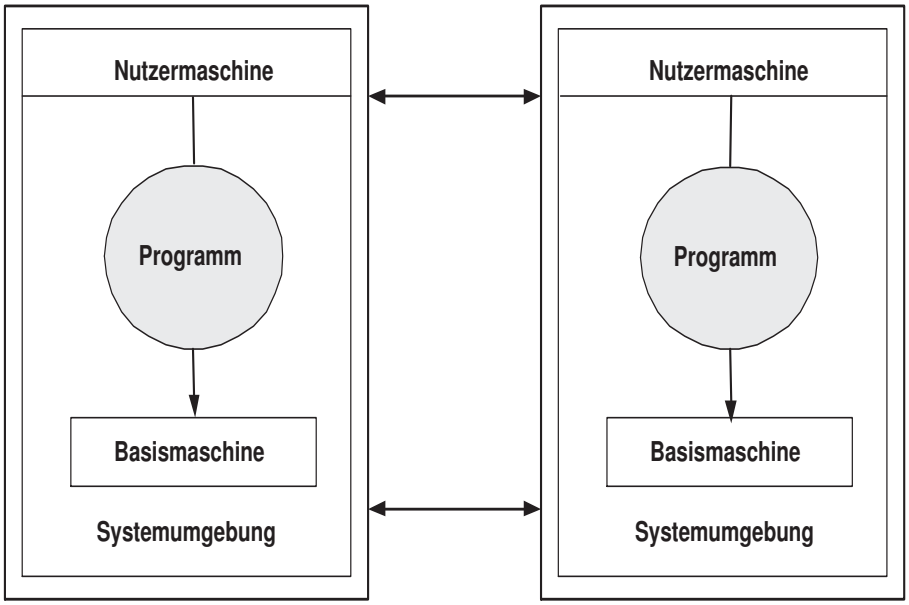

Abbildung 5.1: Zusammenhang Nutzer- und Basismaschine

diese beiden prinzipiellen Möglichkeiten im Detail ein. Dabei interessieren uns im Rahmen dieses Kurses insbesondere Anwendungssysteme, die sich im betrieblichen Einsatz befinden und zur Lösung betriebswirtschaftlicher Probleme dienen. Wir sprechen von betriebswirtschaftlichen Anwendungssystemen.

5.2 Betriebswirtschaftliche Individualsoftware

Individualsoftware

Wir definieren zunächst den Begriff der Individualsoftware.

Definition 5.2.1 (Individualsoftware) *Individualsoftware ist eine Zusammenstellung genau bestimmter Ausprägungen von Verfahrensabläufen als Programmbausteine, deren Funktionalität für wenige, konkrete, vorher festgelegte Anwendungsfälle benötigt wird.*

Betriebswirtschaftliche Individualsoftware wird somit für eine spezielle betriebliche Anforderung individuell entworfen und implementiert. Sie wird entweder direkt im Unternehmen selber entwickelt oder fremdbezogen [25].

Softwaresysteme in Unternehmen beschäftigen sich mit fachlichen Anwendungen, da Softwareentwicklungsprojekte in einem Unternehmen in der überwiegenden Zahl der Fälle durch ein konkretes betriebliches Problem ausgelöst werden. Sie stellen technische Schnittstellen zu Betriebssystemen, Datenbanken und Middleware zur Verfügung.

Jede Softwarekomponente gehört aus diesem Grund genau einer der folgenden vier Kategorien an [2]:

Softwarekategorien

1. Die Softwarekomponente ist unabhängig von einer spezifischen Anwendung und einem bestimmten technischen Kontext.
2. Die Softwarekomponente wird durch die Anwendung determiniert, sie ist unabhängig von der Technik.
3. Im Gegensatz zur vorherigen Kategorie ist bei der dritten Kategorie die Softwarekomponente nicht durch ein Anwendungsproblem beeinflusst, sondern dient der Lösung eines technischen Problems.
4. Die letzte Kategorie umfasst Softwarekomponenten, die sowohl Anwendungsprobleme als auch technische Probleme lösen.

Wir betrachten das nachfolgende Beispiel für die unterschiedlichen Kategorien von Softwarekomponenten.

Beispiel 5.2.1 (Kategorien für Softwarekomponenten) *Quellcode mit einem anwendungsbezogenen Hintergrund verwendet Begriffe wie „Customer Order“, „Lot“, „Schedule“ oder „Product“, während Quellcode mit technischem Bezug beispielsweise die Application-Programming-Interfaces (API) der ODBC- oder der Microsoft-Foundation-Classes (MFC)-Klassenbibliotheken kennt.*

In Anlehnung an [2] wird anwendungsbestimmte Software mit „A“, Software mit technischem Fokus mit „T“ sowie Software, die weder Anwendungs- noch Technik-Bezug hat, mit „0“ gekennzeichnet. Die vier möglichen Kategorien sind somit „0“, „A“, „T“ sowie „AT“.

Wir diskutieren an dieser Stelle diese vier Kategorien unter dem Gesichtspunkt der Softwarewiederverwendung.

- **0-Software** wird typischerweise durch „neutrale“ Klassenbibliotheken gebildet, die abstrakte Konzepte wie Strings und Container zur Verfügung stellen. Sie ist daher ideal wiederverwendbar, allerdings für sich alleine nutzlos.
- **A-Software** kann wiederverwendet werden, wenn die durch die A-Software implementierte Anwendungslogik vollständig oder partiell benötigt wird. Wir betrachten das nachfolgende Beispiel für A-Software.

  Beispiel 5.2.2 (A-Software) *Kunden- und Adressverwaltung sind häufig genutzte fachliche Funktionalitäten in vielen betrieblichen Anwendungssystemen. Sowohl Kunden- als auch Adressverwaltung setzen ihrerseits auf einer Datumsverwaltung auf.*

- **T-Software** kann stets dann wiederverwendet werden, wenn ein neu zu bauendes Anwendungssystem die durch die T-Software implementierten technischen Konzepte einsetzen will. Wir zeigen die nachfolgenden Beispiele für T-Software.

Beispiel 5.2.3 (T-Software) *Verteilte Anwendungen, die auf der .NET-Klassenbibliothek basieren, verwenden das in Abschnitt 4.2.5 beschriebene .NET-Remoting-Rahmenwerk. Alle betrieblichen Informationssysteme verwenden in irgendeiner Form eine Datenverwaltung, d.h., die Datenzugriffskomponente kann als T-Software realisiert werden.*

- **AT-Software** fokusiert gleichzeitig auf die Bereiche Anwendung und Technik. Sie ist schwierig zu warten, weiterzuentwickeln und besitzt einen geringen Wiederverwendungsgrad.

Individualsoftware wird häufig als AT-Software realisiert, da die Trennung nach Zuständigkeiten, wie sie durch 0-, A- und T-Software gegeben ist, aufgrund des Fehlens von ausreichender Zeit bzw. eines zu geringen Budgets für die Entwicklung der Software nicht erreicht werden kann.

Die in Abschnitt 5.3 zu behandelnde betriebswirtschaftliche Standardsoftware ermöglicht häufig eine bessere Trennung der Zuständigkeiten in „A“- und „T“-Software, da mehr Sorgfalt beim Entwurf der Anwendungsfunktionalität aufgewendet werden muss, wenn diese in vielen Situationen einsetzbar sein soll. Das gilt auch für objektorientierte bzw. komponentenbasierte Rahmenwerke, die in Abschnitt 5.4 dargestellt werden.

5.3 Betriebswirtschaftliche Standardsoftwaresysteme

In diesem Abschnitt führen wir betriebswirtschaftliche Standardsoftwaresysteme ein. Wir diskutieren Vor- und Nachteile derartiger Softwaresysteme. ERP-Systeme werden beschrieben. Anschließend gehen wir auf Auswahl, Einführung sowie Betrieb und Wartung betriebswirtschaftlicher Standardsoftwaresysteme ein.

5.3.1 Begriffsbestimmungen

Standard-software

Wir definieren nachfolgend den Begriff der Standardsoftware.

Definition 5.3.1 (Standardsoftware) *Unter Standardsoftware wird Software verstanden, die ohne wesentliche Änderungen in unterschiedlichen Unternehmen einsetzbar ist.*

Innerhalb dieses Kurses wird ein besonderer Schwerpunkt auf Standardsoftwaresysteme gelegt, die sich im betrieblichen Einsatz befinden. Vor allen Dingen Planungs-, Dispositions-, Administrations- und Kontrollsysteme sind von Interesse. Diese Art von Standardsoftware wird als betriebswirtschaftliche Standardsoftware bezeichnet.

Betriebswirtschaftliche Standardsoftware kann entsprechend des vorliegenden horizontalen Integrationsgrades in Software für einzelne Funktionsbereiche, in ERP- und in E-Business-Komplettpakete unterteilt werden. Wir führen den Begriff des E-Business-Systems wie folgt ein [14, 16, 42].

E-Business-Systeme

Definition 5.3.2 *E-Business-Systeme sind integrierte Anwendungssysteme, die betriebliche Prozesse und die unternehmensübergreifenden Koordinations- und Kooperationsbeziehungen unterstützen.*

Spezielle E-Business-Systeme sind u.a. Gegenstand des Wahlpflichtmoduls „Entscheidungsunterstützungsmethoden in unternehmensweiten Softwaresystemen".

Betriebsgrößenorientierung

Betriebswirtschaftliche Standardsoftware existiert für Großbetriebe, mittelgroße Betriebe sowie für Kleinbetriebe. Man spricht hier von Betriebsgrößenorientierung. Standardsoftware für Großbetriebe versucht eine vollständige Abdeckung aller betrieblichen Funktionen. Für Kleinbetriebe bzw. mittelgroße Betriebe steht hingegen die übersichtliche Gestaltung sowie die kostengünstige Unterstützung der wesentlichen Teilbereiche im Mittelpunkt des Interesses [14, 13]. Daneben spielen eine schnelle Implementierbarkeit sowie eine einfache Bedienbarkeit der Standardsoftware eine große Rolle.

Branchenorientierung

Als letztes Differenzierungsmerkmal für betriebswirtschaftliche Standardsoftware wird die Branchenorientierung herangezogen. Wir unterscheiden zwischen branchenneutraler bzw. branchenspezifischer betriebswirtschaftlicher Standardsoftware.

In Abschnitt 1.3 wurde der Begriff „Branche" bei der Behandlung von Brancheninformationssystemen im Wesentlichen mit Wirtschaftszweig gleichgesetzt. Es erscheint jedoch sinnvoll, einen Branchenbegriff zu betrachten, der sich stärker als der Wirtschaftszweig an Produktgruppen orientiert [27]. Wir betrachten dazu in Anlehnung an [27] das folgende Beispiel.

Beispiel 5.3.1 (Branche) *Unternehmen der Textilindustrie, des Textilhandwerks und des Textilhandels werden zur Textilbranche zusammengefasst.*

Anforderungen an Branchenlösungen werden im Rahmen dieser Kurseinheit genauer in Abschnitt 5.3.5 untersucht.

5.3.2 Merkmale betriebswirtschaftlicher Standardsoftwaresysteme

Merkmale von Standardsoftware

In diesem Abschnitt sollen Merkmale betriebswirtschaftlicher Standardsoftware anhand der nachfolgenden Kriterien herausgearbeitet werden:

- Branchenneutralität,
- Internationalisierung,

- Anpassungsfähigkeit der Software an spezifische betriebliche Gegebenheiten,
- Funktionsumfang,
- Preis,
- Anzahl der Installationen.

Betriebswirtschaftliche Standardsoftwaresysteme sind häufig nicht auf genau eine Branche zugeschnitten, denn dies würde dem Grundgedanken von Standardsoftware widersprechen und die Einsatzmöglichkeiten der Standardsoftware verkleinern. Wir verweisen an dieser Stelle auf die Diskussion in Abschnitt 5.3.5.

Länderspezifika

Betriebswirtschaftliche Standardsoftware kann typischerweise in unterschiedlichen Ländern verwendet werden. Das wird unter anderem dadurch erreicht, dass mehrere Sprachvarianten für eine konkrete betriebswirtschaftliche Standardsoftware vorliegen und dass länderspezifische Besonderheiten und Anforderungen unterschiedlich abgebildet werden. Wir betrachten dazu das nachfolgende Beispiel.

Beispiel 5.3.2 (Abbildung von Länderspezifika) *Es ist notwendig, länderspezifische Verfahren zur Lohn- und Gehaltsabrechnung, unterschiedliche kaufmännische Bewertungsmethoden, landestypische Bilanzierungsverfahren, Datumsformate und Währungen in betriebswirtschaftlicher Standardsoftware abzubilden.*

Außerdem ist es erforderlich, dass Texte, u.a. zur Beschreibung von Materialien, in den betriebswirtschaftlichen Standardsoftwaresystemen in mehreren Sprachen und auch Zeichensätzen hinterlegt sind.

konfigurierbare betriebswirtschaftliche Standardsoftware

Wir beschäftigen uns an dieser Stelle zunächst mit **konfigurierbarer betriebswirtschaftlicher Standardsoftware**. Die zweite Klasse betriebswirtschaftlicher Standardsoftwaresysteme, sogenannte **komponentenbasierte betriebswirtschaftliche Software**, wird an späterer Stelle in Abschnitt 5.4 genauer dargestellt.

Konfigurierbare betriebswirtschaftliche Standardsoftware ist dadurch charakterisiert, dass eine Anpassung der betriebswirtschaftlichen Standardsoftware an konkrete betriebliche Gegebenheiten durch das Setzen von Parametern erreicht werden kann. Wir definieren den Begriff der Parametrisierung wie folgt.

Definition 5.3.3 (Parametrisierung) *Wir verstehen unter Parametrisierung (Customizing) eine Anpassung des Aussehens und der Funktionalität der Standardsoftware, ohne dass der Quellcode der Software geändert werden muss.*

Release

Wir definieren in diesem Zusammenhang den Begriff der Releasefähigkeit.

Definition 5.3.4 (Releasefähige Software) *Eine Software wird als releasefähig bezeichnet, wenn bei neuen Versionen der Software (Releases) keine Anpassung am Quellcode der in den Unternehmen installierten Software vorgenommen werden muss.*

Customizing

Im Idealfall bleiben die durch das Customizing vorgenommenen unternehmensspezifischen Anpassungen und Einstellungen erhalten. Neue Releasestände werden typischerweise jährlich im Rahmen von Wartungsverträgen ausgeliefert. Die Installation des neuen Releases wird entweder durch externe Dienstleister oder direkt durch Systembetreuer im Unternehmen vorgenommen.

Vor der Nutzung des Anwendungssystems müssen zur Definitionszeit Parameter gesetzt werden, die zur Ausführungszeit (Laufzeit) das Verhalten der Software bestimmen. Wir unterscheiden zwischen strukturellen und verfahrensrelevanten Parametern.

Strukturelle Parameter

Strukturelle Parameter dienen der Abbildung der Aufbauorganisation sowie der Produkte und Ressourcen eines Unternehmens. Wir betrachten dazu das nachfolgende Beispiel.

Beispiel 5.3.3 (Strukturelle Parameter) *Die Werke eines Unternehmens sind als Parameter in die betriebswirtschaftliche Software einzupflegen.*

Verfahrensrelevante Parameter

Verfahrensrelevante Parameter bestimmen die zu implementierende Funktionalität in Abhängigkeit von den eingestellten Werten. Das folgende Beispiel zeigt spezielle verfahrensrelevante Parameter.

Beispiel 5.3.4 (Verfahrensrelevante Parameter) *Die Auswahl eines einzusetzenden Prognoseverfahrens ist ein verfahrensrelevanter Parameter. Innerhalb eines bestimmten Prognoseverfahrens müssen zahlreiche Parameter gewählt werden, welche die Güte des Prognoseverfahrens entscheidend bestimmen. Diese Parameter sind ebenfalls als verfahrensrelevante Parameter der betriebswirtschaftlichen Standardsoftware aufzufassen.*

Eine Parametrisierung kann als die Konfiguration eines Und-Oder-Baumes aufgefasst werden [1]. Ein Und-Oder-Baum enthält attributierte Und-Knoten zur Abbildung von Struktur und Verfahren sowie Oder-Knoten, die verschiedene Verfahrensausprägungen angeben und damit einen Und-Knoten zur Abbildung von Verfahren als Vaterknoten besitzen. Wir betrachten dazu das folgende Beispiel.

Beispiel 5.3.5 (Und-Oder-Baum) *In Abbildung 5.2 zeigen wir einen Und-Oder-Baum, der für Lose eine Vorwärts- oder Rückwärtsterminierung vorsieht.*

Modifikation/ Erweiterung

Falls die durch die Konfigurierbarkeit gegebene Anpassungsfähigkeit an spezielle betriebliche Situationen nicht ausreichend ist, ist es notwendig, Modifikationen bzw. sogar Erweiterungen der betriebswirtschaftlichen Standardsoftware vorzunehmen. Während das Customizing häufig keine Programmiertätigkeiten erfordert, muss bei Modifikationen und Erweiterungen programmiert werden.

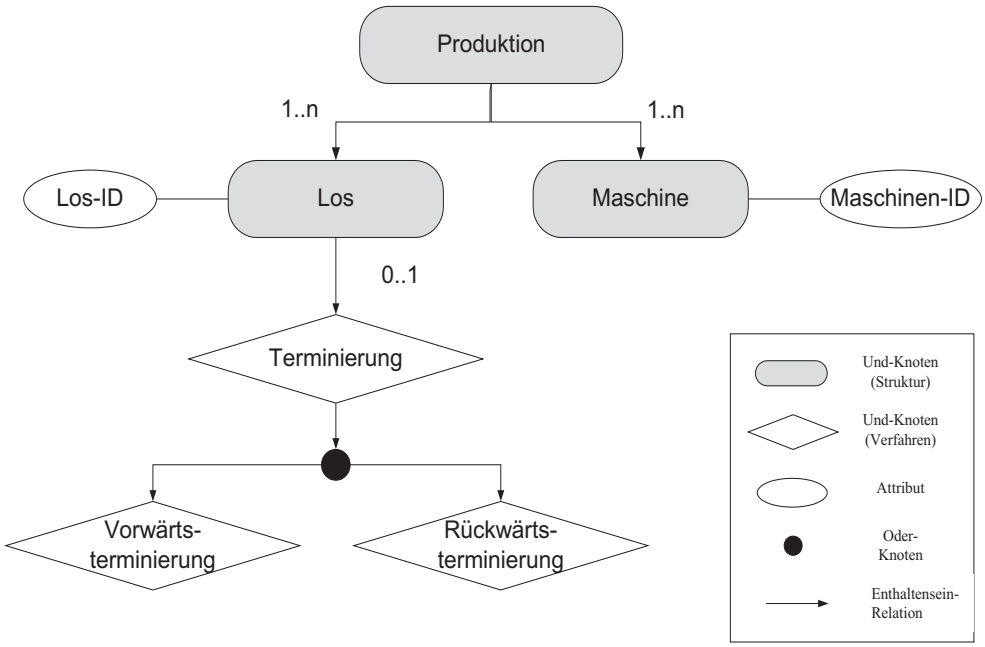

Abbildung 5.2: Und-Oder-Baum für die Parametrisierung

Adaption

Customizing, Modifikationen und Erweiterungen sind spezielle Ausprägungen der **Adaption**, die nachfolgend beschrieben wird. Adaption umfasst neben der bereits dargestellten Parametrisierung die nachfolgenden beiden Kategorien:

- **Modifikation:** Standardsoftwareprodukte sind mit Entwicklungsumgebungen ausgestattet, die eine Änderung von Programmen erlauben. Probleme gibt es dabei oft mit der Wahrung der Releasefähigkeit der Software.

- **Erweiterung:** Die Software wird um eigenentwickelte Komponenten erweitert (Individualentwicklung). Dabei können vollständig neue Komponenten entstehen. Weiterhin können neue Komponenten durch Kopieren bestehender Komponenten abgeleitet werden. Probleme entstehen mit der Releasefähigkeit der Software, insbesondere treten auch Schnittstellenprobleme auf.

Im nachfolgenden Beispiel betrachten wir die Parametrisierung innerhalb der betriebswirtschaftlichen Standardsoftware SAP R/3.

Beispiel 5.3.6 (Parametrisierung von SAP R/3) *In SAP R/3 wird ein interpretativer Ansatz zur Berücksichtigung von Parametern gewählt. Parameter, in SAP R/3 auch als Customizing-Daten bezeichnet, werden über eine Customizing-Transaktion in die zum R/3-System zugehörige Datenbank eingestellt. Anwendungskomponenten, die von den eingestellten Parametern abhängig*

sind, berücksichtigen, d.h. interpretieren, die neuen Einstellungen sofort. Zeitintervalle können definiert werden, welche die Gültigkeit von bestimmten Parametereinstellungen begrenzen.

Anschließend zeigen wir noch im nachfolgenden Beispiel, wie eine Erweiterung von SAP R/3 erfolgen kann.

Beispiel 5.3.7 (Erweiterung von SAP R/3) *Eine Erweiterung von R/3 ist durch die Verwendung von User-Exits als spezielle vordefinierte Schnittstellen möglich, die zur Erweiterung der entsprechenden Komponenten genutzt werden können. Dadurch ist es möglich, die Entwicklung individueller Programmerweiterungen vorzunehmen und diese dann in den bestehenden Kontrollfluss der R/3-Anwendungen aufzunehmen.*

Die bisherigen Ausführungen zur Adaption haben sich ausschließlich auf konfigurierbare betriebswirtschaftliche Standardsoftwaresysteme bezogen, die häufig in Form von monolithischen bzw. Systemen mit klassischer Client-Server Architektur realisiert werden. Wie wir aber bereits in Abschnitt 2.3.4 und Abschnitt 2.6.3 von Kurseinheit 2 gesehen haben, gibt es auch die Möglichkeit, bestimmte benötigte Anwendungsfunktionalität durch Kombination von Komponenten oder noch allgemeiner von Diensten zu erreichen.

Funktionalität

Die durch eine betriebswirtschaftliche Standardsoftware bereitgestellte Funktionalität ist auf ein bestimmtes betriebswirtschaftliches Einsatzgebiet abgestimmt. Wir betrachten dazu das folgende Beispiel zur Veranschaulichung.

Beispiel 5.3.8 (Funktionalität) *Betriebswirtschaftliche Standardsoftware für die Produktionsplanung und -steuerung ist in Form von PPS-Systemen gegeben. Software für das Kundenbeziehungsmanagement wird durch CRM-Systeme bereitgestellt.*

Für das jeweilige Einsatzgebiet wird eine umfassende Menge von Funktionen bereitgestellt. Beim Auftreten von unvorhergesehenen Anforderungen und Aufgaben ist eine individuelle Anpassung der Software an die konkrete Anforderung notwendig. Der im Standard vorhandene Funktionsumfang übersteigt teilweise den Funktionsumfang von Individualsoftware für analoge Problemstellungen beträchtlich.

Lizenzgebühren

Die anfallenden Kosten für die Nutzung einer betriebswirtschaftlichen Standardsoftware sind gut zu schätzen. Diese Kosten, als Lizenzgebühren bezeichnet, werden häufig in Abhängigkeit von der Anzahl der Anwender festgelegt. Dabei ist zu berücksichtigen, dass nicht unerhebliche Nebenkosten für die Einführung, die Wartung, die für den Betrieb benötigte Hardware sowie für die Nutzung von Datenbankmanagementsystemen auftreten.

Man spricht von betriebswirtschaftlicher Standardsoftware, wenn die Software in mindestens einem zweistelligen Kreis von Unternehmen eingesetzt wird [13]. Häufig werden auch Referenzkunden erwartet. Alternativ kann eine garantierte und nachgewiesene Releasefähigkeit eine Mindestanzahl von Installationen ersetzen.

5.3.3 Vorteile beim Einsatz von betriebswirtschaftlicher Standardsoftware

Vorteile von betriebswirtschaftlicher Standardsoftware

Vorteile existieren bei der Verwendung von betriebswirtschaftlicher Standardsoftware unter den nachfolgenden Gesichtspunkten [13]:

1. Verteilung der Entwicklungskosten,
2. Einsatzbedingungen,
3. Qualität,
4. Zeitbedarf für die Einführung,
5. Einkauf externen Wissens,
6. geringerer Personalbedarf,
7. Verfügbarkeit von Zusatzleistungen,
8. Integrationsfähigkeit.

Entwicklungskosten

Wir beschreiben der Reihe nach die unterschiedlichen Vorteile von betriebswirtschaftlicher Standardsoftware. Die Entwicklungskosten, auf die Anzahl der Installationen bezogen, sind niedriger als bei Individualsoftware. Durch diese Verteilung auf eine Vielzahl von Anwendern sind auch hohe Entwicklungskosten möglich.

Der Ausgangspunkt für eine Einführung einer betriebswirtschaftlichen Standardsoftware ist ein betriebliches Problem, d.h. ein konkreter Sachzwang. Häufig veranlasst erst dieser Sachzwang die notwendige Investition. Die Zeitspanne, die zwischen Planung und Implementierung der Software vergeht, ist somit ein relevanter Faktor. Eine geringe Zeitspanne beschleunigt die Lösung des betrieblichen Problems. Die Kosten, das Risiko und der frühestmögliche Implementierungszeitpunkt sind bei Standardsoftware relativ genau festlegbar.

Softwarequalität

Die Qualität von Software wird bestimmt durch

- **Benutzerfreundlichkeit**: Standardsoftware ist oft komfortabler als Individualsoftware, da bei der Erstellung von Individualsoftware Probleme mit dem geplanten Fertigstellungszeitpunkt sowie dem zur Verfügung stehenden Budget die Regel sind und deshalb häufig bei der Gestaltung der graphischen Benutzeroberfläche und der Erstellung der Dokumentation gespart wird.

- **Störanfälligkeit**: Standardsoftware ist weniger störanfällig als Individualsoftware, da der praktische Einsatz und somit ein Test in einer Vielzahl von Unternehmen stattfindet.

- **Funktionalität**: Standardsoftware bietet in einem größeren Umfang als Individualsoftware Funktionalität an. Ein kritischer Vergleich der angebotenen mit der tatsächlich benötigten Funktionalität ist deshalb notwendig.
- **Flexibilität**: Flexibilität besitzt in diesem Zusammenhang die Ausprägungen Portabilität und Adaptivität. Für die Portabilität gilt, dass Standardsoftware in der Regel nicht an eine bestimmte Hardware bzw. ein bestimmtes Betriebssystem gebunden ist. Bei Individualsoftware ist das hingegen häufig der Fall. Adaptivität ist bei Individualsoftware oft mehr ausgeprägt als bei Standardsoftware. Bei Standardsoftware besteht oft ein Kopplungsproblem mit anderen Anwendungssystemen der vorhandenen Systemlandschaft im konkreten Unternehmen.

Zeitbedarf für Einführung

Standardsoftware kann in wesentlich kürzerer Zeit eingeführt werden als Individualsoftware, aber der Zeitaufwand für die Anpassungsphase darf nicht unterschätzt werden. Aufwand entsteht dabei für die Einstellung unternehmensspezifischer Parameter und individueller Berichtsformate sowie bei der Übernahme der Daten aus Altsystemen in die neu eingeführte betriebswirtschaftliche Standardsoftware.

Wissenstransfer

Betriebswirtschaftliche Standardsoftware ist das Resultat von Verbesserungsanstrengungen. Beim Kauf von Standardsoftware erwirbt das Unternehmen die Ergebnisse der Erfahrungen anderer Unternehmen, d.h., es findet ein Einkauf externen Wissens statt [13]. Das nachfolgende Beispiel veranschaulicht einen möglichen Wissenstransfer.

Beispiel 5.3.9 (Wissenstransfer) *Als positiver Nebeneffekt ist festzustellen, dass sich die verwendete Terminologie im Unternehmen vereinheitlicht, da eine Anpassung an die Begriffsbildungen des betriebswirtschaftlichen Standardsoftwaresystems auftritt. Das erleichtert die Kommunikation zwischen den unterschiedlichen Fachabteilungen und der IT-Abteilung.*

Personalbedarf

Beim Einsatz von Standardsoftware wird wesentlich weniger Personal als bei der Neuerstellung einer Individualsoftware benötigt. Das führt dazu, dass häufig keine eigenen Anwendungsentwicklungsabteilungen mehr notwendig sind.

Zusatzleistungen

Die nachfolgenden Zusatzleistungen sind mit der Anwendung einer betriebswirtschaftlichen Standardsoftware verbunden:

- **Planungshilfen bei der Systemauswahl**: Hierzu zählen Vorführungen bei Referenzkunden, Testinstallationen oder Aufdecken von Schwächen im Unternehmen durch den Einsatz externer Berater.
- **Installationshilfen**: Externe Berater unterstützen das Unternehmen insbesondere bei der Installation der Systeme. Dabei wird eine Anpassung an individuelle Bedürfnisse und insbesondere auch an die vorhandene Hard- und Software des Unternehmens in dessen Systemlandschaft vorgenommen. Weiterhin bieten die Hersteller von Standardsoftware auch Unterstützung bei der Schulung der Mitarbeiter des Unternehmens an. Die

Schulungsanstrengungen werden durch vorhandene Lernprogramme unterstützt. Das nachfolgende Beispiel illustriert diesen Sachverhalt.

Beispiel 5.3.10 (Lernprogramm) *Im SAP R/3-Umfeld enthält das International-Development-and-Education-System (IDES) eine Vielzahl von Fallstudien, die den Benutzer schrittweise mit der Nutzung des SAP R/3-Systems vertraut machen.*

- **Unterstützung bei Wartung sowie Erweiterung und Aktualisierung der Systeme**: Wartungsserviceleistungen sind preiswert, da in regelmäßigen Abständen Releases angeboten werden. Gleichzeitig existieren aber Probleme mit der Release- und Wartungsfreundlichkeit durch unternehmensspezifische Modifikationen und Erweiterungen der Software. Änderungen an der Software können nur durch dafür vorgesehene Schnittstellen (User-Exits) oder durch Parametrisierung vorgenommen werden. Quellcodemodifikationen sind häufig nicht zulässig. Wir betrachten das folgende Beispiel für die Unterstützung von Wartungsaktivitäten.

  Beispiel 5.3.11 (Wartungsunterstützung) *Im SAP-Umfeld existiert die Möglichkeit zum Absenden von Problembeschreibungen und Fehlermeldungen an die SAP AG im Rahmen des Online-Service-Systems (OSS). Das Herunterladen von Hot-Packages, einer Sammlung von Fehlerkorrekturen zum SAP R/3-System, ist möglich. Remote-Consulting als kostenpflichtige Beratungsleistung kann angefordert werden.*

Betriebswirtschaftliche Standardsoftwaresysteme sind integrierte Lösungen, welche die Daten-, Prozess- und Funktionsintegration unterstützen. Die in Kurseinheit 1 behandelte integrierte Informationsverarbeitung wird wesentlich durch betriebswirtschaftliche Standardsoftwaresysteme unterstützt.

5.3.4 Nachteile und Risiken beim Einsatz von betriebswirtschaftlicher Standardsoftware

Nachteile

Nachteile existieren unter den nachfolgenden Gesichtspunkten [13]:

- Hardwarebedarf inklusive Betriebssystem und Datenbankmanagementsystem,
- Herstellerabhängigkeit,
- Anpassungsaufwand,
- Schnittstellenprobleme,
- Verlust von Wettbewerbsvorteilen,

- psychologische Aspekte.

Hardwarebedarf

Wir beschreiben der Reihe nach die unterschiedlichen Gesichtspunkte. Der erhöhte Hardwarebedarf bei Standardsoftware lässt sich dadurch erklären, dass in betriebswirtschaftlichen Standardsoftwaresystemen im Allgemeinen mehr Funktionen als benötigt realisiert werden. Ein auf diese Funktionalität ausgelegtes umfangreiches Datenmodell ist für betriebswirtschaftliche Standardsoftware typisch. Das führt zu einer ineffiziente Nutzung der Hardwareressourcen. Als Ergebnis erhalten wir lange Antwortzeiten und großen Speicherbedarf für betriebswirtschaftliche Standardsoftware. Ziel der Neubeschaffung von Software ist die Ablösung von Altsystemen. Die Altsysteme laufen aber typischerweise auf alter Hardware. Moderne betriebswirtschaftliche Standardsoftwaresysteme stellen hohe Ansprüche an die Hardware des Servers und der Client-Rechner. Aus diesem Grund ist bei Neuerwerb von betriebswirtschaftlicher Standardsoftware oft auch neue Hardware notwendig. Erschwerend kommt hinzu, dass die Entwicklung der Standardsoftware auf modernen und damit schnellen Maschinen stattfindet. Eine entsprechende Hardware wird daher auch für den Einsatz der betriebswirtschaftlichen Standardsoftware benötigt.

Herstellerabhängigkeit

Während der wirtschaftlichen Nutzungsdauer der betriebswirtschaftlichen Standardsoftware, häufig mehr als zehn Jahre, entsteht durch eine Bindung an den entsprechenden Hersteller eine starke Abhängigkeit. Aus diesem Grund ist große Sorgfalt bei der Lieferantenauswahl und der Anforderungsspezifikation bei Kaufentscheidungen notwendig. Falls die von der Software geforderten Standards für die Anbindung weiterer Anwendungssysteme proprietär sind, d.h. ausschließlich Software eines Anbieters verwendet werden kann, entsteht eine starke Abhängigkeit. Es kann zu großen Schwierigkeiten kommen, da eine Quasi-Monopolstellung des Anbieters vorliegt.

Anpassungsaufwand

Anpassungsaufwand entsteht durch den notwendigen Abgleich zwischen den Möglichkeiten der betriebswirtschaftlichen Standardsoftware und den Erfordernissen der betrieblichen Organisation andererseits. Anpassungen zur Abbildung der im Unternehmen zu verwendenden Schriftstücke, Formulare, Berichte und technologischen Dokumente sind notwendig. Teilweise sind dafür bis zu 40 Prozent des notwendigen Aufwandes erforderlich [13]. Außerdem sind Anpassungsaufwände für die im Unternehmen verwendeten Nummernsysteme vorzusehen.

Übungsaufgabe 5.1 (Nummernsysteme) *Welche Arten von Nummernsystemen in Unternehmen kennen Sie? Gehen Sie auf Unterscheidungsmerkmale ein. Nennen und erläutern Sie Beispiele für Nummernsysteme.*

Schnittstellen

Die Einführung von betriebswirtschaftlicher Standardsoftware bedeutet meistens die Ablösung eines oder mehrerer vorhandener Informationssysteme, die über Schnittstellen mit anderen Anwendungssystemen verbunden sind. Für die neu einzuführende Standardsoftware müssen dann ebenfalls Schnittstellen geschaffen werden. Obwohl Systeme wie SAP R/3 integrierte Lösungen darstellen, sind aber

trotzdem Schnittstellen zu anderen Softwaresystemen notwendig. Wir betrachten dazu das nachfolgende Beispiel.

Beispiel 5.3.12 (Schnittstellen) *APS-Systeme können als Erweiterung von ERP-Systemen aufgefasst werden, da ERP-Systeme typischerweise nicht in der Lage sind, die für eine Lieferkette charakteristischen Entscheidungsprobleme in ihrer Gesamtheit zu lösen (vergleiche hierzu die detaillierteren Ausführungen in Abschnitt 6.1.3.2 von Kurseinheit 6). Das APS-System Advanced-Planner-and-Optimizer (APO) greift auf die im SAP R/3-System vorgehaltenen Stamm- und Bewegungsdaten unter Verwendung der Lifecache-Technologie zu. Das bedeutet, dass Daten aus dem SAP R/3-System im Hauptspeicher des Rechners abgelegt werden, damit die APO-Software darauf zugreifen kann.*

Wettbewerbsvorteile

Der Verlust von Wettbewerbsvorteilen geht häufig mit der Einführung betriebswirtschaftlicher Standardsoftware einher. In der Literatur wird dazu eine kontroverse Diskussion geführt. Die Entscheidung, ob eine betriebswirtschaftliche Individual- oder Standardsoftware eingeführt wird, auch als „Make-or-Buy"-Entscheidung bezeichnet, ist eine individuelle Entscheidung, die von der vorliegenden Informationssystemarchitektur und der Wettbewerbssituation des Unternehmens abhängig ist. Wir betrachten dazu das nachfolgende Beispiel, in dem die Einführung von betriebswirtschaftschaftlicher Individualsoftware sinnvoll ist.

Beispiel 5.3.13 (Make-or-Buy I) *In der Handelsbranche sind lediglich geringe Gewinnspannen anzutreffen. Aus diesem Grund sind nur durch eine besonders effiziente Informationsverarbeitung Wettbewerbsvorteile möglich. Dadurch bedingt, wird meistens eine Entscheidung für Individualsoftware vorgenommen.*

Wir untersuchen ein weiteres Beispiel, bei dem eine Entscheidung für eine betriebswirtschaftliche Standardsoftware möglich ist.

Beispiel 5.3.14 (Make-or-Buy II) *Wir betrachten die Herstellung integrierter Schaltkreise. In dieser Domäne liegt der Schwerpunkt auf der Beherrschung des technologischen Prozesses. Eine Differenzierung der einzelnen Unternehmen wird dadurch gewährleistet. In der Folge sind einige wenige MES-Produkte (vergleiche die Ausführungen in Abschnitt 6.1.3.3) als Quasistandard in den Halbleiterfabriken anzutreffen.*

psychologische Aspekte

Psychologische Aspekte spielen eine Rolle, wenn eine eigene IT-Abteilung vorhanden ist und das Management sich trotzdem für eine betriebswirtschaftliche Standardsoftware entscheidet. Diese Entscheidung wird typischerweise als mangelndes Vertrauen in die IT-Abteilung interpretiert. Wenn dann Wartungs- und Anpassungsarbeiten an der Standardsoftware durch die IT-Abteilung vorgenommen werden sollen, kommt es zu innerbetriebliche Spannungen. In derartigen Situation ist die Rolle eines führungsstarken IT-Leiters wesentlich.

5.3.5 Branchenlösungen

Branchenlösungen

Zusätzlich zu den „großen“, branchenneutralen ERP-Systemen bieten die Hersteller von betriebswirtschaftlicher Standardsoftware „Branchenlösungen“ an [26, 27]. Im Abschnitt 5.3.1 wurde bereits darauf hingewiesen, dass der Begriff „Branche“ in der betriebswirtschaftlichen Literatur nicht sauber definiert ist. Die Hersteller betriebswirtschaftlicher Standardsoftware bildeten aus diesem Grund eigene Cluster von Funktionalitäten, um zu Softwarepaketen zu gelangen, die von ihnen Branchenlösungen genannt werden [27]. Wir betrachten dazu in Anlehnung an [27] das folgende Beispiel, das belegt, dass eine branchenspezifische Gruppierung von Informationsverarbeitungsfunktionalität nicht unbedingt den richtigen Weg darstellt.

Beispiel 5.3.15 (Branchenlösung) *Die Branchenlösung „High Tech“ der SAP AG soll für Halbleiterhersteller, PC- und Peripheriegerätehersteller, Hersteller von Consumer Electronics und Softwarehäuser gleichermaßen geeignet sein. Die in diesem Kurs an verschiedenen Stellen diskutierten Eigenschaften von Produktionsprozessen in Halbleiterfabriken lassen aber wenig Gemeinsamkeiten mit Abläufen in Softwarehäusern erkennen.*

Betriebstyp

Neben Branchen ist es sinnvoll, Betriebstypen zu untersuchen. Betriebstypen erhält man durch Betrachtung einer bestimmten Menge von Merkmalen mit ihren Ausprägungen. Wir betrachten dazu das nachfolgende Beispiel.

Beispiel 5.3.16 (Merkmale und Ausprägungen) *Ein Merkmal stellt die Unternehmensgröße mit den möglichen Ausprägungen Groß-, Mittel-, Klein- und Kleinstbetrieb dar. Ein zweites Merkmal ist der Fertigungstyp mit den möglichen Ausprägungen Massen- und Einzelfertigung.*

Die betrachteten Merkmale und die zugehörigen möglichen Ausprägungen sind in einer Betriebstypenmatrix anzuordnen. Die Einteilung in Betriebstypen ermöglicht eine flexiblere Ordnung als die Einteilung in Branchen, da es möglich ist, zusätzliche Merkmale und weitere Ausprägungen hinzuzunehmen. Die Notwendigkeit, eine Einordnung in ein disjunktes Gerüst vorzunehmen, entfällt somit [27].

Die Einteilung nach Betriebstypen besitzt die nachfolgend aufgezählten Schwächen bezüglich der Erreichung des Ziels, geeignete Software für Unternehmen zu entwickeln [27]:

- Die vorgeschlagenen Typologien sind häufig einseitig auf einen bestimmten Funktionsbereich des Unternehmens, eine spezielle Branche oder aber bestimmte Betriebsgruppen wie Zulieferer ausgerichtet. Es ist aber zu beachten, dass gleiche Merkmale auch andere Informationsverarbeitungsfunktionen außerhalb der betrachteten Unternehmen beeinflussen.

- In vielen Typologien werden die Merkmale als unabhängig voneinander angenommen. Betriebliche Ausprägungen bedingen bzw. beeinflussen sich aber häufig gegenseitig. Wir betrachten dazu das nachfolgende Beispiel.

  Beispiel 5.3.17 (Bedingtheit von Merkmalsausprägungen) *Das Merkmal „Fertigungstyp“ in der Ausprägung „kundenindividuelle Einzelfertigung“ weist darauf hin, dass ein komfortables Angebotssystem notwendig ist.*

- Die einem bestimmten Betriebstyp zugeordneten Informationsverarbeitungsfunktionen, Methoden und Parameter sind oft nicht genau bezeichnet. Wir betrachten dazu das folgende Beispiel.

  Beispiel 5.3.18 (Funktionalität) *Häufig ist offensichtlich, dass ein bestimmter Betriebstyp ein Modul zur Werkstattsteuerung benötigt. Es bleibt aber offen, welche Prioritätsregel gerade für diesen Betriebstyp besonders geeignet ist.*

  Letztendlich macht auch eine Betriebstypenbildung eine Parametrisierung wie bei branchenneutraler betriebswirtschaftlicher Standardsoftware nicht entbehrlich.

Kern-Schalen-Modell

Aus der bisherigen Diskussion folgt, dass weder Branche noch Betriebstyp alleine geeignet sind, als Klassifikationsmerkmal für betriebswirtschaftliche Standardsoftware zu dienen. Wir beschreiben deshalb im Folgenden in Anlehnung an [26, 27, 23] ein idealtypisches **Kern-Schalen-Modell**, das beide Kategorien geeignet vereint.

Im Kern ist die Funktionalität angesiedelt, die von einem Großteil der Unternehmen benötigt wird. In den Schalen befindet sich die Funktionalität, die von weniger Unternehmen verwendet werden kann.

Für den Kern werden branchen- und betriebstypunabhängige Bausteine vorgeschlagen. Der Kern besteht aus den zwei nachfolgenden Teilen, die aufeinander aufbauen:

- dem technischen Kern,
- dem betriebswirtschaftlichen Kern.

technischer Kern

Der technische Kern enthält Basissoftware. Wir betrachten das nachfolgende Beispiel für mögliche Basissoftware.

Beispiel 5.3.19 (Technischer Kern) *Betriebssysteme und Netzwerkunterstützungssoftware, sofern nicht bereits im Betriebssystem beeinhaltet, sowie Datenbankmanagementsysteme und Middleware können Bestandteil des technischen Kerns sein.*

Auf den technischen Kern setzt der betriebswirtschaftliche Kern auf. Dieser Kern besteht aus Komponenten, deren Funktionalität in vielen Unternehmen verwendet werden kann. Wir konkretisieren diese Aussage im nachfolgenden Beispiel. betriebswirtschaftlicher Kern

Beispiel 5.3.20 (Betriebswirtschaftlicher Kern) *Die folgende Funktionalität des Rechnungswesens:*

- *Finanzbuchhaltung,*
- *Debitoren-/Kreditorenbuchhaltung,*
- *Lohnbuchhaltung,*
- *Dienstreiseabrechnung*

kann Bestandteil eines betriebswirtschaftlichen Kerns sein. Für die Lagerhaltung kann die Bestandsfortschreibungsfunktionalität Bestandteil dieses Kerns sein. Die Fakturierung im Rahmen des Versands kann ebenfalls in den betriebswirtschaftlichen Kern übernommen werden.

Typischerweise sind nur wenige Komponenten von Branche bzw. Betriebstyp unabhängig. Aus diesem Grund wird der betriebswirtschaftliche Kern immer klein im Vergleich zu den Schalen sein.

Die betriebstypische Schale enthält Funktionalität, die in mehreren Branchen anwendbar ist. Wir betrachten dazu das nachfolgende Beispiel. betriebstypische Schale

Beispiel 5.3.21 (Betriebstypische Schale) *Häufig ist es sinnvoll, Materialspezifika in der Produktion zu berücksichtigen. In Molkereien muss aus diesem Grund der schwankende Fettgehalt der Milch durch die Planungssoftware geeignet abgebildet werden. Reihenfolgeabhängige Umrüstzeiten treten in der Chemie- und Elektronikbranche auf. Hier ist durch eine geeignete Ablaufplanungssoftware sicherzustellen, dass geeignete Bearbeitungsreihenfolgen für die Lose auf den Maschinen gewählt werden, so dass die Umrüstzeiten klein sind.*

In der Branchenschale wird Informationsverarbeitungsfunktionalität vorgehalten, die ausschließlich für die Anwendung in einer Branche vorgesehen ist. Wir betrachten dazu das nachfolgende Beispiel aus der Verschnittoptimierung in Anlehnung an [27]. Branchenschale

Beispiel 5.3.22 (Branchenschale) *Wir untersuchen Verschnittprobleme. Dabei handelt es sich abstrakt um das Problem, eine Fläche oder einen Raum derart in Teilflächen oder -räume aufzuteilen, dass zum einen die Abmessungen der Teilflächen und -räume eingehalten werden, andererseits aber möglichst wenig Teilbereiche ungenutzt bleiben. Die Teilbereiche, die übrig sind, werden als Verschnitt bezeichnet. Bei oberflächlicher Betrachtung erscheint das Verschnittproblem betriebstypisch. Das ist aber nicht der Fall, da bei der Beladung eines Containers mit Paketen dreidimensionale Verschnittprobleme*

behandelt werden müssen, während in der Glasindustrie zweidimensionale Verschnittprobleme gelöst werden müssen. Somit hat das Verschnittproblem branchentypische Ausprägungen, die zu unterschiedlichen Verfahren für die verschiedenen Branchen führen.

Schale für unternehmensindividuelle Funktionalität

Neben betriebs- und branchentypischer Funktionalität in den weiter innen gelegenen Schalen wird in der äußeren Schale unternehmensindividuelle Funktionalität vorgehalten. Das skizzierte Kern-Schalen-Modell ist in Abbildung 5.3 dargestellt.

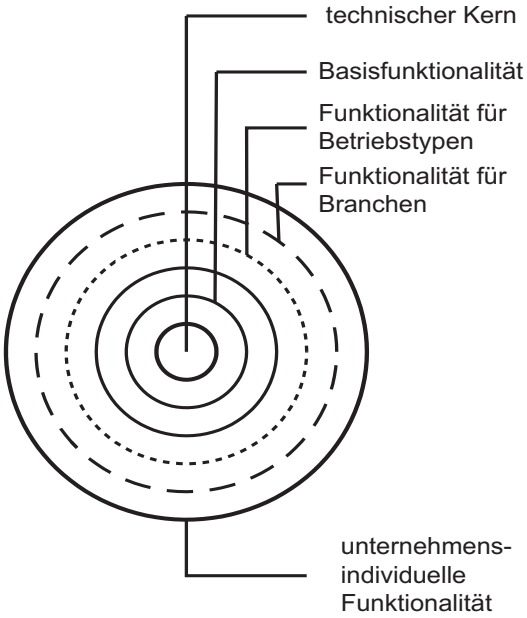

Abbildung 5.3: Kern-Schalen-Modell [23]

Übungsaufgabe 5.2 (Kern-Schalen-Modell) *Nennen und erläutern Sie Beispiele aus einem Unternehmen Ihrer Wahl für die äußere Schale des Kern-Schalen-Modells.*

Das Kern-Schalen-Modell weist eine gewisse Ähnlichkeit mit dem Architekturkonzept BCA auf, das in Abschnitt 2.6.3 behandelt wurde. Die Anwendungssystemebene von BCA wird aber durch das Kern-Schalen-Modell konkretisiert.

Wir erwarten, dass in Zukunft Informationsverarbeitungsfunktionalität durch Dienstekomposition bereitgestellt werden wird. Es ist sinnvoll, die einzelnen Basis- und Prozessdienste im Sinne von Abschnitt 4.4 dem Kern sowie den unterschiedlichen Schalen zuzuordnen. Auf das Kern-Schalen-Modell wird in Kurseinheit 7 im Rahmen der Diskussion des FABMAS-Prototyps erneut Bezug genommen.

5.3.6 ERP-Systeme als Beispiel für betriebswirtschaftliche Standardsoftware

5.3.6.1 Begriffsbildung und Architektur

ERP-Begriff

Ein aus unterschiedlichen Komponenten bestehendes integriertes Anwendungssystem zur Unterstützung der operativen Prozesse in den wesentlichen betrieblichen Funktionsbereichen

- Produktion,
- Beschaffung und Lagerhaltung,
- Finanz- und Rechnungswesen,
- Vertrieb,
- Personalwirtschaft

wird entsprechend Definition 1.1.31 als ERP-System bezeichnet [14]. Die Integration wird dabei in erster Linie durch zentrale Datenbanken unterstützt, die Daten-, Prozess- und Funktionsintegration ermöglichen. Historisch gesehen sind ERP-Systeme Nachfolger von PPS-Systemen, die im Vergleich zu der auf das Produktionsmanagement ausgerichteten Funktionalität der PPS-Systeme zusätzlich Funktionalität für das Finanz- und Rechnungswesen sowie für die Personalwirtschaft anbieten.

Transaktionssysteme

ERP-Systeme sind spezielle Systeme zur Abwicklung von Geschäftstransaktionen. Derartige Systeme werden auch als **Transaktionssysteme** bezeichnet. Kern dieser Klasse von Anwendungssystemen sind eine oder mehrere relationale Datenbanken, deren Inhalt zur Bearbeitung von Geschäftsvorfällen durch den Benutzer gelesen und geändert werden kann. Transaktionssysteme werden auch alternativ als **operative Systeme** bezeichnet, da sie zur Unterstützung der Abwicklung betrieblicher Prozesse dienen.

Architektur von ERP-Systemen

ERP-Systeme folgen der aus Abschnitt 2.6.2 bekannten mehrstufigen Client-Server-Architektur. Wir unterscheiden zwischen einer Benutzeroberfläche, der Präsentationsschicht zugeordnet, verschiedenen Anwendungskomponenten, die zur Anwendungsschicht gehören, und dem Basissystem. Das Basissystem besteht aus einem Anwendungskern sowie Integrations- und Entwicklungskomponenten. Es interagiert mit einer oder mehreren Datenbanken. Die beschriebene Situation ist in Abbildung 5.4 dargestellt.

Die einzelnen Schichten werden als Subsysteme realisiert, die einer Client-Server-Architektur folgen. Die Subsysteme sind in vielen Fällen hardware- und betriebssystemunabhängig. Sie sind in der Lage, mit unterschiedlichen relationalen Datenbanksystemen zusammenzuarbeiten. Im Prinzip können Subsysteme (Komponenten) unterschiedlicher Hersteller zu einem Anwendungssystem zusammengefügt werden. In der Praxis steht diesem Ansatz allerdings die grobgranulare Struktur heutiger ERP-Systeme entgegen.

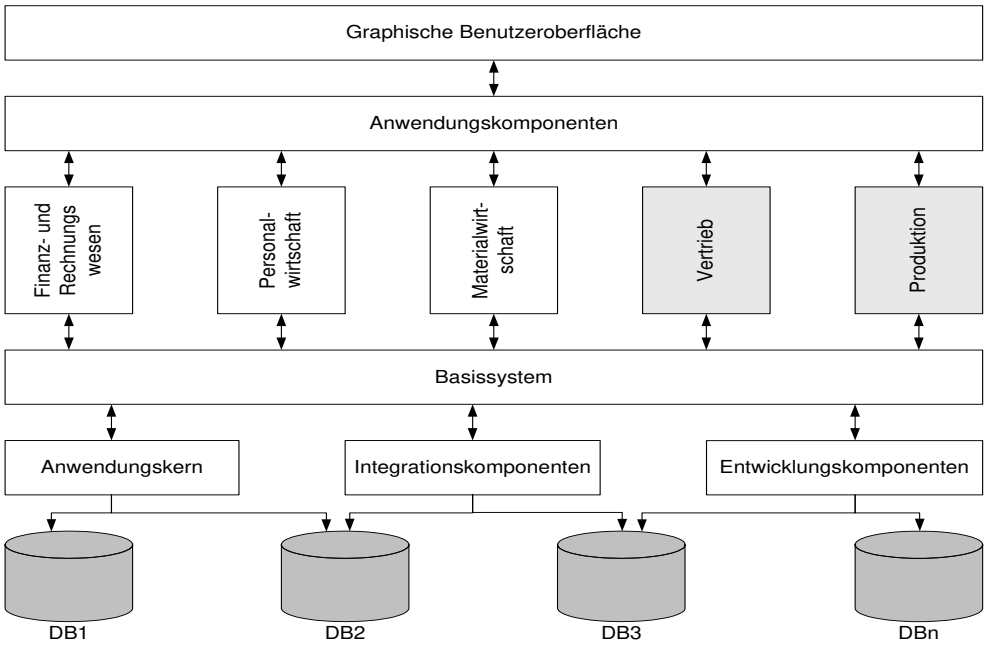

Abbildung 5.4: Grundlegende ERP-Architektur

Wir beschreiben in den nächsten beiden Abschnitten das Basissystem sowie die betriebswirtschaftlichen Komponenten „Finanz- und Rechnungswesen“, „Personalwirtschaft“ sowie „Materialwirtschaft“ am Beispiel von mySAP ERP. Die Ausführungen sind aber auf andere ERP-Systeme prinzipiell übertragbar. Die Untersuchung der Funktionalität, die in den Komponenten „Produktion“ und „Vertrieb“ eines ERP-Systems enthalten ist, wird auf Kurseinheit 6 verschoben.

5.3.6.2 SAP R/3 und mySAP ERP

SAP R/3 und mySAP ERP

Das ERP-System mySAP ERP ist der Nachfolger der betriebswirtschaftlichen Standardsoftware SAP R/3, die schon an verschiedenen Stellen dieses Kurses meistens in Form von Beispielen behandelt wurde. Im Jahr 2000 waren 24.000 R/3-Installationen vorhanden. Es gab eine Million lizensierter Benutzer von SAP R/3 [14].

NetWeaver

mySAP ERP ist eine ERP-Lösung auf Basis von NetWeaver. NetWeaver ist eine Architektur und Middleware, die Anwendern betriebsübergreifende Geschäftsprozesse durch eine Integration unterschiedlicher IT-Systeme der SAP AG sowie anderer Softwareanbieter gestattet.

Auf Basis der NetWeaver-Technologie werden durch die SAP AG SAP xApps

Composite-Application

Packaged Composite Applications bereitgestellt. Wir definieren zunächst den Begriff einer Composite-Application in Anlehnung an [45].

Definition 5.3.5 (Composite-Application) *Eine Anwendung, die über Dienstaufrufe auf bereits existierende Daten und Funktionen zugreift, die*

durch vorhandene Anwendungssysteme einer Systemlandschaft zur Verfügung gestellt werden, wird als Composite-Application bezeichnet. Die Daten und Funktionen werden durch die Composite-Application zu neuen Geschäftsprozessen kombiniert. Zusätzlich wird eine neue Oberfläche sowie Geschäftslogik hinzugefügt.

Eine Composite-Application nutzt somit einerseits Anwendungsdienste, andererseits aber auch Systemdienste. Diese steuern den Ablauf und die Fehlerbehandlungen in Composite-Applications [14]. Somit stellt der von SAP geprägte Begriff der Composite-Application letztendlich nur eine Spezialisierung des in Abschnitt 2.3.4.3 eingeführten Begriffs der Dienstekomposition dar.

Composite-Applications haben den Vorteil, dass auch Altsysteme über Webservices in die entstehende Anwendung eingebunden werden können.

Die mySAP Business Suite basiert ebenfalls auf der NetWeaver-Technologie. Die Business Suite umfasst neben mySAP ERP Komponenten für unternehmensübergreifende Anwendungen wie Lieferkettenmanagement (SCM) oder das Kundenbeziehungsmanagement (CRM).

5.3.6.3 Basissystem

Basissystem

Das Basissystem stellt eine Infrastruktur für die Anwendungskomponenten zur Verfügung. Außerdem werden Schnittstellen zur Datenbank und zur Benutzeroberfläche, wie in Abbildung 5.4 gezeigt, durch das Basissystem bereitgestellt.

Das Basissystem enthält weiterhin Funktionen zur zentralen Steuerung des ERP-Systems. Diese Funktionen sind im Einzelnen:

- Administrationsfunktionen wie die Benutzerverwaltung,
- Schnittstellen zum Betriebssystem,
- Customizingfunktionen,
- die Entwicklungsumgebung sowie verschiedene Programmierschnittstellen.

Benutzerverwaltung

Die Benutzerverwaltung ist ein wesentlicher Bestandteil des Basisystems. Durch ein Berechtigungskonzept wird festgelegt, welcher Nutzer Zugriff auf welche Funktionalität des ERP-Systems hat. Aufgrund des hohen Verwaltungsaufwands sind personenbezogene Berechtigungen ungeeignet. In mySAP ERP findet deshalb ein rollenbasiertes Konzept Anwendung. Die für bestimmte Stellen notwendigen Zugriffsberechtigungen werden in einer Rolle gebündelt. Diese kann dann unterschiedlichen Benutzern zugewiesen werden [14].

Das Basissystem enthält Infrastrukturfunktionen zur Nutzung von modernen Webtechnologien. Außerdem werden auch Anwendungsfunktionen mit Querschnittscharakter wie das Workflow-Management in das Basisystem aufgenommen.

Die SAP AG bezeichnet nach der Einführung der NetWeaver-Technologie das

Basissystem als umfassende „Integrations- und Anwendungsplattform“. Durch NetWeaver werden die folgenden Bereiche unterstützt [6]:

- Das NetWeaver-Portal ermöglicht die Einbindung von menschlichen Aufgabenträgern in Geschäftsprozesse. Das Portal stellt einen webbasierten Einstiegspunkt für Endanwender zu Informationsquellen, Anwendungen und Daten zur Verfügung.

- SAP Business Information (BI) umfasst Funktionen zur Datenintegration und das Data-Warehouse SAP BW. SAP BW ist in der Lage, Daten aus einer Vielzahl von Datenquellen zu extrahieren und zu nutzen. Der BW-Server stellt Werkzeuge zum Modellieren, Extrahieren, Speichern und Aufrufen der Daten zur Verfügung.

Prozessintegration

- Die für eine prozesszentrierte Zusammenarbeit von SAP- und Nicht-SAP-Anwendungen erforderlichen Funktionen werden von der „Exchange Infrastructure (XI)“ zur Verfügung gestellt. XI bietet eine Plattform zur Prozessintegration (PI) an. Unter Verwendung von XI/PI können systemübergreifend Geschäftsprozesse abgebildet werden. Die Konzepte, die bei XI/PI zum Einsatz kommen, sind denen in ESB-Architekturen ähnlich (vergleiche hierzu die Ausführungen in Abschnitt 2.3.4.4).

Laufzeitumgebung

- Die Anwendungsserver AS-ABAP und AS-Java stellen die Laufzeitumgebungen für alle SAP-basierten Anwendungen zur Verfügung. Advanced-Business-Programming-Language (ABAP) ist eine proprietäre Programmiersprache der SAP AG [3]. AS-ABAP stellt somit eine Weiterentwicklung der klassischen SAP Basis dar. Die ABAP-Laufzeitumgebung ermöglicht die Entwicklung von betriebswirtschaftlichen Programmen auf einer hardware-, betriebssystem- sowie datenbankunabhängigen Plattform. AS-Java unterstützt die durch J2EE bzw. JEE5 definierten Standards für Enterprise-Java-Anwendungen.

5.3.6.4 Betriebswirtschaftliche Komponenten

Wir gehen der Reihe nach auf die Bereiche

- Finanz- und Rechnungswesen,

- Personalwirtschaft,

- Materialwirtschaft

ein. Wir definieren dazu zunächst in Anlehnung an [14] den Begriff Finanz- und Rechnungswesen.

Definition 5.3.6 (Finanz- und Rechnungswesen) *Der Bereich Finanzierung und Investition des Finanz- und Rechnungswesens beinhaltet die Bereitstellung und zielgerichtete Verwendung finanzieller Mittel. Die Aufzeichnung, Berichterstattung und Analyse der durch die betrieblichen Leistungsprozesse entstehenden finanziellen Transaktionen ist Gegenstand des Bereichs Rechnungswesen.*

Finanz- und Rechnungswesen

Nachfolgend betrachten wir exemplarisch lediglich die Finanzbuchhaltung, die Bestandteil des Rechnungswesens ist. Die Finanzbuchhaltung dient dazu, alle finanziellen Geschäftsvorfälle eines Unternehmens auf Konten aufzuzeichnen [14]. Verschiedene Kontentypen existieren. Wir betrachten dazu das folgende Beispiel.

Beispiel 5.3.23 (Kontentyp) *Bestandskonten dienen zur Erstellung der Bilanz. Eine Bilanz ist eine Gegenüberstellung des Vermögens und der Kapitalstruktur zu einem bestimmten Stichtag. Jedem Posten der Bilanz wird ein Bestandskonto zugeordnet. Auf Erfolgskonten werden erfolgswirksame Geschäftsvorfälle gebucht. Erfolgskonten gehen in die Gewinn- und Verlustrechnung ein. Die Bilanz und die Gewinn- und Verlustrechnung werden als Hauptbücher bezeichnet.*

In mySAP ERP baut die Organisationsstruktur der Finanzbuchhaltung auf Konten auf. Die einzelnen Konten gehören zu einem Kontenplan. Jeder Kontenplan ist genau einem Mandanten zugeordnet. Unter einem **Mandanten** verstehen wir eine handelsrechtlich, organisatorisch sowie datentechnisch abgeschlossene Einheit innerhalb eines SAP-Systems. Ein Mandant verfügt über eigene Stammdaten, die in separaten Tabellen für diesen Mandanten vorgehalten werden. Auf die Abbildung der Organisationsstruktur eines Unternehmens in SAP R/3 und mySAP ERP wird in Abschnitt 5.3.8 dieser Kurseinheit genauer eingegangen.

Mandant

Konten sind Kontenarten und -gruppen zugeordnet. Die Kontenarten

Kontenart

- Hauptbuchkonten,
- Debitoren,
- Kreditoren

existieren. Kreditoren sind Lieferanten. Debitoren werden als Kunden bezeichnet.

Jeder Geschäftsvorfall wird in mySAP ERP durch einen Beleg erfasst. Ein Beleg umfasst einen Belegkopf und Belegpositionen. Der Belegkopf enthält Daten, die für den gesamten Beleg gültig sind. Belegpositionen dienen dazu, die erbrachten Leistungen abzubilden.

Nach der knappen Einführung in das Finanz- und Rechnungswesen beschäftigen wir uns mit der Personalwirtschaft. Der Begriff der Personalwirtschaft wird wie folgt definiert [14].

Personalwirtschaft

Definition 5.3.7 (Personalwirtschaft) *Die Bereitstellung und der zielgerichtete Einsatz von Mitarbeitern in Unternehmen wird als Personalwirtschaft bezeichnet.*

Personalinformationssysteme haben die Aufgabe, die Personalabteilung bei der Stammdatenverwaltung zu unterstützen. Personal- und Fachabteilungen nutzen ein Personalinformationssystem für die nachfolgenden Tätigkeiten:

- Personaladministration,
- Personalplanung,
- Personalbeschaffung,
- Personalentwicklung,
- Personalführung.

In SAP R/3 und mySAP ERP heißt die Komponente, die als Personalinformationssystem fungiert, Human Resources (HR). Personaladministration, Personalplanung sowie Personalführung werden in der Gesamtheit in HR als Personalcontrolling bezeichnet.

Materialwirtschaft

Wir betrachten nun abschließend die Materialwirtschaft. Der Begriff der Materialwirtschaft wird wie folgt definiert [14].

Definition 5.3.8 (Materialwirtschaft) *Die Planung, Steuerung, Verwaltung und Überwachung der Materialbestände und -bewegungen innerhalb eines Betriebs sowie zwischen dem Betrieb und Lieferanten, Kunden sowie Distributionsdienstleistern wird als Materialwirtschaft bezeichnet.*

Die wichtigsten Aufgaben der Materialwirtschaft sind nachfolgend aufgeführt:

- Einkauf,
- Bestandsführung,
- Disposition,
- Rechnungsprüfung.

Da in der Industrie die Produktion mit Roh-, Hilfs- und Betriebsstoffen, Zulieferteilen sowie Halbfabrikaten versorgt werden muss, sind Produktion und Materialwirtschaft eng verzahnt. Die Materialwirtschaft wird im Handel als Warenwirtschaft bezeichnet.

Da wir auf den Einkauf bereits in Abschnitt 2.3.1.5 eingegangen sind, verzichten wir an dieser Stelle auf eine Beschreibung. Die **Materialstammdatenverwaltung** bildet die Grundlage für alle Tätigkeiten der Materialwirtschaft. Sie dient der Pflege der Stammdaten, die zur Beschreibung der Eigenschaften, der Zusammensetzung und der Verwendungszwecke der Materialien dient. Wir betrachten das folgende Beispiel für die Zusammensetzung von Materialien.

Beispiel 5.3.24 (Zusammensetzung von Materialien) *Die bereits mehrfach in diesem Kurs betrachteten Stücklisten, die Erzeugnisstrukturen abbilden, beschreiben die Zusammensetzung von Materialien.*

Das Konzept der **Materialsichten** ist für die Abbildung des Materialstammes in mySAP ERP charakteristisch. Durch die Einteilung der Daten in verschiedene Sichten kann eine Pflege des Materialstammes durch mehrere Abteilungen vorgenommen werden. Wir betrachten dazu das folgende Beispiel.

Beispiel 5.3.25 (Sichten im Materialstamm) *Zu den Grunddaten gehören u.a. Beschreibungen, Basismengeneinheiten sowie die Europäische Artikelnummer (EAN). Diese Daten werden in der Sicht „Grunddaten“ vorgehalten.*

Die in Beispiel 5.3.25 betrachtete Sicht „Grunddaten“ wird auf Mandantenebene gepflegt. Sichten wie „Einkauf“, „Prognose“ oder „Kalkulation“ hingegen werden auf Werksebene gepflegt. Im Rahmen des Customizings können individuelle Sichten angelegt werden.

Materialart

Die **Materialart** legt fest, welche Materialsichten verfügbar sind und welche Felder gepflegt werden müssen. Außerdem schreibt die Materialart vor, wie die Materialien zu beschaffen sind und welche Bestandsführungspflichten bestehen. Die nachfolgenden Materialarten werden unter anderem in mySAP ERP angeboten:

- Rohstoffe,
- Fertigerzeugnisse,
- Handelswaren,
- Nicht-Lagermaterial,
- Dienstleistungen.

Bestandsführung

Materialbestände und -bewegungen werden wert- und mengenmäßig in der Bestandsführung erfasst. Die Parameter für die Bestandsführung werden im Materialstamm gepflegt. In Abhängigkeit von der Verwendung des Materials müssen verschiedene Sichten verwendet werden. Wir betrachten dazu das nachfolgende Beispiel.

Beispiel 5.3.26 (Bestandsführung) *Falls ein Material wertmäßig erfasst werden soll, muss die Sicht „Buchhaltung“ gepflegt werden. In der Buchhaltungssicht werden die Einstellungen für die Bewertung vorgenommen. Der Wert eines Materials wird berechnet als Produkt aus Bestandsmenge und Materialpreis. Die Materialbewertung wird u.a. für die Bilanz verwendet.*

Warenbewegungen werden in mySAP ERP mit Belegen erfasst. Beim Wareneingang oder -ausgang im Lager wird eine Verbuchung vorgenommen. Das System unterscheidet zwischen externen und internen Bewegungsarten. Wir betrachten dazu ein Beispiel in Anlehnung an [14].

Beispiel 5.3.27 (Bewegungsarten) *Der Verkauf von Waren an Kunden stellt eine externe Bewegungsart dar. Durch einen Verkauf wird offensichtlich der Bestand verändert. Die Umlagerung von Materialien innerhalb eines Betriebs von einem ersten Lagerort an einen zweiten ist eine interne Warenbewegung. Im Gegensatz zu externen Warenbewegungen wird dadurch der Gesamtbestand nicht verändert.*

Im Rahmen einer Inventur werden die tatsächlich vorhandenen Materialien mit der Bestandsführung abgeglichen. mySAP ERP unterstützt Inventuren.

Disposition

Die Disposition legt fest, welches Material zu welchen Zeitpunkten in welcher Menge benötigt wird. Sie überwacht Lagerbestände. Unter Berücksichtigung von eingehenden Aufträgen bzw. Prognosen über die Bestandsentwicklung werden die Bestände vorgeplant.

Wir unterscheiden zwischen manueller und automatischer Disposition. Bei der manuellen Disposition wird eine Bestellanforderung an den Einkauf übergeben. Die automatische Disposition basiert üblicherweise auf dem **Bestellpunktsystem**. Die Grundidee des Bestellpunktsystems besteht darin, dass der Bestand aller Materialien ständig mit dem Melde- und dem Sicherheitsbestand verglichen wird. Falls der Lagerbestand durch Warenabgänge, Materialreservierungen oder aufgrund von Bedarfsprognosen den Meldebestand unterschreitet, wird eine Bestellung ausgelöst [22]. Die Höhe des Meldebestandes, mit MBH bezeichnet, ergibt sich wie folgt:

$$MBH := E(BBZ) + SBH, \tag{5.1}$$

wobei wir mit BBZ den Bedarf während der Beschaffungszeit bezeichnen. In Beziehung (5.1) dient E wie üblich zur Bezeichnung des Erwartungswertes. Die Höhe des Sicherheitsbestandes SBH ist die Differenz aus dem erwarteten maximalen und durchschnittlichen Bedarf während der Beschaffungszeit.

Die Disposition erfolgt entweder plan- oder verbrauchsgesteuert. Bei der planbasierten Disposition werden Bedarfe im Rahmen der Absatz- und Produktionsplanung ermittelt (vergleiche dazu die Ausführungen in Abschnitt 6.1.2). Bei der verbrauchsgesteuerten Disposition werden hingegen Verbräuche der Vergangenheit dazu verwendet, um mit Hilfe von Prognoseverfahren zu erwartende Bedarfe vorherzusagen. mySAP ERP bietet dazu verschiedene Prognoseverfahren an.

Rechnungsprüfung

Die Rechnungsprüfung dient dazu, die Bestellungen mit den Wareneingangsanzeigen sowie den Eingangsrechnungen zu vergleichen und auf sachliche Richtigkeit hin zu untersuchen. Die Erstellung von Wareneingangsanzeigen erfolgt bei der Warenannahme sowie beim Eingang ins Lager. Wenn bei der Rechnungsprüfung Fehler auffallen, so wird die Rechnung gesperrt und zunächst nicht bezahlt.

Anhand der in diesem Abschnitt vorgestellten Aufgaben eines ERP-Systems in den Bereichen Finanz- und Rechnungswesen, Personal- sowie Materialwirtschaft ist zu sehen, dass der Betrieb eines ERP-Systems neben technischem vor allen Dingen auch betriebswirtschaftliches Wissen voraussetzt.

5.3.6.5 Extended-ERP- und ERP-II-Konzept

Die Einbindung von Kunden und Lieferanten in die Informationssysteme der einzelnen Unternehmen wurde durch Internettechnologien Mitte bis Ende der 90er Jahre ermöglicht. Klassische ERP-Systeme können diese neuen Anforderungen nur unzureichend unterstützen. Aus diesem Grund wurden ERP-Systeme um unternehmensübergreifende Anwendungen angereichert. Das führt zum Begriff des Extended-ERP. Wir definieren diesen Begriff in Anlehnung an [46] wie folgt.

Extended-ERP

Definition 5.3.9 (Extended-ERP-System) *Ein ERP-System, das Funktionserweiterungen zur zwischenbetrieblichen Prozessunterstützung enthält, aber noch nicht alle zwischenbetrieblichen Funktionen vollständig abbildet, wird Extended-ERP-System genannt.*

ERP-II

Wir definieren nun den Begriff eines ERP-II-Systems [46].

Definition 5.3.10 (ERP-II-System) *Ein integriertes Anwendungssystem, das neben der ERP-Funktionalität sämtliche unternehmensübergreifenden Funktionen beinhaltet, wird als ERP-II-System bezeichnet.*

Offensichtlich sind Extended-ERP-Systeme zwischen ERP-Systemen und ERP-II-Systemen angesiedelt. Ein ERP-System unterstützt Geschäftsprozesse eines einzelnen Unternehmens. Ein Extended-ERP-System beschäftigt sich mit ausgewählten Teilbereichen der gesamten Wertschöpfungskette. Hansen und Neumann bezeichnen in [14] Extended-ERP- und ERP-II-Systeme als integrierte E-Business-Systeme.

Wir weisen darauf hin, dass durch die NetWeaver-Technologie und die dadurch gegebenen Möglichkeiten zur Realisierung von Composite-Applications Extended-ERP- bzw. sogar ERP-II-Systeme realisiert werden können. Die ebenfalls auf der NetWeaver-Technologie basierende mySAP Business Suite mit mySAP ERP als Kern ist ebenfalls als Extended-ERP-System aufzufassen.

5.3.7 Auswahl betriebswirtschaftlicher Standardsoftware

5.3.7.1 Phasenmodell für die Auswahl von betriebswirtschaftlicher Standardsoftware

Auswahlphase

Die Einführung betriebswirtschaftlicher Standardsoftware untergliedert sich in die Auswahl eines Softwareanbieters und anschließend in die Anpassung der ausgewählten Standardsoftware an konkrete betriebliche Anforderungen. Die Aufgaben der Auswahlphase bestehen in der

- Definition von Anforderungen,
- Auswahl eines geeigneten Produkts aus dem verfügbaren und bekannten Marktangebot.

Um die Auswahlphase erfolgreich durchführen zu können, sind zum einen Kenntnisse über bisherige betriebliche, insbesondere organisatorische Abläufe notwendig. Weiterhin sind methodische Erfahrungen in der Identifikation und Beurteilung von Anbietern und Produkten erforderlich.

Auswahlprozess

Der Auswahlprozess wird in die nachfolgenden sechs Phasen gegliedert [13]:

1. Zielfindungsprozess,
2. Anforderungsanalyse,
3. Marktübersicht,
4. Screening,
5. Endauswahl,
6. endgültige Entscheidung.

Der **Zielfindungsprozess** dient der Festlegung von Zielen, die mit der Einführung einer neuen Standardsoftware verbunden sind. Typischerweise wird zwischen organisatorischen und technischen Zielen unterschieden.

Aufbauend auf den Zielen wird eine **Anforderungsanalyse** durchgeführt. Ergebnis der Anforderungsanalyse ist eine Menge von Anforderungen, welche die Aufgaben der betriebswirtschaftlichen Standardsoftware festlegen.

Bei der **Erstellung der Marktübersicht** handelt es sich um die Erstellung einer Liste von Standardsoftwareprodukten, die den herausgearbeiteten Anforderungen genügen. Die Marktübersicht wird durch Literaturstudium und durch eine Anbieterbefragung gewonnen.

Während der **Screening-Phase** wird eine bestimmte Untermenge der prinzipiell möglichen Standardsoftwareprodukte detaillierter untersucht. Die jeweilige Software kann dabei in Workshops vorgestellt und exemplarisch auf Stamm- und Bewegungsdaten des Unternehmens angewandt werden. Außerdem sind auch zeitlich begrenzte Testinstallationen im jeweiligen Unternehmen möglich.

In die **Endauswahl** gelangen als Ergebnis der Screening-Phase maximal drei mögliche betriebswirtschaftliche Standardsoftwarepakete. Mit den jeweiligen Anbietern werden Vertragsverhandlungen aufgenommen.

Unter Berücksichtigung der Ergebnisse der Vertragsverhandlungen wird dann abschließend die **endgültige Entscheidung** über die Auswahl eines Standardsoftwareproduktes getroffen. Das beschriebene Vorgehen ist in Abbildung 5.5 noch einmal veranschaulicht.

Anbieterauswahl

Die nachfolgende Tabelle 5.1 macht deutlich, dass die Kosten für die Anbieterauswahl wesentlich geringer als für den betrieblichen Einführungsprozess einer Standardsoftware sind. Aus diesem Grund ist es nicht verständlich, wenn bei der Auswahlentscheidung gespart wird.

Die folgenden Fehler können bei der Anbieterauswahl begangen werden [13]:

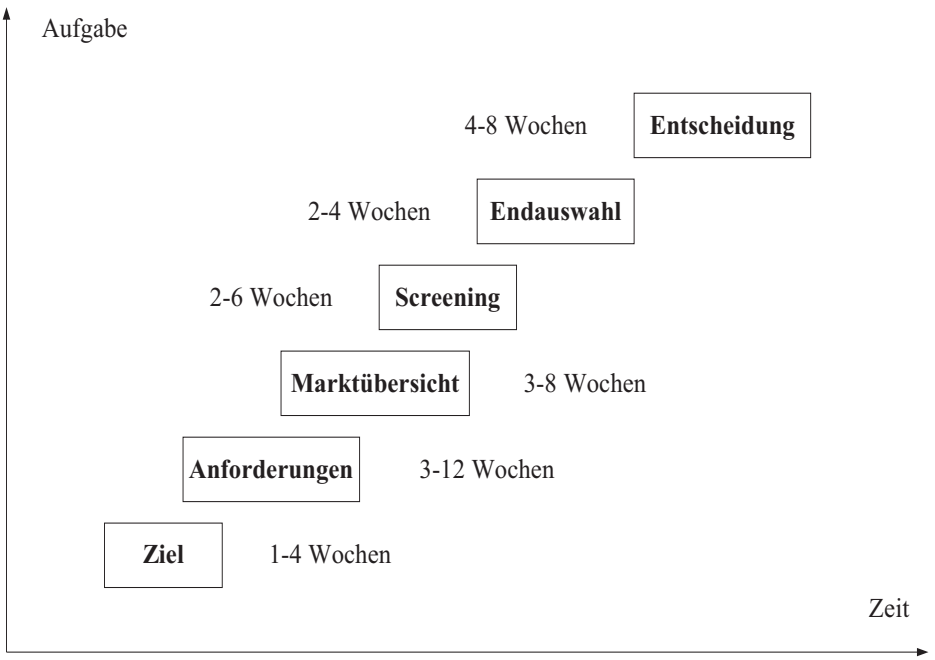

Abbildung 5.5: Phasen des Auswahlprozesses [13]

1. **Verschwommene bzw. unartikulierte Zielsetzungen**: Zu dieser Klasse von Fehlern zählen der Wunsch nach höherer Wirtschaftlichkeit als Beispiel für ein verschwommenes Ziel sowie der Wunsch nach Verringerung des Personals in der IT-Abteilung als Beispiel für ein unartikuliertes Ziel.

2. **Überzogene Erwartungshaltung**: Es wird erwartet, dass die neu eingeführte Standardsoftware alle vorhandenen Probleme wie fehlende Daten oder ineffiziente organisatorische Abläufe löst.

3. **Verzicht auf Analysen und Konzepte**: Dadurch wird der Suchraum für mögliche Standardsoftwarepakete häufig stark eingeschränkt. Zu wenig Kriterien für die Anbieterauswahl sind eine Folge dieses Vorgehens. Eine systematisch gewonnene Übersicht über das Markangebot fehlt. Auf Grund der nicht erstellten Konzepte ist eine einheitliche Vergleichs- und Bewertungsgrundlage nicht vorhanden. Die Angaben der Anbieter über Funktions- und Lösungskompetenz können nicht überprüft werden. Häufig findet überhaupt kein Anbietervergleich statt, da langjährige Zusammenarbeit mit einem Anbieter vorliegt oder Konzernentscheidungen ungeprüft übernommen werden. Die Konzentration auf eine favorisierte Hardwareplattform ist ebenfalls ein Fehler, wenn darauf aufbauend die Auswahl der Software vorgenommen wird.

Tabelle 5.1: Gegenüberstellung von Auswahl und Einführung von Standardsoftware [13]

Kriterium	Auswahl	Einführung
Kosten	gering	hoch
Beeinflussung der betrieblichen Abläufe im Unternehmen	gering	hoch
erforderliche Kenntnisse	betriebliche Abläufe, Marktübersicht	betriebliche Abläufe, ausgewählte Standardsoftware
Notwendigkeit der Hinzunahme externer Berater	nicht erforderlich	häufig erforderlich
Projektmanagement	geringe Anforderungen	hohe Anforderungen
Dauer	gering	hoch

4. **Lange Dauer des Auswahlprozesses**: Durch einen zu lange andauernden Prozess der Auswahl wird der Erfolg der Softwareeinführung gefährdet, da sich wesentliche Annahmen dann bereits geändert haben können.

5.3.7.2 Exkurs über Projektorganisation

Projektorganisation

Die Auswahl und Einführung einer umfassenden betriebswirtschaftlichen Standardsoftwarelösung werden als Projekt durchgeführt. Aus diesem Grund werden in diesem Abschnitt grundlegende Aussagen zur Projektorganisation getroffen.

Projekt

Für weitergehende Informationen zum IT-Projektmanagement verweisen wir auf [7]. Wir definieren zunächst den Projektbegriff.

Definition 5.3.11 (Projekt) *Ein Vorhaben, das im Wesentlichen durch die Einmaligkeit seiner Bedingungen gekennzeichnet ist, wird Projekt genannt. Unter Bedingungen verstehen wir dabei:*

- *Zielvorgaben,*
- *zeitliche, finanzielle und personelle Bedingungen der Projektabwicklung,*
- *klare Abgrenzung gegenüber anderen Vorhaben,*
- *bestimmte projektspezifische Organisationsformen.*

Projektparameter

Vor Beginn eines Projektes sind nach Möglichkeit die folgenden Parameter zu dokumentieren:

- Umfang des Projektes,
- Dauer des Projektes,
- projektspezifische Besonderheiten, wie die Erfahrung der Mitarbeiter, die für das Projekt zur Verfügung stehen,
- Komplexität,
- erwartete Schwierigkeiten,
- Bedeutung des Projektes für Auftragnehmer und Auftraggeber wie Prestigegewinn,
- Risiko bezüglich Technik, Zeitbedarf, finanziellem Aufwand bzw. Zielerreichung,
- Kontinuität und Intensität, d.h. Art und Weise des Mitarbeitereinsatzes,
- Organisation und Führungsverständnis.

Projektorganisation

Eine Projektorganisation bzw. deren Struktur wird einerseits bestimmt durch die Zuordnung von Projektaufgaben zu Personen und andererseits durch die Herstellung von Kommunikationsbeziehungen zwischen den am Projekt beteiligten Personen. Projekte werden wie folgt strukturiert:

- Teilprojekte,
- Teilprojekte werden in Phasen zerlegt,
- jede Phase besteht aus Arbeitspaketen,
- jedes Arbeitspaket besteht aus Aktivitäten,
- jede Aktivität besteht aus Aufgaben.

Die Struktur von Projekten ist in Abbildung 5.6 veranschaulicht.

Die **Aufgaben des Projektleiters** bestehen in der Projektplanung und -steuerung, der inhaltlichen Ausgestaltung und Kontrolle der Projektergebnisse, der Unterstützung des Projekterfolgs durch Beschaffen der benötigten Ressourcen sowie in der Festlegung der projektspezifischen Organisationsform. Ein Projektleiter ist insbesondere auch für die Festlegung des Projektinhalts verantwortlich. Das verlangt eine Untersetzung des Projektziels durch Teilziele sowie die Konkretisierung und Detaillierung der zu lösenden Aufgaben.

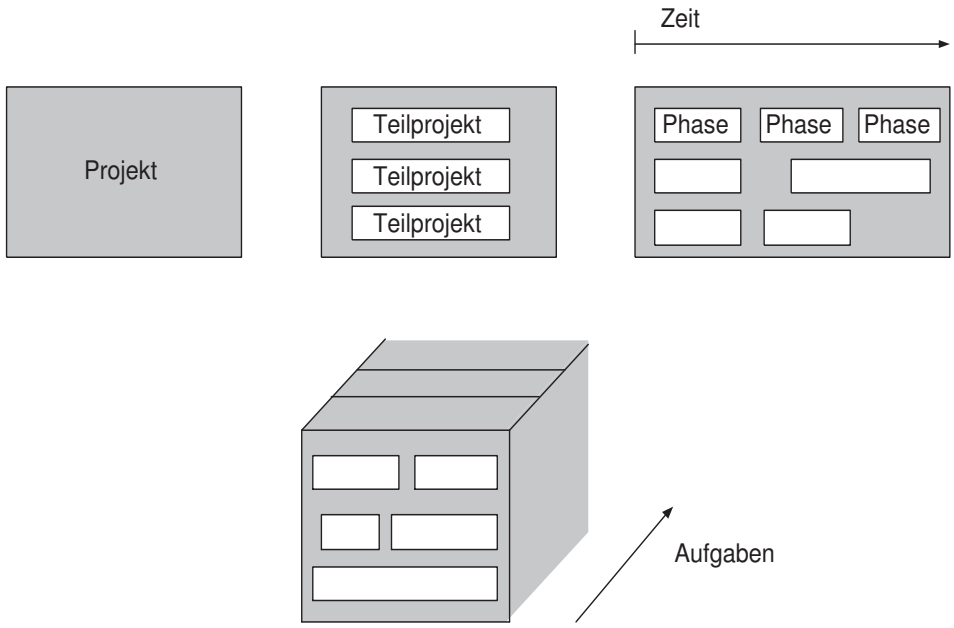

Abbildung 5.6: Grundlegende Struktur von Projekten

5.3.7.3 Zieldefinition und Schrittfolge zur Auswahl von betriebswirtschaftlicher Standardsoftware

Zieldefinition

Eine formale Zieldefinition ist aus den folgenden Gründen notwendig:

1. Orientierung für Entscheidungen innerhalb der Projektlaufzeit,
2. Projektcontrolling nach der Einführung der betriebswirtschaftlichen Standardsoftware auch hinsichtlich der Projektziele.

Eine Zieldefinition besitzt die folgenden Bestandteile [13]:

1. Ausgangsdefinition,
2. angestrebte organisatorische Verbesserungen,
3. angestrebte technische Verbesserungen,
4. Zieltermin,
5. angestrebte Verbesserungen der Wettbewerbssituation,
6. voraussichtliches Budget.

Das folgende Beispiel soll die notwendigen Bestandteile einer Zieldefinition genauer veranschaulichen.

Beispiel 5.3.28 (Zieldefinition) *Das zur Zeit eingesetzte MES im Unternehmen W-Fab erfüllt in mehreren Punkten nicht mehr die im Unternehmen bestehenden Anforderungen. Ziel ist es, die bisher benutzte Individualsoftwarelösung durch eine Standardsoftware abzulösen. Das neue Softwaresystem soll den Zeitaufwand, der bisher für das Lostracking, für die Ermittlung gestoppter Lose, für das Einpflegen neuer Produkte sowie für die Übernahme von Kundenauftragsdaten aus dem PPS-System erforderlich ist, wesentlich verkürzen. Das System soll dem heutigen Stand der Technik, was Datenbank, Mehrbenutzerbetrieb und browserbasierte Benutzeroberfläche betrifft, entsprechen. Zusätzlich soll es möglich sein, eine spezielle Ablaufplanungssoftware zu implementieren, die auf den im MES vorhandenen Stamm- und Bewegungsdaten aufsetzt. Mit Beginn des nächsten Wirtschaftsjahres soll das neue Softwaresystem produktiv zur Verfügung stehen. Durch das neu einzuführende System soll die Termintreue der Halbleiterfabrik erhöht werden. Dadurch wird eine verbesserte Kundenbindung erreicht. Weiterhin sind aufgrund einer verbesserten Termintreue weniger Konventionalstrafen fällig. Insgesamt ergibt sich dadurch eine Verbesserung der Wettbewerbsfähigkeit des Unternehmens. Für das Projekt steht im ersten Jahr ein Budget von x Euro, im zweiten Jahr ein Budget von y Euro zur Verfügung. Mit der Auswahl eines geeigneten Systems wird eine Arbeitsgruppe unter Frau K. beauftragt.*

Auswahlschritte

Nach erfolgter Zieldefinition sind die betriebswirtschaftlichen Ziele durch entsprechende Kenngrößen zu untersetzen. Anschließend sind die nachfolgenden Schritte zur Auswahl von betriebswirtschaftlicher Standardsoftware erforderlich:

1. Ziele operationalisieren,
2. Einflussgrößen bestimmen,
3. Einflussgrößen operationalisieren,
4. Zuordnung von Einflussgrößen zu Zielen,
5. Nutzenbestimmung,
6. Aufwandsbestimmung,
7. Gegenüberstellung von Nutzen und Aufwand,
8. Gesamtbewertung durchführen.

Operationalisierung von Zielen

Wir gehen nun auf die einzelnen Schritte detailliert ein. Nachdem die Ziele festgelegt worden sind, ist es erforderlich, diese quantitativ zu erfassen. Dazu muss eine Skala eingeführt werden, auf deren Basis eine Bewertung der Zielerreichung erfolgen kann. Für Ziele, die leicht zu operationalisieren sind, ist es möglich, einen konkreten Zahlenwert anzugeben. Wir betrachten das nachfolgende Beispiel für ein leicht operationalisierbares Ziel.

Beispiel 5.3.29 (Operationalisierbares Ziel) *Wir betrachten ein Unternehmen der Hochtechnologiebranche, in dem ein Anwendungssystem fehlt, mit dem der Auftragsstatus verfolgt werden kann. Zur Lösung dieses Problems soll ein MES eingeführt werden. Das Ziel der Einführung eines derartigen Systems besteht darin, den Anteil verspäteter Aufträge zu senken. Mit n_{tardy} wird die Anzahl der Aufträge mit Verspätung bezeichnet. Die Gesamtheit aller Aufträge ist n_{total}. Der Quotient $\frac{n_{tardy}}{n_{total}} * 100\%$ ermöglicht eine quantifizierbare Angabe eines Zieles für die Einführung des MES.*

Wir betrachten nun Ziele, die schwierig zu operationalisieren sind. In dieser Situation ist es nicht mehr möglich, absolute Zahlen auf einer Skala anzugeben. Man behilft sich damit, das zu erreichende Ziel durch Attribute zu charakterisieren und für diese Attribute Ausprägungsstufen einzuführen. Dadurch erfolgt der Übergang von einer kontinuierlichen zu einer diskreten Skala.

Ausprägungsstufen

Wir veranschaulichen die Einführung von Ausprägungsstufen nachfolgend am Beispiel der Flexibilität von PPS-Systemen [20].

Beispiel 5.3.30 (Ausprägungsstufen für eine Zieloperationalisierung) *PPS-Systeme dienen unter anderem dazu, Feinplanungsaktivitäten durchzuführen. Als Ziel wird die Erreichung einer hohen Flexibilität eines PPS-Systems bezüglich der Feinplanung betrachtet. Als Attribut wird die Fähigkeit zu einer nachträglichen Veränderung von Terminen und Mengen betrachtet. In Tabelle 5.2 sind die gewählten Ausprägungsstufen und eine entsprechende Beschreibung dieser dargestellt. Es ist deutlich zu erkennen, dass das Ziel „hohe Flexibilität“ nicht gut geeignet ist, um es quantitativ zu beschreiben.*

Operationalisierung von Einflussgrößen

Die Operationalisierung der Einflussgrößen wird in ähnlicher Art und Weise wie für die Ziele vorgenommen. Falls Einflussgrößen nicht direkt quantitativ operationalisierbar sind, werden wieder die Einflussgrößen zunächst durch Attribute charakterisiert und für die Attribute dann Ausprägungsstufen betrachtet. Wir veranschaulichen dieses Vorgehen im nachfolgenden Beispiel aus der PPS-Domäne [20].

Beispiel 5.3.31 (Operationalisierung einer Einflussgröße) *Wir betrachten dazu das Ziel einer „hohen Planungssicherheit“. Offensichtlich hängt die Erreichung dieses Ziels von der im Unternehmen durchgeführten Termin- und Kapazitätsplanung ab. Die Einflussgröße Termin- und Kapazitätsplanung wird hier beispielhaft durch die beiden Attribute Planungsverfahren und Kapazitätsmodellierung charakterisiert. Diese sind in Tabelle 5.3 dargestellt. Eine detaillierte Darstellung der Termin- und Kapazitätsplanung erfolgt in Abschnitt 6.1.2.*

Es ist wünschenswert, einen funktionalen Zusammenhang zwischen Zielen und Einflussfaktoren herzustellen. In Gleichung (5.2) wird dies für das Ziel j, das von den Einflussfaktoren $E_1, \ldots, E_l$ abhängt, gezeigt. Es gilt:

$$Z_j := f_j(E_1, \ldots, E_l). \tag{5.2}$$

Tabelle 5.2: Ausprägungsstufen für die Flexibilität eines PPS-Systems

Ausprägungsstufe	Charakterisierung
sehr gut	Bis zum geplanten Start der Fertigungsaufträge, durch Lose untersetzt, sind Änderungen von geplanten Fertigstellungsterminen und Mengen uneingeschränkt möglich. Dabei werden alle Folgewirkungen erfasst.
gut	Bis zum Fertigungsstart sind in der Regel Änderungen möglich.
mittelmäßig	Bis zum Fertigungsstart sind Änderungen in einem gewissen Ausmaß möglich.
niedriges Niveau	Nach erfolgter Feinplanung sind Änderungen für Termine und Mengen nur nach einem Neuentwurf des gesamten Ablaufplans möglich.
unzureichend	Nach erfolgter Feinplanung sind keine Änderungen für Termine und Mengen mehr möglich.

Tabelle 5.3: Operationalisierung von Termin- und Kapazitätsplanung

	Attribut	
Ausprägungsstufen	Planungsverfahren	Kapazitätsmodellierung
mittelmäßig	Rückwärtsterminierung	Gegenüberstellung von belegter und verfügbarer Kapazität
gut	zusätzlich Vorwärtsterminierung, manueller Kapaziätsabgleich	zusätzlich Betrachtung von reservierten Kapazitäten
sehr gut	zusätzlich automatischer Kapazitätsabgleich bzw. simultane Kapazitäts- und Terminplanung (Berücksichtigung endlicher Kapazitäten)	zusätzliche Betrachtung von vorgemerkten Kapazitäten

Oft ist es aber nicht möglich, explizit eine Funktion f_j anzugeben. In diesem Fall ist es sinnvoll, Regeln zu formulieren, die den Zusammenhang zwischen Zielen und Einflussfaktoren angeben. Wir betrachten dazu das folgende Beispiel.

Beispiel 5.3.32 (Beispielregeln) *Die nachfolgenden Regeln dienen dazu, zu beschreiben, wie das Ziel „hohe Planungssicherheit“ von den Einflussfaktoren Termin- und Kapazitätsplanung sowie Programmplanung (vergleiche dazu die Ausführungen in Abschnitt 6.1.2.3) abhängt.*

Regel 1:
WENN Termin- und Kapazitätsplanung erfolgt auf hohem Niveau.
UND Programmplanung findet auf hohem Niveau statt.
DANN Planungsicherheit ist hoch.

Regel 2:
WENN Lediglich eine eingeschränkte Termin- und Kapazitätsplanung erfolgt.
UND Programmplanung findet auf hohem Niveau statt.
DANN Eine mittlere Planungssicherheit liegt vor.

Auch hier wird klar, dass eine explizite Angabe eines funktionalen Zusammenhanges nur schwer möglich ist.

Wir sind bisher davon ausgegangen, dass die Einflussfaktoren bekannt sind. Die Ermittlung von Einflussfaktoren in Form von Anforderungen wird in Abschnitt 5.3.7.4 genauer erläutert.

Alternativenvergleich

Im Folgenden nehmen wir an, dass eine bestimmte Menge von betriebswirtschaftlichen Standardsoftwaresystemen vorliegt, von denen das am besten geeignete ausgewählt werden soll. Die Gewinnung einer solchen Menge alternativer Systeme auf Basis der gefundenen Anforderungen wird in Abschnitt 5.3.7.5 behandelt.

Aufwandabschätzung

Eines der Grundprobleme beim **Alternativenvergleich** besteht darin, dass der Aufwand dem erwarteten Nutzen gegenübergestellt wird. Bei betriebswirtschaftlicher Standardsoftware unterscheiden wir zwischen einmaligen und laufenden Aufwänden:

- **Einmalige Aufwände**: Diese Art von Aufwänden tritt in Form von initialen Lizenzgebühren, Schulungsgebühren und Kosten für die Einführungsberatung auf.
- **Laufende Aufwände**: Laufende Aufwände werden typischerweise durch laufende Lizenzgebühren, Wartungskosten und administrative Kosten für den Betrieb der Software verursacht.

Der Aufwand für Alternative j im Jahr k wird mit A_{jk} bezeichnet. Durch Abzinsung auf den aktuellen Zeitpunkt erreicht man eine Vergleichbarkeit der verschiedenen Aufwände. Der Zinssatz wird mit i bezeichnet. Es gilt für den Barwert

der Aufwände von Alternative j:

$$BW(A_j) := \sum_{k=1}^{m} \frac{A_{jk}}{(1+i)^{k-1}}. \tag{5.3}$$

Nutzenbewertung

Nach der Berechnung der diskontierten Aufwände ist es notwendig, den Nutzen der Einführung einer betriebswirtschaftlichen Standardsoftware abzuschätzen. Falls durch die Umstellung der betriebswirtschaftlichen Standardsoftware die Ziele $Z_s, s = 1, \ \ldots, n$ erreicht werden sollen, wird angenommen, dass dabei Ziel Z_s mit einem Gewicht w_s versehen ist. Dabei gilt $\sum_{s=1}^{n} w_s = 1$ und $w_s \geq 0$. Wir nehmen weiterhin an, dass für jedes der Ziele Z_s ein Erfüllungsgrad $a_s, \ 0 \leq a_s \leq 1$ und ein Nutzen $N(Z_s)$ bekannt sind. Der Gesamtnutzen der Alternative j ergibt sich durch:

$$N_j(Z_1, \ldots, Z_n) := \sum_{s=1}^{n} a_s w_s N(Z_s). \tag{5.4}$$

Für die Gesamtbewertung von Alternative j wird dann der Quotient

$$U_j := \frac{N_j}{BW(A_j)} \tag{5.5}$$

betrachtet. Es wird diejenige Alternative ausgewählt, die zum größten Wert für U_j führt. Wir zeigen nun im folgenden Beispiel den Vergleich zweier Alternativen.

Beispiel 5.3.33 (Aufwands- und Nutzenbetrachtung) *Die Aufwände und der Nutzen von zwei PPS-Systemen werden gegenübergestellt. In Tabelle 5.4 und Tabelle 5.5 befinden sich jeweils die Aufwände für Systemalternative 1 und 2. Es wird jeweils der Aufwand für fünf aufeinanderfolgende Jahre angegeben. Insbesondere Kosten für internes Personal, Wartung und Kapitalkosten (Tilgung von Krediten) fallen für diesen Zeitraum an. In den beiden Tabellen werden jeweils zusätzlich die Zahlungsströme, auf das Jahr des Projektbeginns abgezinst, angegeben. Der Zinssatz beträgt 10%. Aus den Ergebnissen in den Tabellen 5.4, 5.5 und 5.6 erhalten wir insgesamt:*

$$U_1 := \frac{3,24 * 100.000 \text{ EUR}}{1.477.600 \text{ EUR}} = 0,219 \tag{5.6}$$

und

$$U_2 := \frac{4,29 * 100.000 \text{ EUR}}{1.991.800 \text{ EUR}} = 0,215. \tag{5.7}$$

Aus Beziehung (5.6) und (5.7) wird deutlich, dass Alternative 1 der Alternative 2 vorzuziehen ist. Dabei ist der Nutzen von Alternative 1 geringer als der von Alternative 2. Alternative 2 verursacht aber größere Aufwände.

Im nächsten Abschnitt beschäftigen wir uns nun mit der Anforderungsspezifikation. Wie bereits dargestellt, können aus Anforderungen Einflussgrößen gewonnen werden.

Tabelle 5.4: Aufwand Alternative 1

	Projekt-begin	Ende 1. Jahr	Ende 2. Jahr	Ende 3. Jahr	Ende 4. Jahr	Ende 5. Jahr
Personal (extern)	150.000	150.000	-	-	-	-
Personal (intern)	-	100.000	100.000	100.000	100.000	100.000
Software	600.000	-	-	-	-	-
Hardware	50.000	-	-	-	-	-
Wartung	-	-	30.000	30.000	30.000	30.000
Kapitalkosten	-	20.000	20.000	20.000	20.000	20.000
Summe (5 Jahre)	**1.670.000**					
abgezinst						
Personal (extern)	150.000	136.400	-	-	-	-
Personal (intern)	-	90.900	82.600	75.100	68.300	62.100
Software	600.000	-	-	-	-	-
Hardware	50.000	-	-	-	-	-
Wartung	-	-	24.800	22.500	20.500	18.600
Kapitalkosten	-	18.200	16.500	15.000	13.700	12.400
Summe (5 Jahre)	**1.477.600**					

5.3.7.4 Anforderungsspezifikation

Anforderungen

Die Auswahlphase für eine Standardsoftware soll möglichst kurz sein, da die zu lösenden Probleme innerhalb des Unternehmens drängen. Daraus folgt, dass die Anforderungsspezifikation nicht zu umfassend und zu detailliert sein darf, um eine Konzentration auf wesentliche Unterschiede zwischen den Anbietern zu ermöglichen [13]. Wir unterscheiden zwischen vier unterschiedlichen Gruppen von Anforderungen:

1. funktionale Anforderungen,
2. technische Anforderungen,
3. Anforderungen an die Benutzerfreundlichkeit,
4. Anforderungen an die Adaptivität der betriebswirtschaftlichen Standardsoftware.

funktionale Anforderungen

Die wichtigsten Anforderungen sind funktionaler Art, da betriebliche Standardsoftwaresysteme zur Lösung konkreter betrieblicher Probleme herangezogen werden sollen. Die **funktionalen Anforderungen** müssen mit dem Funktionsumfang des Standardsystems abgeglichen werden. Es ist festzustellen, wie schwierig

Tabelle 5.5: Aufwand Alternative 2

	Projekt-begin	Ende 1. Jahr	Ende 2. Jahr	Ende 3. Jahr	Ende 4. Jahr	Ende 5. Jahr
Personal (extern)	250.000	250.000	-	-	-	-
Personal (intern)	-	150.000	150.000	150.000	150.000	150.000
Software	600.000	-	-	-	-	-
Hardware	50.000	-	-	-	-	-
Wartung	-	-	50.000	50.000	50.000	50.000
Kapitalkosten	-	40.000	40.000	40000	40.000	40.000
Summe (5 Jahre)	**2.300.000**					
abgezinst						
Personal (extern)	250.000	227.300	-	-	-	-
Personal (intern)	-	136.400	124.000	112.700	102.500	93.100
Software	600.000	-	-	-	-	-
Hardware	50.000	-	-	-	-	-
Wartung	-	-	41.300	37.600	34.200	31.000
Kapitalkosten	-	36.400	33.100	30.100	27.300	24.800
Summe (5 Jahre)	**1.991.800**					

eine Funktionserweiterung durch Modifikation ist. Nicht abdeckbare Funktionalität muss identifiziert werden. Neben der Konzentration auf das Vorhandensein der benötigten Funktionalität muss auch das Laufzeitverhalten der angebotenen Funktionen untersucht werden. Dabei ist das zu bewältigende Mengengerüst zunächst im Unternehmen abzuschätzen und dann in die Bewertung des Laufzeitverhaltens einzubeziehen. Die funktionalen Anforderungen implizieren häufig technische Anforderungen.

technische Anforderungen

Technische Anforderungen dienen der Festlegung der Anwendungssystemeigenschaften und der Beschreibung der Eingliederung der betriebswirtschaftlichen Standardsoftware in die vorhandene bzw. geplante Systemlandschaft des Unternehmens. Die folgenden Anforderungen ergeben sich typischerweise (vergleiche hierzu auch die Ausführungen in Abschnitt 2.5 zu Qualitätsmerkmalen von Informationssystemarchitekturen):

- Plattformunabhängigkeit,
- Konfigurierbarkeit,
- Erweiterbarkeit unter technischen Gesichtspunkten,
- Performanz,

Tabelle 5.6: Nutzen für Alternative 1 und 2 (in 100.000 EUR)

		Alternative 1		Alternative 2	
Umstellungs-ziel	Gewicht	Nutzen	Erfüllungs-grad	Nutzen	Erfüllungs-grad
Planungssicherheit	0,30	8,50	40%	7,20	50%
Termintreue	0,50	6,70	50%	8,00	70%
Flexibilität	0,20	9,00	30%	10,25	20%
Gesamt	-	3,24		4,29	

- Sicherungskonzept,
- Versionierbarkeit von Datenbeständen,
- zur Verfügung stehende Schnittstellentechnologien.

Die Anforderung der Plattformunabhängigkeit ist häufig Wunschdenken. Möglichkeiten zur Erweiterung werden durch die Architektur des Softwaresystems und die zu seiner Realisierung verwendeten Softwaretechnologien entscheidend bestimmt.

Übungsaufgabe 5.3 (Plattformunabhängigkeit) *Worin können Probleme bezüglich der Plattformunabhängigkeit bestehen? Nennen Sie Beispiele.*

Die Performanz ist häufig ein entscheidendes Kriterium dafür, ob eine betriebswirtschaftliche Standardsoftware von den Anwendern akzeptiert wird. Das Sicherungskonzept beinhaltet insbesondere Möglichkeiten zum Backup und zur Archivierung.

Übungsaufgabe 5.4 (Sicherungskonzept) *Diskutieren Sie, welche Kriterien bei einer Detaillierung des Sicherungskonzepts zu berücksichtigen sind.*

Eine Versionierbarkeit von Datenbeständen spielt in Dokumentenverwaltungs- sowie in Data-Warehouse-Systemen eine große Rolle. Die zur Verfügung stehenden Schnittstellentechnologien bestimmen, wie die betriebswirtschaftliche Standardsoftware mit anderen Anwendungssystemen gekoppelt werden kann (vergleiche dazu die Ausführungen in Abschnitt 3.5 der Kurseinheit 3 über die Konstruktion von betrieblichen Informationssystemen).

Benutzerfreundlichkeit

Bei der **Benutzerfreundlichkeit** steht die Bedienung des betriebswirtschaftlichen Standardsoftwaresystems mit einem Minimum an Schulungsaufwand im Mittelpunkt des Interesses. Als Anforderungen ergeben sich den Aufgaben angemessene Dialoge, die den Benutzer bei der Erfüllung seiner Aufgaben unterstützen und nicht behindern. Die Dialoge sollen weitestgehend selbsterklärend sein. Die graphische Benutzeroberfläche mit den zugehörigen Dialogen

soll erwartungskonform sein. Die Dialoge haben robust zu sein, d.h., sie sind in der Lage, mit fehlerhaften Eingaben umzugehen. Die graphischen Benutzeroberflächen sind personalisierbar, d.h., bei der Formulargestaltung sowie beim Design von Masken können individuelle Wünsche der Anwender berücksichtigt werden. Die betriebswirtschaftliche Standardsoftware soll Möglichkeiten zur Lernunterstützung bereitstellen. Wir betrachten dazu das folgende Beispiel.

Beispiel 5.3.34 (Lernunterstützung) *Eine Unterstützung der Anwender kann in Form von speziellen Hilfesystemen für die Software erfolgen. Im SAP R/3-System erhält man durch Drücken der F1-Taste Unterstützung für die aktuell aufgerufene Bildschirmmaske.*

adaptive Anforderungen

Nach den funktionalen, technischen und Bedienbarkeitsanforderungen sind insbesondere **adaptive Anforderungen** von entscheidender Bedeutung. Adaptive Anforderungen haben die Fähigkeit des Standardsoftwaresystems, sich an veränderte organisatorische Bedingungen im Unternehmen anzupassen, zum Gegenstand. Wir erinnern an die in Abschnitt 2.3.1.2 beschriebenen Modellebenen von SOM:

1. **Außensicht des betrieblichen Systems**: Festlegung von Unternehmenszielen,
2. **Innensicht des betrieblichen Systems**: Geschäftsprozesse,
3. **Spezifikation von Ressourcen**.

Veränderungen in der 1. und 2. Modellebene führen zu ständigen Veränderungen in den Anwendungssystemen. Diese Veränderungen sind die Ursache dafür, dass Anforderungen an das adaptive Verhalten eines Informationssystems gestellt werden.

strukturelle Analogie

Sinnvoll ist in diesem Zusammenhang die Herstellung einer strukturellen Analogie zwischen Organisation und Informationssystemarchitektur, da durch diese Isomorphie die als Folge veränderter Unternehmensziele und Geschäftsprozesse geänderte Organisation direkten Eingang in die Informationssysteme finden kann [13]. In Tabelle 5.7 geben wir verschiedene konkrete Ausprägungen der strukturellen Analogie an.

Die nachfolgenden Gründe sind für die Anwendung der strukturellen Analogie bei der Gestaltung von betrieblichen Informationssystemen verantwortlich [13]:

- Es liegen ähnliche Muster vor. Die Abbildung von Geschäftsprozessen und organisatorischen Strukturen in Informationssystemen wird bei Vorliegen einer strukturellen Analogie einfacher. Es ergeben sich Vorteile bei der Spezifikation und Wartung betrieblicher Informationssysteme, da Anforderungen aus der organisatorischen Realität abgeleitet werden können.

Tabelle 5.7: Ausprägungen der strukturellen Analogie

Organisation	IT-Architektur
Segmentierung der Organisation	Objektorientierung als Programmierparadigma
organisatorische Ablauforientierung über Abteilungsgrenzen hinweg	Workflowmanagement über Anwendungssystemgrenzen hinweg
kontinuierliche Reorganisation und entsprechende Lernvorgänge	kontinuierlicher Releasewechsel bei Standardsoftwaresystemen, als Ergebnis erhält man eine verbesserte Funktionalität und geringere Fehleranfälligkeit
Auflösung von Unternehmensgrenzen	Nutzung verteilter Anwendungen im Netzwerk über Middlewaretechnologien bzw. Webservices und die damit verbundenen Kommunikations- und Synchronisationsmechanismen (vergleiche dazu Abschnitt 4.2.1 und 4.3)

- Die Strukturähnlichkeit von Organisation und Informationssystemarchitektur erleichtert die Einführung und Nutzung von Informationssystemen, da die Informationssysteme auf Grund der Analogien zu bereits existierenden organisatorischen Strukturen besser und schneller verstanden werden können.
- Die Abbildung von rechnerunterstützten Informationsflüssen entlang den von der Organisation vorgegebenen Informationswegen vermeidet unnötige, zu lange oder redundante Informationsflüsse.

Sammeln von Anforderungen

Abschließend wird das Vorgehen bei der Aufstellung von Anforderungsspezifikationen beschrieben [13]. Die drei Arbeitsschritte:

1. Sammeln,
2. Bewerten,
3. Verdichten

der Anforderungen sind dafür charakteristisch. Das Sammeln von Anforderungen beinhaltet die Durchführung von Interviews mit den prozessbeteiligten Mitarbeitern und dem Management zum Kennenlernen von Wünschen und Schwachstellen bisheriger IT-Lösungen.

Bewertung von Anforderungen

Eine Bewertung der Anforderungen erfolgt durch eine Katalogisierung der Anforderungen sowie durch eine Verabschiedung der Anforderungen durch die

Unternehmensleitung. Als Ergebnis erhalten wir einen verabschiedeten Anforderungskatalog.

Im nächsten Schritt wird der Anforderungskatalog verdichtet. Ein verdichteter Katalog hat den Vorteil, dass er durch den Anbieter der Standardsoftware leichter überprüft werden kann. Der verdichtete Anforderungskatalog soll verbindlicher Vertragsbestandteil sein. Der Anbieter der Standardsoftware hat dann keine Möglichkeit zum Rückzug auf die „normalerweise“ zur Verfügung stehende Funktionalität.

Verdichten von Anforderungen

5.3.7.5 Alternativen- und Anbieterauswahl

Eine Vorauswahl von Anbietern dient dazu, potentielle Anbieter von betriebswirtschaftlicher Standardsoftware zu identifizieren. Die Vorauswahl kann durch

Anbieterauswahl

- das Studium von Fachzeitschriften und Büchern,
- den Besuch von Messen wie zum Beispiel der IT2INDUSTRY (http://www.it2industry.de),
- Recherchen im Internet auf Seiten wie www.softguide.de, www.management-software.de, www.it-matchmaker.com,
- Verwendung von Übersichten von Dienstleistern erfolgen.

Das nachfolgende Beispiel verdeutlicht Möglichkeiten der Unterstützung des Auswahlprozesses durch ein Literaturstudium.

Beispiel 5.3.35 (Unterstützung der Auswahl durch Literatur) *Der in der Monographie [8] beschriebene Produktvergleich von ERP-Systemen oder die jährlich erscheinende ERP-Studie des Konradin Verlages können zur Auswahl herangezogen werden. In der Zeitschrift ERP Management erscheinen ebenfalls regelmäßig Produktvergleiche.*

Wir betrachten das folgende Beispiel für die Unterstützung der Vorauswahl von Anbietern durch spezialisierte Dienstleister.

Beispiel 5.3.36 (Unterstützung der Auswahl durch Dienstleister) *Das Forschungsinstitut für Rationalisierung (FIR) e.V. gibt Hilfestellungen bei der Auswahl von PPS-Systemen. Die Trovarit AG stellt eine strukturierte Vorgehensweise für Auswahlprojekte zur Verfügung.*

Nach der Vorauswahl wird eine Anbieterbefragung durchgeführt. Das Ziel der Anbieterbefragung besteht darin, aus der Gruppe vorausgewählter Anbieter eine verbleibende Restgruppe von ca. drei Systemen auszuwählen. Die Anbieterbefragung ist als schriftliche Befragung durchzuführen, einmal getätigte Antworten sind bei Vertragsabschluss verbindlich. Der Umfang der Befragung ist in Grenzen zu halten. Die Fragen sind so zu gestalten, dass die Antworten der unterschiedlichen Anbieter vergleichbar sind. Eine angemessene Zeitspanne ist für die

Anbieterbefragung

Antworten vorzugeben.

Die Anbieterbefragung besteht aus Informationen zum Unternehmen, das die betriebswirtschaftliche Standardsoftware einsetzen will, sowie aus einer Reihe von Fragen an den Softwareanbieter. Das auswählende Unternehmen liefert:

- eine Kurzbeschreibung des Unternehmens,
- eine knappe Beschreibung der wichtigsten Geschäftsprozesse, die im Zusammenhang mit der auszuwählenden beriebswirtschaftlichen Standardsoftware stehen.

Der Anbieter der Standardsoftware beantwortet die nachfolgenden Fragen:

- Entwicklungsstand der angebotenen Software,
- Informationen über die Architektur des betriebswirtschaftlichen Standardsoftwaresystems,
- Informationen bezüglich Serviceleistungen, Kosten und Leistungsumfang für einen Wartungs- und Servicevertrag,
- Datenbankkonzept und systemübergreifende Funktionalität,
- Informationen zu den wichtigsten spezifizierten Anforderungen, untergliedert nach Stammdaten, Prozessen und Querschnittsfunktionen.

Übungsaufgabe 5.5 (Anbieterbefragung) *Konkretisieren Sie den Fragenkatalog bezüglich der Systemarchitektur des betriebswirtschaftlichen Standardsoftwaresystems.*

Richtangebot

Als Ergebnis der Anbieterbefragung erhält das Unternehmen ein Richtangebot der Anbieter, das sich aus

- Lizenzkosten in Abhängigkeit von der Anzahl der Nutzer und Server,
- Kosten für weitere eventuell erforderliche Software,
- geschätzte Kosten für Schulung, Beratung, Customizing und Installation

zusammensetzt. Bei der Entscheidung für den Kreis der Anbieter, die für die Endauswahl in Frage kommen, ist bei ansonsten gleichen Bewertungen eine Betrachtung der räumlichen Nähe durchzuführen, bei ansonsten gleichem Funktionsumfang das System mit niedrigsten Anschaffungs- und Wartungskosten auszuwählen und die Vorliebe für bestimmte Hardware-/Betriebssystemkombinationen zu berücksichtigen.

Anbieterpräsentation

Unmittelbar an die Anbieterbefragung schließt sich als nächster Schritt eine **Anbieterpräsentation** an [13]. Die Grundidee der Anbieterpräsentation

besteht darin, dass der Standardsoftwareanbieter mit den spezifischen Stammdaten und Prozessen des jeweiligen Unternehmens konfrontiert wird. Es wird untersucht, ob diese Daten und Prozesse mit der konkreten Standardsoftware abgebildet werden können. Falls keine spezifischen Stammdaten verwendet werden können, präsentieren die Vertriebsmitarbeiter des Softwareanbieters häufig nur Standardfolien. Es ist wichtig, alle von der Systemeinführung betroffenen Unternehmensbereiche in die Anbieterpräsentation einzubeziehen.

Übungsaufgabe 5.6 (Anbieterpräsentation) *Warum ist es notwendig, dass sowohl die DV-Verantwortlichen als auch die Fachabteilungen bei der Anbieterpräsentation anwesend sind?*

Die Qualität des Standardsoftwareanbieters wird unter den folgenden Gesichtspunkten bewertet [13]:

- Erreichbarkeit und Qualität der Anwenderhotline,
- Anzahl neuer Releases pro Jahr und jeweiliger Umstellungsaufwand,
- Qualität der mitgelieferten Dokumentation,
- Kompetenz des Beratungspersonals,
- Qualität der vom Anbieter durchgeführten Schulungen,
- Effizienz der Softwareeinführung bei Referenzkunden unter anderem unter dem Gesichtspunkt der Einhaltung des geplanten Einführungstermins,
- Erfahrungen mit Rechnungslegung und Kulanz des Anbieters,
- Verlässlichkeit bei der Erfüllung nicht exakt spezifizierter Aufgaben wie mündlich oder telefonisch getroffene Zusagen,
- Bereitschaft des Anbieters zu Kompromissen im Streitfall.

Vertragsgestaltung

Nach Abschluss der Bewertungsarbeit entscheidet sich das Unternehmen für genau einen Anbieter betriebswirtschaftlicher Standardsoftware und nimmt **Vertragsverhandlungen** mit diesem Anbieter auf. Ziel der Verhandlungen ist der Abschluss eines Vertrags über ein Standardsoftwareeinführungsprojekt. Der Vertragsbegriff ist in diesem Kontext wie folgt definiert.

Definition 5.3.12 (Vertrag Standardsoftwareeinführungsprojekt) *Der Vertrag ist ein juristisches Abbild des Projektgeschehens für die Einführung der betriebswirtschaftlichen Standardsoftware. Gegenwärtiges und zukünftiges Verhalten beider Vertragsparteien und die korrekte Strukturierung und Abbildung der aktuellen Willensübereinkunft werden geregelt.*

Die allgemeinen Vertragsbedingungen der Softwareanbieter sind häufig nur eingeschränkt nutzbar, es muss auch die konkrete betriebliche Situation im Vertrag abgebildet werden. Dazu ist in den meisten Fällen ein spezifischer Projektvertrag notwendig. Ein derartiger Vertrag enthält vier Gruppen vertraglicher Regelungen:

1. Leistungsbeschreibung und alle hier einzuordnenden Regelungen,
2. Vergütung,
3. Organisations- und Verfahrensregeln während der Laufzeit des Projektes,
4. rechtliche Regeln, die die Behandlung von Situationen außerhalb des Projektes bzw. des korrekten Projektablaufs enthalten.

Die Leistungsbeschreibung enthält typischerweise eine Vereinbarung über die Erstellung einer Projektstudie, eines Pflichtenheftes oder einer Anforderungsspezifikation.

Bei der Vergütung sind die Unternehmen, welche die Standardsoftware einführen wollen, an einem Festpreisangebot interessiert, während die Hersteller der Standardsoftware an einer Vergütung nach Aufwand interessiert sind. In der Praxis haben sich Mischformen dieser beiden extremen Ausprägungen durchgesetzt. Im ersten Schritt wird dazu ein Festpreis festgelegt. Falls der Softwareanbieter dieses Budget überzieht, wird für bestimmte Leistungen nach Aufwand abgerechnet. In jüngerer Zeit haben aber auch verschiedene Rabattmodelle eine gewisse Bedeutung erlangt.

Unter den festzulegenden **Organisationsregeln** versteht man die Festlegung eines Zeitplans, die Definition von Meilensteinen, die Benennung verantwortlicher Personen für Auftragnehmer und Auftraggeber, Festlegung bezüglich einer zu lebenden Meetingkultur sowie eine Festschreibung der Art und Weise der Projektdokumentation. Regeln für die Abnahme des Einführungsprojektes werden als **Verfahrensregeln** bezeichnet.

Rechtliche Regeln zur Behandlung von Situationen außerhalb des Projektes enthalten in Anlehnung an [13] unter anderem Vereinbarungen über

- die Fehlerbeseitigung innerhalb definierter Reaktionszeiten,
- die Auslieferung neuer Programmstände bei veränderten Rahmenbedingungen,
- die Hotline zur Bedienung der Programme und für Unterstützung bei auftretenden Fehlern,
- weitere Serviceleistungen wie geschützte Downloadbereiche, in denen Informationen zu bekannten Fehlern und zur Lösung bestimmter Problemstellungen vorgehalten werden,

- Festlegungen zu Gewährleistungspflichten.

Im nächsten Abschnitt werden wir genauer auf unterschiedliche Möglichkeiten zur Lizenzierung und dem Betrieb betriebswirtschaftlicher Standardsoftware eingehen.

5.3.7.6 IT-Liefer- und Betreibermodelle

IT-Liefermodelle

IT-Liefermodelle beschäftigen sich mit der Frage, wie eine betriebswirtschaftliche Standardsoftware den Anwendern zur Verfügung gestellt wird.

In [25] wird dazu zunächst der Begriff der **traditionellen betriebswirtschaftlichen Standardsoftware** eingeführt. Darunter versteht man Software, die von einem Softwarehersteller für eine Gruppe von Kunden mit ähnlichen Anforderungen geschrieben worden ist. Das Unternehmen, in dem die betriebswirtschaftliche Software eingesetzt werden soll, erwirbt eine Lizenz und installiert und betreibt die Software anschließend eigenständig (in Abschnitt 2.3.3 hatten wir in einem ähnlichen Zusammenhang von konventionellen Informationssystemen gesprochen). Die nachfolgenden Lizenzmodelle sind möglich, deren Lizenzkosten abhängig sind von:

- Leistungsumfang der betriebswirtschaftlichen Standardsoftware,
- der Anzahl der Arbeitsplätze, an denen die Software zur Verfügung steht,
- der Leistungsfähigkeit des Servers, auf dem die Software installiert ist.

Es kommen auch pauschalisierte Lizenzen für Organiationseinheiten zum Einsatz.

ASP

In den letzten Jahren wurde verstärkt versucht, durch Outsourcing des Betriebs von Anwendungssystemen Kosten zu sparen. Eine spezielle Möglichkeit zur Gestaltung von Outsourcing-Bemühungen ist das Application-Service-Providing (ASP). Wir definieren diesen Begriff wie folgt [25].

Definition 5.3.13 (ASP) *Ein Dienstleistungskonzept, bei dem Application-Service-Provider Anwendungssysteme auf einem zentralen Server bereitstellen, wird als ASP bezeichnet.*

Die durch den Benutzer zu verwendenden Präsentations- und Nutzdaten werden vom Server zu einem Client transportiert. Die gesamte Verarbeitung der Daten erfolgt auf dem Server. Auf die durch das ASP bereitgestellten Anwendungssysteme kann über Webbrowser oder spezielle graphische Benutzeroberflächen zugegriffen werden. Für die nachfolgenden Aufgaben ist u.a. der Application-Service-Provider zuständig:

- Durchführung der Wartung der Anwendungssysteme,
- Einspielen von Updates für die Anwendungssysteme,

- Organisation der Benutzerverwaltung,
- Sicherstellung der Sicherheit für das Anwendungssystem,
- Durchführung von Datensicherungen in regelmäßigen Abständen.

Der Application-Service-Provider verlangt dafür von den Anwendern der Software eine nutzungsabhängige Gebühr. Man spricht in diesem Zusammenhang von einer One-to-many-Strategie. Wir betrachten das nachfolgende Beispiel.

Beispiel 5.3.37 (ASP) *Das im Lehrbetrieb eingesetzte mySAP ERP-System wird im Hochschulkompetenzzentrum (HCC) der SAP AG an der Technischen Universität Magdeburg betrieben. Die Studierenden und Lehrkräfte der FernUniversität greifen darauf über die SAP GUI (vergleiche dazu auch Beispiel 2.6.5) zu.*

SaaS

Wir bemerken, dass in den letzten Jahren die Bezeichnung ASP zunehmend durch Software-as-a-Service (SaaS) ersetzt wird.

Übungsaufgabe 5.7 (Application-Service-Providing) *Erläutern Sie die Unterschiede zwischen externem und internem Application-Service-Providing.*

5.3.8 Einführung betriebswirtschaftlicher Standardsoftware

Nachdem in Abschnitt 5.3.7 dargestellt wurde, wie bei der Auswahl einer betriebswirtschaftlichen Standardsoftware vorzugehen ist, wird in diesem Abschnitt die Einführung eines derartigen Softwaresystems in einem Unternehmen beschrieben.

5.3.8.1 Phasenmodell für die Einführung

Einführungsphase

Wir nennen zunächst die Phasen der Einführung von betriebswirtschaftlicher Standardsoftware in einem Unternehmen in Anlehnung an [13]. Dabei wird stets vorausgesetzt, dass eine konkrete Standardsoftware bereits ausgewählt wurde.

1. **Projektorganisation überprüfen**: grundsätzliche Entscheidungen bezüglich der Art und Weise der Einführung, eigene Projektstruktur festlegen und externe Berater und Softwareanbieter einbeziehen,
2. **Feinspezifikation (Workshopphase)**: Gestaltung von Lösungen für die zu unterstützenden Geschäftsprozesse,
3. **Prototypphase**: Installation des Systems beim Anwender entsprechend der Ergebnisse der Feinspezifikationsphase und Test der vorgenommenen Einstellungen,

4. **Probebetrieb (Pilotbetrieb)**: Integration von Realdaten,

5. **Produktivbetrieb**: Mitarbeiter des Unternehmens arbeiten mit dem System auf Realdaten.

Die unterschiedlichen Phasen sind in Abbildung 5.7 dargestellt. Die Abbildung gibt Auskunft darüber, wer in den einzelnen Phasen mit dem Standardsoftwaresystem arbeitet und welche Daten dabei verwendet werden.

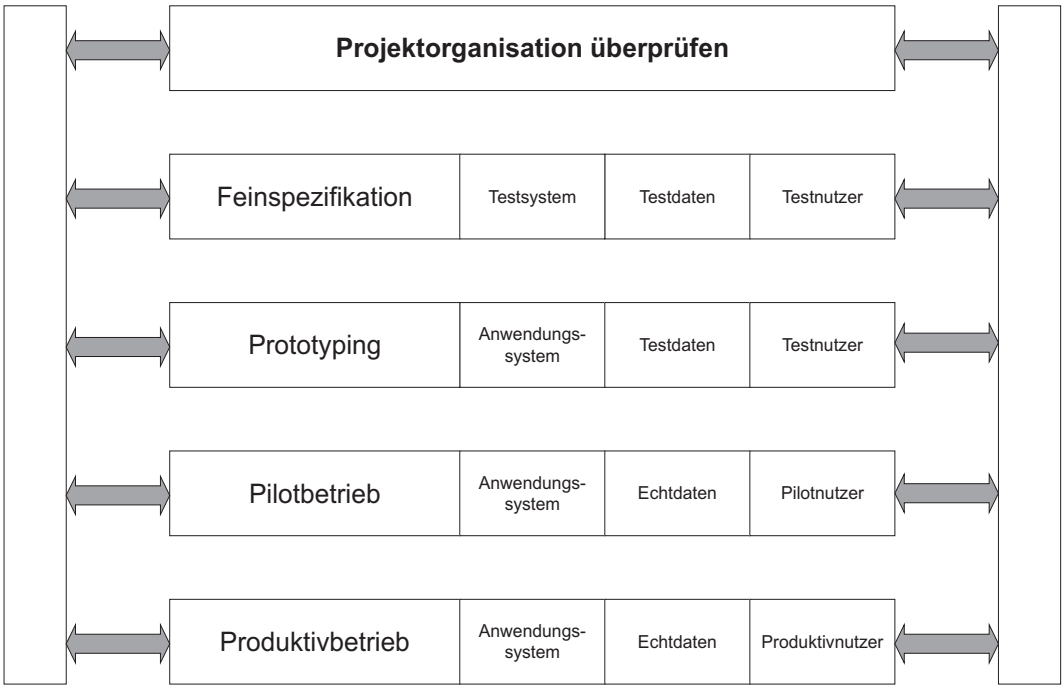

Abbildung 5.7: Phasen des Einführungsprozesses [13]

Als Beispiel für ein produktbezogenes Phasenmodell kann die Implementation-Roadmap als Bestandteil des Accelerated SAP (ASAP)-Vorgehensmodells [1] dienen. ASAP

Beispiel 5.3.38 (Implementation-Roadmap ASAP) *ASAP ist ein Hypertextsystem, das verschiedene Startpunkte für die Navigation enthält. ASAP ermöglicht im Wesentlichen bereits eine Unterstützung der Einführung eines SAP R/3-Systems bevor die Software überhaupt installiert wurde. Dadurch kann die Einführung von SAP R/3-Systemen im Unternehmen teilweise beträchtlich beschleunigt werden. Wesentlich sind drei als Roadmaps bezeichnete Vorgehensmodelle:*

1. *Einführung (Implementation-Roadmap),*

2. *Kontinuierliche Verbesserung des Anwendungssystems (stetige Optimierung),*
3. *Upgrades (Upgrade-Roadmaps).*

Die Implementation-Roadmap umfasst die folgenden fünf aufeinanderfolgenden Phasen:

1. *Projektvorbereitung,*
2. *Business-Blueprint,*
3. *Realisierung,*
4. *Produktionsvorbereitung,*
5. *Go-Live-and-Support.*

ASAP sieht ein sequentielles Durchlaufen der einzelnen Phasen vor. Eine einzelne Phase besteht aus mehreren Arbeitspaketen.

Es ist zu erkennen, dass die Phasen der Implementation-Roadmap mit wenigen Ausnahmen in die Phasen des allgemeinen Modells zur Einführung eingeordnet werden können. Die einzelnen Phasen des allgemeinen Einführungsvorgehensmodells werden im weiteren Verlauf dieser Kurseinheit im Detail diskutiert. An vielen Stellen wird auch ausgeführt, wie eine Umsetzung im Rahmen von ASAP erfolgt.

Übungsaufgabe 5.8 (Implementation-Roadmap) *Ordnen Sie die Phasen der Implementation-Roadmap der SAP AG den Phasen des allgemeinen Modells zu. Überlegen sie, warum die Phasen nicht deckungsgleich sind und die SAP AG ein eigenes Phasenmodell entwickelt hat.*

5.3.8.2 Überprüfung der Projektorganisation

Projektorganisation überprüfen

Diese Phase dient der Planung und Vorbereitung des gesamten Einführungsprojektes. Im ersten Schritt wird ein Projektplan erstellt. Anschließend muss ein Projektmanagement etabliert werden. Das dafür erforderliche Vorgehen wurde bereits in Abschnitt 5.3.7.2 dieser Kurseinheit diskutiert. Es ist notwendig, die Projektabläufe festzulegen. Ein weiterer wesentlicher Schritt ist die Festlegung einer **Einführungsstrategie**.

Einführungsstrategie

Wir unterscheiden zwischen einer **Big-Bang-** und einer **inkrementellen** Vorgehensweise. Bei einer Big-Bang-Strategie wird das Altsystem in einem Schritt durch das neue betriebswirtschaftliche Standardsoftwaresystem abgelöst. Von einem lokalen Big-Bang wird gesprochen, wenn in dezentralen Organisationen zunächst die Einführung der Software in einer Organisationseinheit vorgenommen wird und anschließend ein lokales Roll-out in den anderen Organisationseinheiten erfolgt.

Bei einer inkrementellen Vorgehensweise kann eine funktions- bzw. prozessorientierte Sichtweise eingenommen werden. Die funktionsorientierte Sicht orientiert sich an der vorhandenen Aufbauorganisation im Unternehmen. Wir veranschaulichen diese Sichtweise im nachfolgenden Beispiel.

Beispiel 5.3.39 (Funktionsorientierte Vorgehensweise) *Es ist möglich, zunächst lediglich das vorhandene Personalwesen durch die einzuführende betriebswirtschaftliche Standardsoftware zu unterstützen.*

Bei einer prozessorientierten Vorgehensweise wird hingegen die Ablauforganisation zugrundegelegt. Es erfolgt eine schrittweise Umstellung ganzer Geschäftsprozesse. Wir betrachten dazu das folgende Beispiel.

Beispiel 5.3.40 (Prozessorientierte Vorgehensweise) *Die Einführung einer betriebswirtschaftlichen Standardsoftware kann mit der Unterstützung der Auftragsabwicklung beginnen.*

Definition der Systemlandschaft

Gegenstand der Einführungsstrategie ist ebenfalls die Definition der Systemlandschaft für die einzuführende betriebswirtschaftliche Standardsoftware. Es muss festgelegt werden, wie viele Installationen der betriebswirtschaftlichen Standardsoftware zu welchem Zweck installiert werden sollen. Typischerweise werden mindestens die folgenden Installationen empfohlen:

- Entwicklungssystem,
- Qualitätssicherungs- und Testsystem,
- Produktivsystem.

Auf dem Entwicklungssystem wird das Customizing des Standardsoftwaresystems vorab vorgenommen, da oft die Nebenwirkungen von bestimmten Einstellungen nicht vorhersehbar sind. Das Entwicklungssystem arbeitet entweder auf bestimmten Ausschnitten der im Produktivsystem vorhandenen Stamm- und Bewegungsdaten oder verwendet diese Daten in ihrer Gesamtheit. Entwicklungsarbeiten für bestimmte in der Standardsoftware zu integrierende Lösungsverfahren werden häufig an Testinstanzen geringerer Komplexität vorgenommen, um Zeit zu sparen. Das Qualitätssicherungs- und Testsystem verwendet die Realdaten des Unternehmens und gleicht weitestgehend dem Produktivsystem in Bezug auf abgebildete Organisationseinheiten und eingestellte Geschäftsprozessparameter. Das Qualitätssicherungs- und Testsystem dient dem Test der im Entwicklungssystem vorgenommenen Einstellungen und Erweiterungen des Standardsoftwaresystems. Das Produktivsystem hingegen arbeitet auf den Realdaten und wird von den Mitarbeitern des Unternehmens permanent benutzt. Falls größere Änderungen unter Verwendung der Entwicklungsumgebung vorgenommen werden müssen, ist es sinnvoll, neben dem Entwicklungssystem ein eigenes Customizingsystem zu betreiben. Wir zeigen im folgenden Beispiel, welche Systemlandschaft für SAP R/3 sinnvoll ist.

Beispiel 5.3.41 (Gestaltung der SAP R/3-Systemlandschaft) *Die SAP AG empfiehlt, mindestens drei separate SAP R/3-Systeme einzusetzen, nämlich ein Entwicklungs-, ein Test- und Qualitätssicherungs- sowie ein Produktivsystem. Ein SAP R/3-System besitzt eine eigene System-ID und kann mehrere Mandanten umfassen.*

Abschließend beinhaltet die diskutierte Phase noch eine grobe Planung der technischen Anforderungen bezüglich Hardware und Kommunikationsinfrastruktur für den Betrieb der betriebswirtschaftlichen Standardsoftware sowie ein Projekt-Kickoff.

In der ASAP-Terminologie werden in dieser Phase Arbeiten der Projektvorbereitung durchgeführt. Wir stellen im folgenden Beispiel noch dar, welche speziellen Tätigkeiten im Rahmen von ASAP ausgeführt werden müssen.

Beispiel 5.3.42 (Phase der Projektvorbereitung in ASAP) *Im Rahmen der Projektvorbereitung schlägt ASAP nach der Klärung der unternehmensspezifischen Einführungsstrategie vor, ein Umfangsdokument zu erstellen, das entsprechend des R/3-Geschäftsprozessmodells Szenarien und die zugehörigen Prozesse beinhaltet. Für jeden Prozess werden der zu implementierende Umfang, Scope genannt, ein unternehmensinterner Prozessverantwortlicher sowie ein externer Berater festgelegt. Das Umfangsdokument ist das Ergebnis von Diskussionen zwischen Projektleitung sowie den von der Umstellung betroffenen Geschäftsprozessteams. Der ermittelte Projektumfang wird in einer Datenbank abgelegt. Diese Datenbank wird als Question&Answer (Q&A)-Datenbank bezeichnet.*

5.3.8.3 Feinspezifikation

Feinspezifikation

Die Aufgaben dieser Phase bestehen in einem Abgleich zwischen den Parametern der einzuführenden Software und der organisatorischen Abläufe, so dass ein effizienter Produktivbetrieb aufgenommen werden kann [13]. Im Einzelnen sind die nachfolgenden Aufgaben zu lösen:

1. Abbildung der Organisationsstruktur im System,
2. Einstellung der Geschäftsprozessparameter,
3. (Vorab)-Prototyping,
4. iterative Wiederholung dieser Schritte.

Die Feinspezifikationsphase wird auch als **Workshopphase** bezeichnet, weil gemeinsame Workshops zwischen IT-Abteilung und Fachabteilungen stattfinden, in denen es um die Abbildung der Organisationsstruktur und die Einstellung der Geschäftsprozessparameter geht.

Im ersten Schritt muss die im jeweiligen Unternehmen vorhandene

Organisationsstruktur

Organisationsstruktur im Standardsoftwaresystem abgebildet werden. Dazu muss eine Zuordnung zwischen den im Unternehmen und den im Softwaresystem verwendeten Begriffen vorgenommen werden. Wir betrachten dazu ein Beispiel für das betriebswirtschaftliche Standardsoftwaresystem SAP R/3 bzw. mySAP ERP.

Beispiel 5.3.43 (Organisationsstruktur und SAP-Software) *In Tabelle 5.8 wird eine Zuordnung von den im Unternehmen zu den im Softwaresystem verwendeten Begriffen exemplarisch gezeigt. Buchungskreise ermöglichen die Erstellung eines Einzelabschlusses mit Bilanz sowie Gewinn- und Verlustrechnung (vergleiche hierzu die Ausführungen in 5.3.6). Sie dienen der Abbildung von natürlichen Unternehmen oder Betriebsstätten eines Konzerns. Es wird deutlich, dass die Geschäftsbereiche orthogonal zu den Buchungskreisen sind, d.h., ein einzelner Buchungskreis kann mehrere Geschäftsbereiche umfassen und umgekehrt.*

Übungsaufgabe 5.9 (Organisationsstruktur und SAP-Software) *Sie haben die Aufgabe, in einem Rechenzentrum, das Rechenleistungen im Bereich Finanzbuchhaltung mit dem System SAP R/3 anbietet, die SAP R/3-Organisationseinheiten für nachfolgende Kunden festzulegen.*
Zur Kunststoff AG gehören zwei selbständige Firmen: Kunststoff-Recycling GmbH NRW und Kunststoffteile-Gesellschaft Frankreich. Eine Konzernabrechnung ist vorgesehen.
Die Kunststoff-Recycling GmbH NRW hat drei rechtlich nicht selbständige Werke: Werk Holzwickede, Werk Bottrop und Werk Unna. Für diese soll jeweils eine interne Bilanz erstellt werden.
Das Transportunternehmen „Stop and Go“ besitzt zwei Filialen (Filiale Sprockhövel und Filiale Castrop-Rauxel), für die eine gemeinsame Gewinn- und Verlustrechnung gefordert ist.
Wählen Sie für obige Aufgabenstellung minimale Strukturen zur Abbildung in SAP R/3. Stellen Sie die Organisationseinheiten in Form von Organigrammen, bestehend aus SAP R/3-Organisationstrukturen, dar.

Einstellung von Geschäftsprozessparametern

Die folgenden Geschäftsprozessparameter sind in der betriebswirtschaftlichen Standardsoftware einzustellen:

- Währungen, Betriebskalender mit Feiertagen, länderspezifische Einstellungen,
- Festlegung der von den einzelnen Unternehmensbereichen genutzten oder geplanten Nummernkreise für Artikel, Maschinen, Personen, Kunden, Lieferanten, Organisationseinheiten, Kostenstellen, -arten, -trägern, Konten, Belegen,
- Festlegungen zum Aufbau und zur Pflege von Stammdatenstrukturen,

Tabelle 5.8: Abbildung von Organisationsstrukturen in SAP R/3

Unternehmensorganisationseinheit	SAP R/3 bzw. mySAP ERP
Konzern	**Mandant**
kleinste Organisationseinheit des externen Rechnungswesens, für die eine vollständige, in sich abgeschlossene Buchhaltung abgebildet werden kann	**Buchungskreis**
aus Markt- und Produktsicht notwendige Organisationseinheiten: Divisionen, Sparten, Geschäftsfelder	**Geschäftsbereich**: buchungskreisübergreifende, konzerninterne Strukturierung zur externen, segmentbezogenen Berichterstattung
internes Rechnungswesen: Kostenrechnung, Organisationseinheit, für die eine eigene, in sich abgeschlossene Kostenrechnung durchgeführt wird	**Kostenrechnungskreis**
Logistiksicht: Organisationseinheiten des Unternehmens sowie Produktgruppen	**Werke**: sind genau einem Buchungskreis zugeordnet, **Sparten**: dienen der Zuordnung von Materialien zu Geschäftsbereichen
Materialwirtschaftssicht: Lager sowie der Organisationseinheitstyp, der zur Beschaffung von Materialien dient	**Lager** und **Einkaufsorganisation**
Vertriebssicht: Organisationseinheitstyp zum Vertrieb bestimmter Produkte und Dienstleistungen eines Unternehmens	**Verkaufsorganisation**

- Eingabe der im System verwendeten Maßeinheiten und der Umrechnungsvorschriften zwischen den Maßeinheiten,
- Festlegung der einzusetzenden Kontenrahmen für Kostenrechnung, Controlling und Buchhaltung,
- Zuordnung der Konten zu den Positionen der Bilanz bzw. der Gewinn- und Verlustrechnung,
- Festlegung von Schnittstellenspezifikationen zu anderen Anwendungssystemen und organisatorische Einbettung dieser Schnittstellen.

Schnittstellen

Wir betrachten noch das nachfolgende Beispiel für die organisatorische Einbettung von Schnittstellen.

Beispiel 5.3.44 (Schnittstellen) *Festlegungen, wer Rechnungsdaten in das Controllingsystem überträgt, dienen der organisatorischen Einbettung der Schnittstellen zum Controllingsystem.*

Teile der Feinspezifikationsphase entsprechen der Business-Blueprint-Phase der Implementation-Roadmap von ASAP. Aus diesem Grund stellen wir an dieser Stelle zusätzlich die Business-Blueprint-Phase in Beispiel 5.3.45 dar. Wir definieren vorab noch den Begriff eines Business-Blueprints in Anlehnung an [1].

Business-Blueprint

Definition 5.3.14 (Business-Blueprint) *Ein Entwurf und Plan, der eine abstrakte Beschreibung des zukünftigen SAP R/3-Systems beinhaltet, wird als Business-Blueprint bezeichnet.*

Beispiel 5.3.45 (Business-Blueprint-Phase von ASAP) *Das Erstellen eines Business-Blueprints wird in mehreren Arbeitspaketen vorgenommen.*

1. **Projektmanagement Business-Blueprint:** *In diesem initialen Arbeitspaket werden unter anderem mehrere Geschäftsprozessworkshops organisiert. Während dieser Workshops wird überlegt, wie man die vorhandenen Geschäftsprozesse mit der einzuführenden betriebswirtschaftlichen Standardsoftware unterstützen kann. Es ist zu untersuchen, ob es sinnvoll ist, Geschäftsprozesse zu verändern.*

2. **Schulung des Projektteams Business-Blueprint:** *Dieses Arbeitspaket umfasst die Organisation von SAP-Schulungen.*

3. **Entwicklung der Systemumgebung:** *Die technischen Anforderungen an Netzwerk, Drucker und PCs werden in diesem Arbeitspaket festgelegt. Die Änderungsauftragsbearbeitung und -durchführung werden in diesem Arbeitspaket ebenfalls konzipiert. Unter Änderungsaufträgen versteht man die Übernahme (Import) von geänderten Objekten vom Entwicklungssystem in das Qualitätssicherungs- und Testsystem sowie die Freigabe von Änderungsaufträgen. Dieses Arbeitspaket dient auch der Spezifikation der Versionsführung. Dabei werden Festlegungen bezüglich der Reihenfolge, in der die einzelnen SAP R/3-Systeme auf eine neue Systemversion gebracht werden, getroffen. Außerdem wird geklärt, welche Versionsarten benutzt werden sollen. Als Versionsart stehen Functionality-Releases, die zusätzliche Funktionalität beinhalten, sowie Correction-Releases, die lediglich Fehler in der Software korrigieren, zur Verfügung. Zeitpunkte für das Einspielen von Hot-Packages werden festgelegt. Unter einem Hot-Package verstehen wir eine neue Version, die lediglich die Korrektur eines einzelnen Fehlers enthält. Die Auswirkungen der Versionsführung auf die Verfügbarkeit des Systems werden ebenfalls diskutiert. SAP R/3 folgt der in Abschnitt 2.6.2 dargestellten dreistufigen Client-Server-Architektur. Folglich muss die Software der Präsentationsschicht, die SAP GUI, auf den Client-Rechnern installiert werden. In diesem Arbeitspaket werden Strategien zur*

Installation der SAP GUI als spezielle Fragestellung der Softwaredistribution diskutiert. Das Entwicklungssystem wird ebenfalls innerhalb dieses Arbeitspakets vorbereitet. Eine Systemadministration wird etabliert. Diese Aktivität umfasst die Entwicklung einer Backupstrategie, das Überwachen von Benutzern sowie Maßnahmen zur Performanceanalyse. Der SAP R/3-typische Einführungsleitfaden **Implementation-Guide (IMG)** *wird konfiguriert. Der in der Q&A-Datenbank ermittelte Projektumfang wird zur Erstellung von Projekt-IMGs herangezogen. Anschließend werden die notwendigen betriebswirtschaftlichen Transaktionen ermittelt. Zusätzlich werden die Customizing-Transaktionen, die zur Systemanpassung herangezogen werden müssen, bestimmt.*

4. **Organisationsstruktur und Geschäftsprozessdefinition:** *In diesem Arbeitspaket wird die Organisationsstruktur durch die Festlegung der strukturellen Parameter des abzubildenden Unternehmens beschrieben. Dabei wird die Aufbauorganisation des Unternehmens zugrundegelegt. Außerdem wird der Informationsfluss zwischen den Organisationseinheiten, durch Ablauforganisation und Prozessorganisation beschrieben, bei der Abbildung berücksichtigt. Technische Restriktionen für einen verteilten Einsatz des Systems werden in diesem Arbeitspaket ebenfalls untersucht und anschließend berücksichtigt.*

BPML

Das Ergebnis der Business-Blueprint-Phase ist die **Business-Process-Master-List (BPML)**.

5.3.8.4 Prototypphase

Prototypphase

Das Hauptziel dieser Phase besteht im Test der zuvor eingestellten Parameter des Standardsoftwaresystems unter realistischen Bedingungen [13]. Wir beschreiben die Aufgaben dieser Phase anhand von SAP R/3. Das Vorgehen ist aber bei anderen betriebswirtschaftlichen Standardsoftwaresystemen ähnlich. Die Arbeiten können im Fall von SAP R/3 in die nachfolgenden Arbeitspakete unterteilt werden.

Systemverwaltung

1. **Arbeiten in der Systemverwaltung:** In diesem Arbeitspaket werden zunächst das Qualitätssicherungs- und Testsystem und das Produktivsystem vorbereitet. Pläne für den Ausfall der Hardware werden erarbeitet. Durchsatzpläne für Tests werden entwickelt, die sicherstellen, dass die Leistungsfähigkeit des betriebswirtschaftlichen Standardsoftwaresystems auch in Spitzenzeiten ausreichend ist. Man spricht hier von **Lasttests**. Die zuvor entwickelten Backup- und Recoveryverfahren werden in diesem Arbeitspaket überprüft.

2. **Übernahme der Stammdaten:** Eine manuelle Datenübernahme ist im Fall relativ kleiner Datenmengen möglich und im Fall schlecht strukturierter Datenbestände sogar erforderlich. Anzustreben ist aber eine weitestgehend automatische Übernahme durch einen automatisierten Transfer vom Altsystem zum neu einzuführenden SAP R/3-System. Dabei existieren prinzipiell zwei Möglichkeiten. Zum einen können standardisierte Programme aus der **Datenübernahmeworkbench** für die wichtigsten Geschäftsobjekte verwendet werden. In diesem Fall ist die Struktur der zu übertragenden Daten weitestgehend festgelegt. Andererseits können auch individuell angepasste Übernahmeprogramme verwendet werden. Derartige Programme ermöglichen die Übertragung von Daten, die in frei strukturierbaren Datenstrukturen abgelegt werden. Wir betrachten dazu das nachfolgende Beispiel. Stammdatenübernahme

   Beispiel 5.3.46 (Datenübernahme mit Direct- und Batch-Input) *Der Direct-Input fügt Daten unter Umgehung aller Prüfungen direkt in die Tabellen der Datenbank des SAP R/3-Systems ein. Anders als beim Direct-Input werden beim Batch-Input einzelne Transaktionen des SAP R/3-Systems wiederholt zur Anwendung gebracht und auf diese Art und Weise auch Plausibilitätsprüfungen während des Einfügens durchgeführt.*

3. **Entwicklung von Schnittstellenprogrammen für andere Anwendungen:** Die effiziente Abwicklung von Geschäftsprozessen über Systemgrenzen hinweg verlangt häufig die Schaffung von Schnittstellen. Wie in Abschnitt 3.5 gezeigt, sind Schnittstellen wesentlicher Bestandteil eines Kopplungssystems. Das Ziel der Schaffung von Schnittstellen besteht in der Reduzierung von Medienbrüchen, d.h. systemseitig bedingten Einschnitten im Informationsfluss. Bei der Festlegung von Schnittstellen müssen die folgenden Punkte beachtet werden (vergleiche dazu auch die Ausführungen in Abschnitt 3.5): Schnittstellen

   (a) Welche Daten müssen zwischen den Anwendungssystemen ausgetauscht werden?

   (b) Welches System ist das steuernde Anwendungssystem?

   (c) Wie werden Fehlersituationen behandelt?

   In SAP R/3-Systemen wird ABAP/4 als Programmiersprache zur Realisierung von Schnittstellen verwendet. Andere mögliche Softwaretechnologien sind unter anderem in [39, 3, 1] beschrieben.

4. **Entwicklung von Berichten:** Mehrere 1000 Standardreports existieren in SAP R/3 [1]. Falls kein geeigneter Report direkt vorhanden ist, kann ein ähnlicher Report als Ausgangsbasis für eine Individualentwicklung im Sinne einer Modifikation genutzt werden. Bei der Reportentwicklung muss Reports

die benötigte Leistungsfähigkeit der Programme beachtet werden, die zu einer angestrebten Online- oder Batchverarbeitung führt.

Berechtigungskonzept

5. **Erarbeitung des Berechtigungskonzeptes:** In einem Berechtigungskonzept wird festgelegt, welche betriebswirtschaftlichen Funktionen durch einen bestimmten Benutzer verwendet werden können. Wir betrachten dazu das folgende Beispiel.

   Beispiel 5.3.47 (Berechtigungskonzept) *Es ist offensichtlich, dass nur ausgewählte Mitarbeiter der Personalabteilung eines Unternehmens Zugang zu den Gehaltsberechnungsfunktionen im Modul HR haben dürfen.*

   In der Phase der Überprüfung der Projektorganisation und der Feinspezifikation wurden Verantwortlichkeiten für Funktionen und Prozesse festgelegt. Innerhalb der Prototypphase muss ein Herunterbrechen der Verantwortlichkeiten auf Ebene der Transaktionen und Programme erfolgen. Dabei kommen Arbeitsplatzmatrizen zum Einsatz [1]. Eine Arbeitsplatzmatrix weist einzelne Aktivitäten bestimmten Rollen zu. Jeder Aktivität ist ein Menüpfad zugeordnet, über den der Benutzer mit einer entsprechenden Rolle die zugehörigen Transaktionen aufrufen kann. Ein Berechtigungskonzept kann nur durch enge Kooperation zwischen Systemadministrator und den Verantwortlichen der Fachabteilungen und Geschäftsprozesse erstellt werden.

Integrationstest

6. **Abschließender Integrationstest:** Das Ziel des Integrationstests besteht in der Überprüfung der funktionalen Anforderungen an das SAP R/3-System. Dazu wird der Betrieb eines Produktivsystems auf dem Qualitätssicherungs- und Testsystem simuliert. Es sind Tests erforderlich, die zu einer Interaktion aller Abteilungen führen. Wir sprechen an dieser Stelle von „End-to-End“-Szenarien. Die Testfälle früherer Phasen können in den Integrationstests benutzt werden. Es sind Lasttests für das jeweilige SAP R/3-System durchzuführen, welche die Frage beantworten, ob die in den vorherigen Phasen ausgewählte Hardware ausreichend dimensioniert ist [13].

Die dritte Phase der Implementation-Roadmap von ASAP beschäftigt sich mit der Realisierung. Wir beschreiben diese Phase im nachfolgenden Beispiel [1].

Beispiel 5.3.48 (Realisierungsphase in ASAP) *Die Phase umfasst abschließende Arbeiten der Feinspezifikationsphase und große Teile der Prototypphase des allgemeines Phasenmodells. Die Umsetzung der Anforderungen und Entwurfsentscheidungen erfolgt in zwei Schritten:*

1. *Baseline-Konfiguration und -Abnahme: Die Konfiguration wird inkrementell in mehreren Schritten durchgeführt. Der Umfang der Baseline-Konfiguration umfasst vorrangige Unternehmensanforderungen, die in der*

zweiten Phase der Implementation-Roadmap von ASAP definiert worden sind. Diese werden automatisch aus der BPML abgeleitet. Die Baseline-Konfiguration umfasst die nachfolgenden Schritte. Während der Konfiguration werden Grundeinstellungen wie Länder und Währungen vorgenommen. Die Abbildung der Unternehmensstruktur findet unter Benutzung von Customizing-Werkzeugen statt. Als nächster Schritt wird der Test durchgeführt. Dabei kommt es zu einer Kontrolle der tatsächlich durchgeführten Einstellungen. Die vorgenommenen Einstellungen werden auf das Qualitätssicherungs- und Testsystem transportiert. Vordefinierte Testfälle werden durchgeführt. Im letzten Schritt erfolgt eine Abnahme durch die Projektleitung.

2. *Detailkonfiguration und -abnahme: An dieser Stelle rücken funktionsübergreifende Aspekte des Unternehmens in den Vordergrund. Bis zu vier Zyklen werden durchlaufen. Die Abnahme der Ergebnisse eines jeden Zyklus erfolgt durch die Projektleitung. Komplexe Geschäftsvorfälle werden abgebildet. Diese stellen Abfolgen miteinander verbundener Transaktionen dar.*

Auf die Darstellung der Phasen Probebetrieb und Produktivbetrieb wird an dieser Stelle verzichtet, da der Probebetrieb der Prototypphase ähnelt und Probleme des Produktivbetriebs im nächsten Abschnitt behandelt werden.

5.3.9 Ständige Verbesserung der betriebswirtschaftlichen Standardsoftware

5.3.9.1 Motivation und notwendige Arbeitsschritte

Verbesserung der betriebswirtschaftlichen Standardsoftware

Die ständige Verbesserung der betriebswirtschaftlichen Standardsoftware ist aus zwei Gründen notwendig. Zum einen verändert sich das technologische Umfeld, insbesondere Betriebssysteme und Datenbanken sowie Webtechnologien, ständig. Andererseits ändert sich die Unternehmensorganisation durch innere Anpassungen und selbstinitiierte Verbesserungsmaßnahmen. Das bedeutet insbesondere, dass die Lücke zwischen den neuartigen Anforderungen und dem Zustand der IT-Landschaft des Unternehmens geschlossen werden muss.

Die Verbesserung der betriebswirtschaftlichen Standardsoftware stellt die längste Phase im Lebenszyklus von Standardsoftware dar. Die dabei durchzuführenden Tätigkeiten können wie folgt beschrieben werden [13]:

1. erneute Veränderung der eingestellten Parameter, um eine vergrößerte Leistungsfähigkeit des betriebswirtschaftlichen Standardsoftwaresystems zu erhalten,

2. weitere Anpassungen des betriebswirtschaftlichen Standardsoftwaresystems aufgrund einer Umstellung der betrieblichen Abläufe,

3. funktionale Erweiterungen der Standardsoftware in Form der Einführung weiterer Module oder der Erstellung individueller Erweiterungen,

4. Übergang (Migration) auf eine aktuelle Version,

5. Entscheidung für Ersatz einer Standardsoftware durch ein neues bzw. anderes betriebswirtschaftliches Standardsoftwaresystem.

Die beschriebenen Arbeitsschritte werden nicht sequentiell, sondern in Abhängigkeit von der Problemsituation und dem Erkenntnisstand ausgeführt.

5.3.9.2 Funktionale Erweiterung der Standardsoftware

Individualentwicklung

Falls der Standardfunktionsumfang für eine bestimmte betriebliche Aufgabe nicht ausreichend ist, muss das Unternehmen eine Individualentwicklung zur Erweiterung des Funktionsumfangs der Standardsoftware vornehmen. Die Unternehmensführung soll aber stets den Beweis für die Notwendigkeit einer Erweiterung verlangen.

Für Erweiterungen bieten sich prinzipiell zwei Möglichkeiten an. Einerseits kann eine Erweiterung unter Verwendung der Entwicklungsumgebung des betriebswirtschaftlichen Standardsoftwaresystems erfolgen. Andererseits können Erweiterungen unter Verwendung anderer Programmiersprachen und der durch das betriebswirtschaftliche Standardsoftwaresystem unterstützten Schnittstellentechnologien durchgeführt werden. Wir betrachten dazu im folgenden Beispiel die Möglichkeiten für SAP R/3 bzw. mySAP ERP.

Beispiel 5.3.49 (Erweiterung des Funktionsumfangs) *Bei einer Erweiterung auf Basis von ABAP besteht die Möglichkeit einer einfachen Integration von Erweiterungen, da vordefinierte Aufrufschnittstellen (User-Exits) in den SAP R/3-Komponenten vorhanden sind. Die ABAP-Development-Workbench dient als Entwicklungswerkzeug. Nach Möglichkeit sind bestehende Programme nicht zu modifizieren, sondern tatsächlich Erweiterungen zu schaffen, da sonst ein Verlust der Releasefähigkeit möglich ist.*

Bei der Schaffung einer Erweiterung ist es stets notwendig, die nachfolgenden Fragen zu beantworten [1]:

1. Welche Geschäftsprozesse sollen durch die Erweiterung unterstützt werden?

2. Welche erweiterten, neuen Funktionalitäten werden dazu benötigt?

3. Welche Daten werden von der Erweiterung benötigt? Welche Daten stellt die Erweiterung zur Verfügung?

4. Können Schnittstellen benutzt werden?

5. Wie und vor allen Dingen von wem sollen die Erweiterungen gepflegt werden? Eine Pflege ist unter anderem bei der Veränderung von Schnittstellen notwendig.

Im nachfolgenden Abschnitt diskutieren wir exemplarisch Schnittstellentechnologien am Beispiel der betriebswirtschaftlichen Standardsoftware SAP R/3.

5.3.9.3 Schnittstellen am Beispiel von SAP R/3

Schnittstellen

Schnittstellen werden überall dort benötigt, wo mehrere Anwendungssysteme, die nicht von vornherein über eine gemeinsame Datenbank integriert sind, miteinander verbunden werden sollen. Wir gehen an dieser Stelle auf Business-Application-Programming-Interfaces (BAPIs) sowie Application-Link-Enabling (ALE) ein. Diese beiden Schnittstellentechnologien werden im Kontext der in Abschnitt 3.5 dargestellten Kopplungsarchitekturen diskutiert.

BAPIs

Der BAPI-Kopplungsmechanismus bietet eine objektorientierte Schnittstelle an, über die auf die durch eine SAP R/3-Komponente zur Verfügung gestellten Daten oder Funktionen zugegriffen werden kann. BAPIs sind im Zusammenhang mit der Business-Framework-Architecture (BFA) von SAP R/3 entwickelt worden [1]. Die BFA teilt das R/3-System in Business-Components auf. Jede einzelne Business-Component besteht aus Geschäftsobjekten, in der SAP-Terminologie als Business-Objects bezeichnet. Geschäftsobjekte wurden bereits in Abschnitt 2.3.1.3 eingeführt. In SAP R/3 dienen sie dazu, die zur Beschreibung einer Entität notwendigen Daten sowie die auf den Daten arbeitenden Funktionen objektorientiert zu kapseln. Durch ein Geschäftsobjekt angebotene Funktionen werden als Methoden bezeichnet. Die in einer Business-Component enthaltenen Geschäftsobjekte bilden die Schnittstelle der Business-Component. Das Business-Object-Repository (BOR) verwaltet die Geschäftsobjekttypen. Im BOR wird vorgehalten, welche BAPIs zu einem Typ existieren. Wir geben das folgende Beispiel für ein BAPI an.

Beispiel 5.3.50 (BAPI) *Wir betrachten das BAPI BAPI_SALESORDER_CREATEFROMDAT2, mit dem ein Auftrag im R/3-System angelegt werden kann.*

Funktionsbausteine

BAPIs werden in Form von Funktionsbausteinen realisiert. Funktionsbausteine sind spezielle Programmmodule, die in ABAP implementiert sind. Eine wesentliche Eigenschaft von Funktionsbausteinen ist die Möglichkeit, ihre Funktionalität rechnerübergreifend mittels Remote-Function-Call (RFC) zu nutzen. RFC basiert prinzipiell auf RPC (vergleiche hierzu die Ausführungen in Abschnitt 4.2.3), gleichzeitig wird aber eine Spezialisierung durch SAP-eigene Protokolle und Schnittstellen vorgenommen. Auf BAPIs kann sowohl funktions- als auch objektorientiert unter Verwendung von Programmiersprachen wie Java oder C# zugegriffen werden. Funktionsbausteine werden in der ABAP-Development-Workbench verwaltet. Die einem Geschäftsobjekt zugehörigen Daten sind in den

R/3-Tabellen abgelegt. Die Tabellen werden im ABAP-Data-Dictionary verwaltet.

Bei einem funktionsorientierten Zugriff wird auf den zum BAPI zugehörigen Funktionsbaustein unter Verwendung von RFC direkt zugegriffen. Bei einem objektorientierten Zugriff hingegen wird auf das BOR zugegriffen. Das BOR leitet den Zugriff weiter. Im aufrufenden Anwendungssystem wird dabei ein Proxy-Objekt verwendet, das im ersten Schritt den Aufruf empfängt und im zweiten zweiten Schritt den RFC-Aufruf durchführt [39]. Die beschriebene Situation ist in Abbildung 5.8 dargestellt.

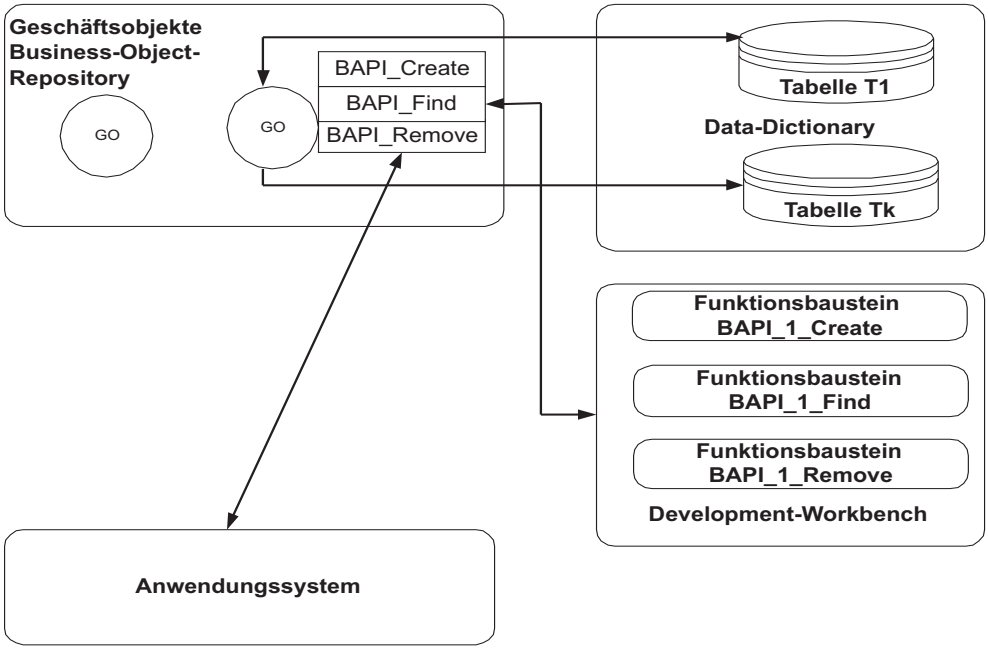

Abbildung 5.8: BAPIs in SAP R/3 (in Anlehnung an [1])

Durch BAPIs können grundsätzlich, allerdings zum Teil sehr aufwendig, alle in Abschnitt 3.5 beschriebenen Kopplungsarchitekturen unterstützt werden [39].

Die Realisierung einer **funktionsorientierten Kopplungsarchitektur mit nicht redundanter Modulhaltung, gemeinsamem Prozess und nicht redundanter Datenhaltung** ist relativ leicht zu erreichen. Die gemeinsam benutzten Funktionen und die dazugehörigen Daten werden als eigener Objekttyp realisiert. Über BAPIs kann auf den Objekttyp zugegriffen werden. Probleme kann die fehlende Unterstützung bei der Überwindung der fachlichen Heterogenität bereiten.

Funktionsorientierte Kopplungsarchitekturen mit redundanter Modulhaltung, separaten Prozessen und redundanter Datenhaltung sind im Gegensatz zu funktionsorientierten Kopplungsarchitekturen mit nicht redundanter Modul- und Datenhaltung nicht leicht unter Verwendung von BA-

PIs zu realisieren. Insbesondere muss die Funktionalität, die Änderungen an den redundanten Modulen vom Quell- ins Zielanwendungssystem überträgt, zusätzlich implementiert werden. Außerdem wird ein Datenabgleich bei redundanter Datenhaltung ebenfalls nur unzureichend unterstützt.

Übungsaufgabe 5.10 (Ereignisorientierte Kopplungsarchitekturen) *Untersuchen Sie die Möglichkeit, eine ereignisorientierte Kopplungsarchitektur mit Hilfe von BAPIs zu realisieren.*

Datenorientierte Kopplungsarchitekturen mit nicht redundanter Datenhaltung können unter Verwendung von BAPIs relativ leicht implementiert werden, wenn gemeinsam benutzte Daten im SAP R/3-System vorgehalten werden. Der Zugriff eines anderen Anwendungssystems auf diese Daten kann dann nach Implementierung eines entsprechenden BAPIs problemlos erfolgen. Probleme bereitet aber die häufig vorliegende fachliche Heterogenität, deren Überwindung durch die BAPIs nicht abgedeckt ist.

Im Gegensatz zu Kopplungsarchitekturen mit nicht redundanter Datenhaltung werden **Kopplungsarchitekturen mit redundanter Datenhaltung** durch BAPIs nicht gut unterstützt.

Übungsaufgabe 5.11 (BAPIs) *Begründen Sie die Aussage, dass BAPIs Kopplungsarchitekturen mit redundanter Datenhaltung nicht gut unterstützen.*

ALE-Kopplung

Für die Realisierung derartiger Kopplungsarchitekturen, insbesondere auch zur Kopplung mehrerer SAP R/3-Systeme, wurde der ALE-Kopplungsmechanismus geschaffen.

Bei einer einfach gehaltenen Unternehmensstruktur wie einer Gesellschaft und wenigen Standorten reicht häufig ein einzelnes SAP R/3-System als Produktivsystem. Bei großen Konzernen, die häufig global agieren, ist diese Situation nicht mehr gegeben. In diesem Fall ist es nicht möglich, eine Architektur zu verwenden, die über genau eine Datenbank verfügt. Es sind Architekturen notwendig, bei denen mehrere Anwendungsserver mit eigenen Datenbanken miteinander gekoppelt werden. Wir betrachten dazu das folgende Beispiel [1].

Beispiel 5.3.51 (Einsatz mehrerer produktiver SAP R/3-Systeme) *Wir untersuchen den Fall, dass ein Konzern aus mehreren Tochtergesellschaften besteht. Die Tochtergesellschaften betreiben eigene, dezentrale SAP R/3-Systeme. Periodisch werden wichtige Informationen auf ein zentrales SAP R/3-System übertragen, das vom Konzern verwendet wird. Das zentrale SAP R/3-System erstellt auf dieser Basis Reports, die für die Konzernzentrale von Bedeutung sind. Umgekehrt erhalten die Tochtergesellschaften Vorgaben der Konzernzentrale, die auf die dezentralen SAP R/3-Systeme übertragen werden. Offensichtlich muss bei diesem Ansatz sichergestellt sein, dass Zentrale und Tochtergesellschaften über einheitliche und aktuelle Daten verfügen.*

Übungsaufgabe 5.12 (Konzernweite Datenhaltung) *Begründen Sie, warum es für einen global agierenden Konzern sinnvoll ist, mehrere SAP R/3 Systeme mit separaten Datenbanken zu unterhalten.*

Die Verteilung von Geschäftsprozessen durch Konfiguration und Betrieb von verteilten SAP R/3- und Fremdanwendungen wird durch ALE ermöglicht. ALE unterstützt einen betriebswirtschaftlich gesteuerten Nachrichtenaustausch zwischen lose gekoppelten, verteilten Anwendungssystemen bei konsistenter Datenhaltung [1]. Jedes der beteiligten Anwendungssysteme besitzt eine eigene Datenbank. Dadurch ist es möglich, dass die Datenbanken redundant und unter Umständen auch inkonsistent sind. Die in den Datenbanken vorgehaltenen Datenbestände werden durch asynchrone Kommunikation von Intermediate Documents (IDocs) synchronisiert. Bei Verwendung von ALE müssen die nachfolgenden Festlegungen durch den Anwender getroffen werden:

IDocs

- Verteilung der Anwendungen auf Rechner,
- Benennung der Daten, die ausgetauscht werden sollen,
- Benennung der Customizing-Daten, die dazu benötigt werden.

Verschiedene Varianten des ALE-Mechanismus existieren. Neben dem IDoc-Mechanismus gibt es auch eine Variante, bei der die Kopplung unter Verwendung von BAPIs vorgenommen wird [39]. Im weiteren Verlauf beschreiben wir eine Variante, die Stammdaten unter Verwendung des Shared-Master-Data (SMD)-Tools synchronisiert. Beim Ändern, Neuanlegen und Lösen eines Stammdatensatzes werden Änderungsbelege für die Verteilung der Stammdaten durch das SMD-Tool erzeugt. Der Änderungszeigermechanismus verwendet diese Belege, um zu entscheiden, ob die protokollierten Änderungen zu übertragen sind. Falls eine Übertragung notwendig ist, werden Änderungen an Stammdatenobjekten in Form von Änderungszeigern protokolliert. Der Änderungsbelegmechanismus bildet gemeinsam mit dem Änderungszeigermechanismus das Ereignissubsystem im Sinne von Abschnitt 3.5. Das ALE-Verteilungssystem erlaubt die Formulierung von Regeln, an welche Ziel-Anwendungssysteme die Daten zu übertragen sind. Somit wird das Empfängersubsystem aus Abbildung 3.11 durch das ALE-Verteilungssystem gebildet. Diese Aufzählung lässt sich fortsetzen.

SMD-Tool

Wir sehen deutlich, dass sowohl für die Datenübertragung als auch für den Datenabgleich durch ALE bereits mächtige Mechanismen existieren, die nur mit hohem Aufwand direkt implementiert werden können.

5.4 Software-Rahmenwerke

5.4.1 Motivation

Bisher wurden im Rahmen dieses Kurses betriebswirtschaftliche Individual- und Standardsoftware diskutiert. Sowohl Individual- als auch Standardsoftware be-

sitzen Vor- und Nachteile. Durch betriebswirtschaftliche Standardsoftware gehen insbesondere in vielen Fällen Wettbewerbsvorteile verloren. Aus diesem Grund wird gefordert, dass betriebswirtschaftliche Standardsoftware nicht für die Kernbereiche eines Unternehmens eingesetzt werden soll [21]. Betriebswirtschaftliche Standardsoftware soll zur Realisierung von Unterstützungsprozessen wie Finanzbuchhaltung und Lagerhaltung verwendet werden. Auf diesen Sachverhalt wurde auch bei der Diskussion des Kern-Schalen-Modells in Abschnitt 5.3.5 hingewiesen.

Da die Entwicklung von Individualsoftware aber zu teuer ist, sind hybride Anwendungsentwicklungskonzepte, die sowohl die Vorteile von Standard- als auch Individualsoftware vereinen und die Nachteile beider Ansätze entsprechend verringern, in starkem Maße wünschenswert. Wir werden in den nachfolgenden Abschnitten zeigen, dass objektorientierte bzw. komponentenbasierte Rahmenwerke einen wichtigen Schritt zur Erreichung dieses Ziels darstellen.

5.4.2 Begriffsbestimmungen

Wir definieren zunächst den Begriff des Rahmenwerkes in Anlehnung an [36, 9]. Rahmenwerk

Definition 5.4.1 (Rahmenwerk) *Ein Rahmenwerk (Framework) ist ein Architekturgerüst, das eine allgemeine generische Lösung für ähnliche Probleme in einem bestimmten Kontext vorgibt. Wiederverwendet werden nicht einzelne Klassen, sondern die gesamte Konstruktion aus zusammenspielenden Komponenten. Ein Rahmenwerk gibt auf diese Art und Weise den Kontrollfluss für die Anwendung vor. Unter dem Kontrollfluss einer Anwendung verstehen wir dabei die zeitliche Abfolge der auszuführenden Operationen.*

Durch Definition 5.4.1 wird Definition 2.6.3 aus Abschnitt 2.6.3 vervollständigt.

Wesentliche Idee für den Einsatz von Rahmenwerken ist die strukturelle und inhaltliche Ähnlichkeit von bestimmten Problemdomänen. Dadurch ist eine Abstraktion der Problemdomäne mit Hilfe von Rahmenwerken möglich. Rahmenwerke legen die grundlegenden Verhaltensweisen und Beziehungen zwischen den Objekten einer Anwendungsdomäne fest. Das durch ein Rahmenwerk vorgegebene halbfertige Architekturgerüst muss an die Bedürfnisse und Anforderungen einer konkreten Anwendung aus der Problemdomäne angepasst werden [37]. Problemdomäne

5.4.3 Eigenschaften von Rahmenwerken

Rahmenwerke besitzen die nachfolgenden drei charakteristischen Merkmale [9]:

1. Invertierung des Kontrollflusses,
2. Vorgabe einer konkreten Anwendungsarchitektur,
3. Anpassbarkeit an die Anforderungen einer konkreten Anwendung durch vordefinierte Erweiterungspunkte.

Traditionell entwickelte Anwendungssysteme nutzen häufig Klassenbibliotheken zur Realisierung der Anwendungsfunktionalität. Die Funktionen in den Klassenbibliotheken werden von der sie nutzenden Anwendung aufgerufen. Sie übernehmen aber nicht den Hauptkontrollfluss der Anwendung. Diese Art des Aufrufs externer Funktionen aus Klassenbibliotheken wird „Call-Down-Prinzip“ genannt. Im Gegensatz zu Rahmenwerken stellen Klassenbibliotheken keine Architektur bereit. Der Anwendungsentwickler ist gezwungen, die Architektur der Anwendung selber zu entwickeln.

Invertierung des Kontrollflusses

Rahmenwerke hingegen kehren den Kontrollfluss einer Anwendung um. Der Hauptkontrollfluss einer Anwendung, die auf Basis des Rahmenwerks entwickelt wurde, wird durch das Rahmenwerk gesteuert. Die Funktionen zur Realisierung der anwendungsspezifischen Funktionalität werden aus dem Rahmenwerk heraus aufgerufen. Diese Art des Aufrufs von Funktionen wird als „Call-Back-Prinzip“ bezeichnet [9]. Die „Call-Down“- und „Call-Back“-Aufrufmechanismen sind in Abbildung 5.9 dargestellt. Wiederverwendbarer Sourcecode ist in dieser Abbildung grau hinterlegt, während von den Entwicklern der Anwendung erstellter Code weiß dargestellt wird.

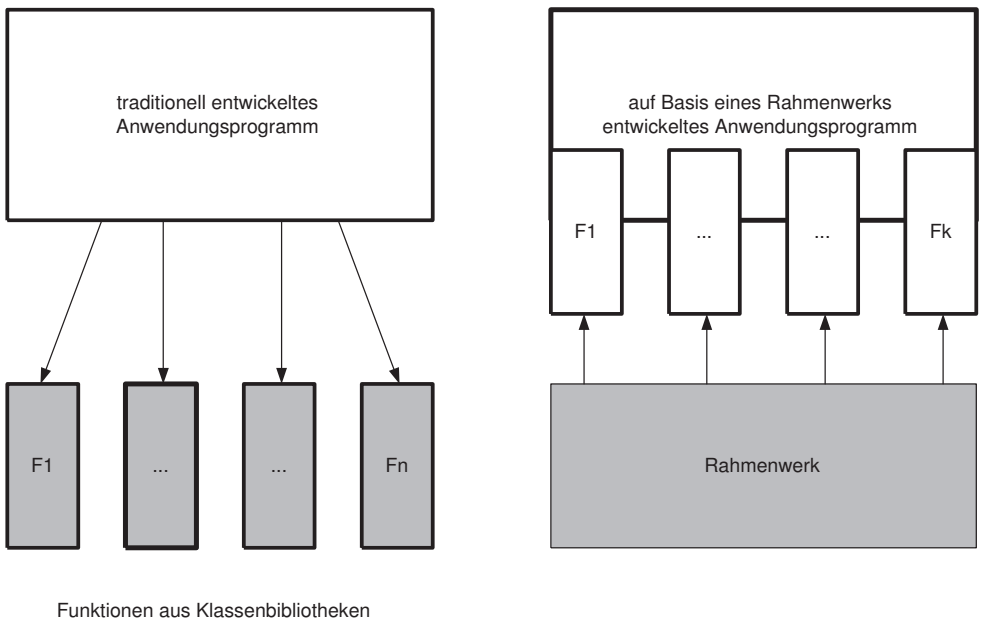

Abbildung 5.9: Kontrollfluss in Anwendungssystemen (an [37] angelehnt)

Anwendungsarchitektur

Ein Rahmenwerk definiert typischerweise bereits wesentliche Elemente einer Anwendungsarchitektur. Die Architektur wird lediglich noch an bestimmten Einschubpunkten flexibel gehalten. Rahmenwerke dienen somit als halbfertige, erweiterbare Softwarearchitekturen, sogenannte „Skelette“. Durch den Anwendungsentwickler müssen Funktionen hinzugefügt werden, welche die im Rah-

menwerk verankerten generischen Algorithmen an die spezielle Situation einer konkreten Anwendung anpassen.

Damit Rahmenwerke für eine Vielzahl von Anwendungen in einer bestimmten Problemdomäne eingesetzt werden können, enthalten sie offene Variationspunkte, die als Erweiterungspunkte bezeichnet werden. Erweiterungspunkte sind Stellen in der Architektur des Rahmenwerks, an denen eine Entscheidung über eine bestimmte Funktionalität bereits typisiert vorliegt, die spezifische Ausprägung der Funktionalität aber erst durch die konkrete Anwendung festgelegt wird.

Erweiterungspunkte

5.4.4 Rahmenwerkarten

Rahmenwerke unterscheiden sich durch die verwendeten Technologien, um ein auf dem Rahmenwerk basierendes Anwendungssystem an die konkreten Erfordernisse einer bestimmten Anwendung anzupassen. Wir unterscheiden **objektorientierte** Rahmenwerke von **komponentenbasierten** Rahmenwerken.

Objektorientierte Rahmenwerke ermöglichen Adaptivität durch Vererbung, durch Überschreibung von Methoden in abstrakten Klassen oder durch Implementierung von Schnittstellen.

objektorientierte Rahmenwerke

Bei komponentenbasierten Rahmenwerken wird Adaptivität an bestimmte Anwendungen durch unterschiedliche Komposition der Komponenten erreicht. Weiterhin existieren Rahmenwerke, die sowohl über objektorientierte als auch komponentenbasierte Eigenschaften verfügen.

komponentenbasierte Rahmenwerke

Die hier vorgenommene Unterscheidung wird in [40] für Rahmenwerke für betriebswirtschaftliche Anwendungen konkretisiert.

5.4.4.1 Objektorientierte Rahmenwerke

Wir definieren zunächst den Begriff des objektorientierten Rahmenwerks in Anlehnung an [19]. Ein objektorientiertes Rahmenwerk ist ein spezielles Rahmenwerk, dessen atomare Einheiten Klassen sind. Wir erhalten:

Definition 5.4.2 (Objektorientiertes Rahmenwerk) *Eine fest vorgegebene Menge von Klassen, die einen abstrakten Lösungsentwurf für eine Familie verwandter Probleme darstellen, wird als objektorientiertes Rahmenwerk bezeichnet.*

In manchen Situationen ist es sinnvoll, anstelle von objektorientierten **agentenbasierte Rahmenwerke** zu betrachten. Agentenbasierte Rahmenwerke sind spezielle objektorientierte Rahmenwerke, bei denen die atomaren Einheiten Agententypen sind. Wie in Abschnitt 4.1 gezeigt, können Softwareagenten als eine natürliche Verallgemeinerung von Objekten angesehen werden.

agentenbasierte Rahmenwerke

Objektorientierte Rahmenwerke werden durch Mengen von konkreten und abstrakten Klassen gebildet. Durch diese Klassen ist ein unvollständiger Entwurf und eine unvollständige Implementierung für Anwendungen in einer bestimmten Domäne gegeben. Ein objektorientiertes Rahmenwerk definiert die Struktur der

Klassen sowie deren Zusammenwirken. Außerdem wird der Kontrollfluss vorgegeben. Ein objektorientiertes Rahmenwerk legt somit die Architektur einer Anwendung fest, die auf Basis des Rahmenwerks entwickelt wird. Der Entwickler wird entlastet und kann sich somit besser um fachliche Details der speziellen Anwendung kümmern. Die Entwickler erweitern das Rahmenwerk um anwendungspezifische Details durch Ableitung von neuen Klassen aus den abstrakten Klassen.

Offen-Geschlossen-Prinzip

Ein objektorientiertes Rahmenwerk muss offen sein bezüglich zukünftiger Erweiterungen. Gleichzeitig muss es geschlossen bezüglich Modifikationen sein. Wir sprechen an dieser Stelle vom **Offen-Geschlossen-Prinzip** [49]. Offenheit bedeutet hier, dass das Rahmenwerk um neue, zur Entwurfszeit unbekannte Funktionalität erweitert werden kann. Unter Geschlossenheit versteht man in diesem Zusammenhang, dass die Anwendungslogik des Rahmenwerks nicht durch Bildung anwendungsspezifisch abgeleiteter Klassen derart modifiziert werden kann, dass ein Verhalten erzeugt werden kann, das nicht der eigentlichen Intention des Rahmenwerks entspricht. Geschlossenheit impliziert insbesondere, dass die Unterklassen lediglich abstrakte Methoden der Oberklassen des Rahmenwerks implementieren. Durch das Rahmenwerk bereits implementierte Methoden dürfen nicht überschrieben und modifiziert werden.

Wir unterscheiden zwei grundsätzliche Arten der Nutzung von objektorientierten Rahmenwerken im Hinblick auf die Wiederverwendung [37]:

1. White-Box-Rahmenwerk,
2. Black-Box-Rahmenwerk.

White-Box-Rahmenwerke

Die grundlegende Idee von **White-Box-Rahmenwerken** besteht darin, dass eine bestimmte Menge abstrakter Klassen durch das Rahmenwerk vorgegeben wird. Konkrete Klassen werden davon abgeleitet, indem bestimmte abstrakte Methoden der Oberklassen überschrieben werden. Dem Rahmenwerkanwender muss die Infrastruktur und Architektur des Rahmenwerks gut bekannt sein. Diese Klasse von Rahmenwerken ist sehr flexibel einsetzbar, da Einblicke in die Implementierung des Rahmenwerks möglich sind. Daraus begründet sich auch die Namensgebung für diese Art von Rahmenwerken.

Black-Box-Rahmenwerke

Im Gegensatz dazu werden bei **Black-Box-Rahmenwerken** nur Objekte konkreter Klassen erzeugt. In Black-Box-Rahmenwerken liegt somit eine bestimmte Anzahl bereits fertiger, ausimplementierter Klassen vor. Das Verhalten des Rahmenwerks kann durch Komposition der Klassen, aber nicht durch Vererbung mit Überschreibung erreicht werden. Die Wiederverwendung der Rahmenwerkfunktionalität findet im Black-Box-Fall auf Basis der wohldefinierten Schnittstellen der Klassen und deren vertraglicher Spezifikationen statt. Der Anwendungsentwickler implementiert anwendungspezifische Funktionalität gegen die externen Rahmenwerkschnittstellen. Adaptives Verhalten des Rahmenwerks wird zusätzlich durch eine Parametrisierung der Objekte erreicht. Wir betrachten dazu das nachfolgende Beispiel.

Beispiel 5.4.1 (Abbildung von adaptivem Verhalten) *Durch eine Verwendung von Templates in der Programmiersprache C++ kann adaptives Verhalten eines Rahmenwerks erreicht werden. Templates stellen, wie bereits in Abschnitt 3.4.3.1 beschrieben, parametrisierbare Klassen dar.*

Der Anwender des Rahmenwerks kann weniger IT-Experte als Domänenexperte sein. Allerdings ist die Flexibilität dieser Art von Rahmenwerken eingeschränkt.

Grey-Box-Rahmenwerke

Eine Vielzahl von Rahmenwerken beeinhaltet Eigenschaften beider Rahmenwerkarten. Diese Rahmenwerke werden als **Grey-Box-Rahmenwerke** bezeichnet. Graustufen können durch Öffnen eines Black-Box-Rahmenwerks hin zu einem White-Box-Rahmenwerk erreicht werden. Umgekehrt ist festzustellen, dass sich White-Box-Rahmenwerke mit zunehmendem Reifegrad hin zu Black-Box-Rahmenwerken entwickeln. Ein häufiger Einsatz von White-Box-Rahmenwerken für konkrete Anwendungen begünstigt diese Entwicklung [37].

Typischerweise werden mehrere objektorientierte Rahmenwerke bei der Entwicklung einer Anwendung eingesetzt. Dies wird unter anderem dadurch verursacht, dass heute Anwendungen typischerweise mehrere Domänen abdecken müssen, objektorientierte Rahmenwerke aber stets nur eine bestimmte Domäne abbilden. Der Einsatz mehrerer Rahmenwerke führt zu den nachfolgend aufgelisteten Problemen [37]:

1. Jedes der an der Anwendung beteiligten Rahmenwerke versucht, den Hauptkontrollfluss der Anwendung zu übernehmen.

2. Die Funktionalität der unterschiedlichen Rahmenwerke kann sich überlappen. Dadurch werden Teile der abzubildenden Domäne in mehreren Rahmenwerken modelliert. Das trifft häufig auch für die Basisklassen der objektorientierten Rahmenwerke zu. Veränderungen an den Basisklassen mit dem Ziel, die unterschiedlichen Rahmenwerke kompatibel zu machen, ziehen aber zahlreiche Änderungen an den von den Basisklassen abgeleiteten Klassen nach sich.

3. Die Interaktion der an der Anwendung beteiligten Rahmenwerke kann nur durch die Einführung einer zusätzlichen höheren Abstraktionsebene gelöst werden. Diese Ebene beschreibt, wie die Interaktion der unterschiedlichen Rahmenwerke erfolgt. Diese Teilsystem-Rahmenwerke werden als Rahmenwerke zweiter Ordnung bezeichnet. Sie dienen dazu, die einzelnen Teilsysteme zu kombinieren. Die Teilsysteme sind selber wieder eigenständige Rahmenwerke. Wir sprechen in diesem Zusammenhang von Rahmenwerken erster Ordnung. Rahmenwerk-Hierarchien können insbesondere auch als Komponentenrahmenwerke aufgefasst werden.

5.4.4.2 Komponentenbasierte Rahmenwerke

komponentenbasierte Rahmenwerke

Komponentenbasierte Rahmenwerke sind spezielle Rahmenwerke, deren atomare Einheiten Softwarekomponenten sind. Wir definieren den Begriff des komponentenbasierten Rahmenwerks in Anlehnung an [9] wie folgt.

Definition 5.4.3 (Komponentenbasiertes Rahmenwerk) *Eine Sammlung von verschiedenen Komponenten mit vordefiniertem Kooperationsverhalten, die mit dem Ziel der Aufgabenerfüllung in einer bestimmten Domäne entworfen werden, wird als komponentenbasiertes Rahmenwerk bezeichnet.*

Komponentenrahmenwerke stellen somit eine Sammlung von Regeln und Schnittstellen dar, welche das Zusammenwirken von Komponenten innerhalb eines Komponentenrahmenwerks regeln.

Vertragsähnliche Vereinbarungen zwischen den einzelnen Softwarekomponenten, die Dienste anbieten, sowie dem Komponentenrahmenwerk als Nutzer dieser Dienste werden durch Regeln und Schnittstellen bereitgestellt.

Spezifische Komponenten, die diesen Verträgen genügen, können an wohldefinierten, durch die Verträge typisierten Erweiterungspunkten in das Komponentenrahmenwerk eingebettet werden. Ähnlich wie bei objektorientierten Rahmenwerken stellt das Komponentenrahmenwerk den Kontrollfluss für eine auf dem Komponentenrahmenwerk basierende Anwendung zur Verfügung, indem es das Zusammenwirken der Komponenten erzwingt. Durch das Komponentenrahmenwerk wird somit die Anwendungsarchitektur vorgegeben.

Die Anpassung des Komponentenrahmenwerks an die unterschiedlichen Anwendungen erfolgt ausschließlich durch Komposition der unterschiedlichen Komponenten. Komponenten stellen somit die Variationspunkte eines komponentenbasierten Rahmenwerks dar. Somit werden keine einzelnen Klassen sondern ganze Komponenten komponiert.

Die Entwicklung einzelner Komponenten erfolgt häufig unter Verwendung objektorientierter Rahmenwerke. Falls ein objektorientiertes Rahmenwerk zur Entwicklung mehrerer Komponenten verwendet wird, besteht die Gefahr, dass Vererbungsbeziehungen von Klassen unterschiedlicher Komponenten erzeugt werden. Dadurch wird die Unabhängigkeit und Abgeschlossenheit von Komponenten (vergleiche hierzu Abschnitt 2.6.3) verletzt.

Das Konzept komponentenbasierter Rahmenwerke kann auch hierarchisch angewandt werden. In diesem Fall werden ganze Rahmenwerke ebenfalls wieder als Komponenten modelliert. Dadurch können unterschiedliche Rahmenwerke mit anderen Rahmenwerken interagieren. Komponentenbasierte Rahmenwerke, die als Komponenten gekapselt sind, können auf natürliche Art und Weise mit anderen komponentenbasierten Rahmenwerken zusammenarbeiten und sind folglich leicht zu integrieren. Da objektorientierte Rahmenwerke in einem geringeren Maße über diese Eigenschaft verfügen, wird durch komponentenbasierte Rahmenwerke ein höherer Grad an Wiederverwendung erreicht.

Neben objektorientierten und komponentenbasierten Rahmenwerken existieren auch **hybride Rahmenwerke**, die sowohl über Eigenschaften von objektorientierten als auch komponentenbasierten Rahmenwerken verfügen. Bei hybriden Rahmenwerken wird die Anpassung an konkrete Anwendungen sowohl durch Vererbung als auch durch Implementierung von geeigneten Schnittstellen erreicht. Die Implementierung von Schnittstellen ermöglicht wie im Fall von rein komponentenbasierten Rahmenwerken die Anpassung durch Komposition.

hybride Rahmenwerke

5.4.5 Konstruktionsprinzipien für objektorientierte Rahmenwerke

Ein wichtiger Schritt bei der Entwicklung eines Rahmenwerks besteht in der Festlegung der abzubildenden Domäne. Bei der Analyse der Anwendungsdomäne muss berücksichtigt werden, dass sich das Verhalten des Rahmenwerks in verschiedenen Anwendungssituationen verändern muss. Die Entwicklung eines Rahmenwerkes verlangt somit stets, Anforderungen an die Flexibilität des Rahmenwerks zu identifizieren und zu spezifizieren.

Im Folgenden wird der auf Pree [36] zurückgehende hotspot-getriebene Ansatz zur Entwicklung von objektorientierten Rahmenwerken beschrieben. Die Kernidee dieses Ansatzes besteht in der Identifikation von geeigneten Variationspunkten. Diese werden als Hotspots bezeichnet. Die Entwurfsmethodik des Hotspot-getriebenen Ansatzes kann wie folgt angegeben werden:

Hotspots

1. Festlegung eines anwendungsdomänenspezifischen Objektmodells,
2. Identifikation der Hotspots,
3. Entwurf und Überarbeitung des Rahmenwerks,
4. Einsatz des Rahmenwerks,
5. Überarbeitung der Hotspots und iterative Wiederholung der Schrittfolge ab Schritt 2.

Der initiale Schritt bei der Konstruktion eines objektorientierten Rahmenwerks besteht in der Erkennung und Festlegung der Schlüsselabstraktionen der untersuchten Anwendungsdomäne. Dazu können Class-Responsibility-Collaboration-Cards (CRC-Karten) [11] eingesetzt werden. CRC-Karten sind ein Hilfsmittel beim Entwurf objektorientierter Softwaresysteme. Ausgangspunkt ist das Identifizieren von Klassen. Für jede dieser Klassen wird eine Karte erstellt, die den Namen der Klasse, deren Verantwortlichkeiten sowie die Namen derjenigen Klassen enthält, die mit der Klasse in Beziehung stehen. Die Karten können zur Erstellung des Klassenmodells (vergleiche die Ausführungen in Abschnitt 3.4.3.1) herangezogen werden.

CRC-Karten

Für die an dieser Stelle notwendige Objektmodellierung ist Wissen über

die Anwendungsdomäne erforderlich. Das initiale Objektmodell wird nach der Identifizierung der Schlüsselabstraktionen in Cluster unterteilt. Eine bestimmte Menge von Klassen bzw. Schnittstellen bildet ein einzelnes Cluster. Der Entwicklungsprozess des Rahmenwerks wird durch die Dekomposition in Cluster begünstigt. Durch die Cluster wird der Entwicklungsprozess in mehrere Softwarelebenszyklen unterteilt, die sich zeitlich überlappen können und unabhängig voneinander sind. Falls sich während der Implementierungsarbeiten für ein Cluster herausstellt, dass die im Cluster zusammengefassten Schlüsselabstraktionen nicht geeignet sind, muss der Entwurf der Schlüsselabstraktionen gleichzeitig mit der Implementierung des Clusters überarbeitet werden.

Eine Möglichkeit zur Unterstützung der Identifizierung von Variationspunkten in objektorientierten Rahmenwerken stellen Variation-Point-Cards, auch als Hotspot-Karten bezeichnet, dar. Hotspot-Karten sind eine spezielle Variante von CRC-Karten. Sowohl Daten als auch Funktionen können als Hotspots identifiziert werden. Pree [36] unterscheidet folglich zwischen Daten- und Funktionen-Hotspot-Karten.

Funktionen-Hotspot-Karten dienen der Dokumentation von Funktionalität, die im zu entwickelnden Rahmenwerk flexibel sein soll.

Daten-Hotspot-Karten legen fest, welche Elemente der Anwendungsdomäne während der Rahmenwerkentwicklung verallgemeinert werden sollen. Daten-Hotspot-Karten werden dadurch realisiert, dass abstrakte Klassen in das Objektmodell aufgenommen werden. Aus diesem Grund sind Daten-Hotspot-Klassen nicht unbedingt erforderlich, anstelle der Karten kann direkt das Objektmodell verändert werden.

Im weiteren Verlauf der Kurseinheit beschäftigen wir uns deshalb ausschließlich mit Funktionen-Hotspots. Wenn wir von Hotspots sprechen, meinen wir Funktionen-Hotspots. Unter Verwendung von Hotspot-Karten kann ein Objektmodell in ein objektorientiertes Rahmenwerk transformiert werden. Eine Hotspot-Karte entspricht dabei in ihrer Granularität einer Methode.

Hooks

Für jeden durch eine Hotspot-Karte beschriebenen Hotspot wird ein **Hook** („Haken“) im objektorientierten Rahmenwerk definiert. Hooks legen fest, wo die zu den Hotspot-Karten zugehörigen Methoden in der Klassenhierarchie des Rahmenwerks platziert werden. Die eben beschriebenen Methoden werden als Einschubmethoden („hook methods“) bezeichnet.

Schablonen- und Einschubmethoden

In objektorientierten Rahmenwerken wird zwischen **Schablonenmethoden** („template methods“) und **Einschubmethoden** unterschieden. Einschubmethoden fungieren als abstrakte Platzhalter im Rahmenwerk. Die bereits implementierten Schablonenmethoden rufen die Einschubmethoden auf. Schablonenmethoden beschreiben abstraktes Verhalten, den generischen Kontrollfluss oder das Interaktionsverhalten von Objekten des Rahmenwerks. Das Verhalten des objektorientierten Rahmenwerks kann ausschließlich durch Überschreiben der Einschubmethoden verändert werden. Schablonenmethoden werden als Frozenspots bezeichnet, da sie die nicht mehr veränderbare Anwendungslogik des ob-

jektorientierten Rahmenwerks darstellen. Im nachfolgenden Beispiel wird das Zusammenwirken von Einschub- und Schablonenmethoden dargestellt.

Beispiel 5.4.2 (Einschub- und Schablonenmethoden) *In Abbildung 5.10 enthält die Klasse $K1$ sowohl die Einschubmethode $h()$ als auch die Schablonenmethode $t()$. Die Methode $t()$ ruft $h()$ auf. Die Klasse $K2$ wird von Klasse $K1$ abgeleitet. Im Zuge der Vererbung überschreibt die Klasse $K2$ die Einschubmethode $h()$ von Klasse $K1$.*

Schablonen- und Einschubmethoden können aber, anders als in Abbildung 5.10 gezeigt, auch in zwei getrennten Klassen enthalten sein. Klassen, die Einschubmethoden zur Verfügung stellen, werden als Einschubklassen („hook classes“) bezeichnet. Klassen mit Schablonenmethoden werden Schablonenklassen („template classes“) genannt. Die Schablonenklasse wird dabei durch die Einschubklasse parametrisiert. Die Klasse $K1$ in Abbildung 5.10 ist sowohl Schablonen- als auch Einschubklasse und parametrisiert sich somit selbst.

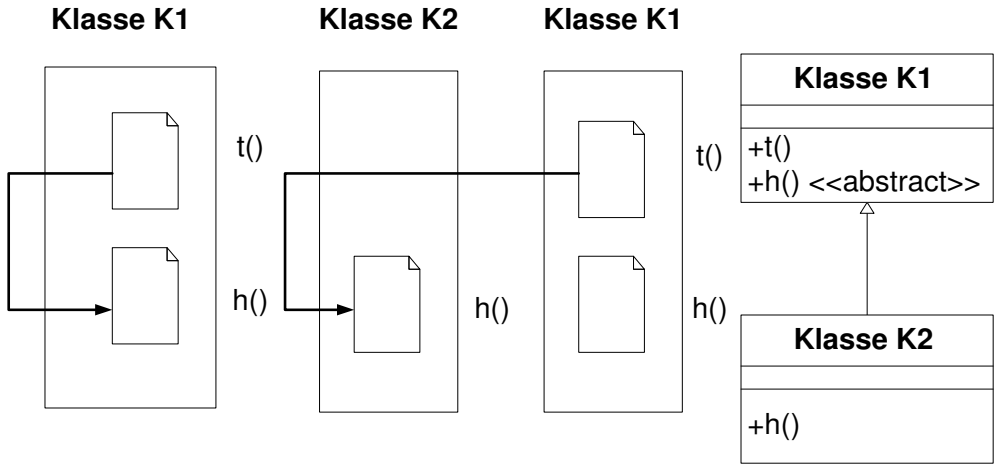

Abbildung 5.10: Zusammenhang von Einschub- und Schablonenmethoden [37]

Die Entscheidung, ob Einschub- und Schablonenmethoden in einer Klasse definiert werden oder auf verschiedene Klassen aufgeteilt werden, hängt davon ab, ob eine Anpassung der auf dem Rahmenwerk basierenden Anwendung zur Laufzeit vorgenommen werden soll oder nicht. Es findet entweder das **Unifikations-** oder das **Separationsprinzip** Anwendung.

Unifikationsprinzip

Schablonen- und Einschubmethoden werden beim Unifikationsprinzip in einer gemeinsamen Klasse TH definiert. Anpassungen an konkrete Situationen in der Anwendungsdomäne sind hier durch Vererbung, d.h. durch Bildung von Unterklassen, möglich. Die Bildung von Unterklassen verlangt aber einen Neustart der Anwendung. Eine Anpassung zur Laufzeit der Anwendung ist somit nicht möglich.

Separationsprinzip

Wenn eine Anpassung der Anwendung zur Laufzeit erreicht werden soll, ist eine Aufteilung von Schablonen- und Einschubmethoden auf zwei Klassen notwendig. Es wird also das Separationsprinzip angewandt. Die Schablonenklasse T ist mit verschiedenen H-Objekten durch eine Kompositionsbeziehung verbunden. Unter einem H-Objekt verstehen wir dabei eine Instanz einer Klasse, welche die Methode H implementiert. Dazu wird in der Schablonenklasse T eine Referenzvariable verwendet, die auf ein bestimmtes H-Objekt verweist. Die H-Objekte können dann zur Laufzeit entsprechend ausgetauscht werden. Damit kann das Verhalten der Anwendung verändert werden, ohne die Anwendung neu zu starten. Das Unifikations- und das Separationsprinzip sind in Abbildung 5.11 dargestellt.

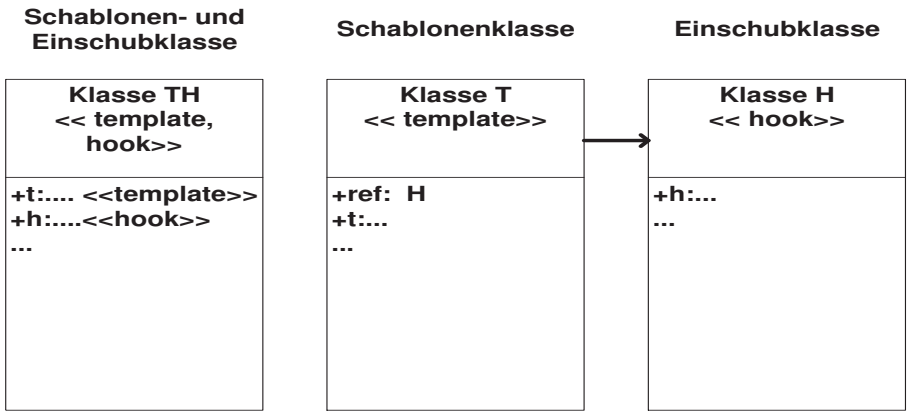

Abbildung 5.11: Unifikations-und Separationsprinzip [37]

Schablonen- und Einschubklassen können auch durch rekursive Komposition miteinander verknüpft werden. Das bedeutet, dass die Schablonenklasse T von der Einschubklasse H abgeleitet sein kann. Es besteht weiterhin die Möglichkeit, dass Einschub- und Schablonenklasse zusammenfallen.

Die Schablonenklasse T verwendet eine weitere Instanz von sich selbst zur Implementierung der Einschubklasse H. Die Einschubklasse H kann aber umgekehrt nicht von der Schablonenklasse abgeleitet werden, da Einschubklassen nur abstrakte Einschubmethoden besitzen. Die beschriebene Situation ist in Abbildung 5.12 dargestellt.

Die Entwicklung eines objektorientierten Rahmenwerks verlangt typischerweise viele Iterationen. Wir betrachten dazu das nachfolgende Beispiel.

Beispiel 5.4.3 (Iterative Rahmenwerkentwicklung) *Die im Rahmenwerk verwendeten Schablonenmethoden können zu restriktiv sein. Das Rahmenwerk ist dadurch zu wenig an konkrete Anwendungen anpassungsfähig. In diesem Fall sind die Anzahl der Schablonenmethoden zu reduzieren und weitere Einschubmethoden bereitzustellen. Umgekehrt kann es nach mehrfacher Anwendung eines*

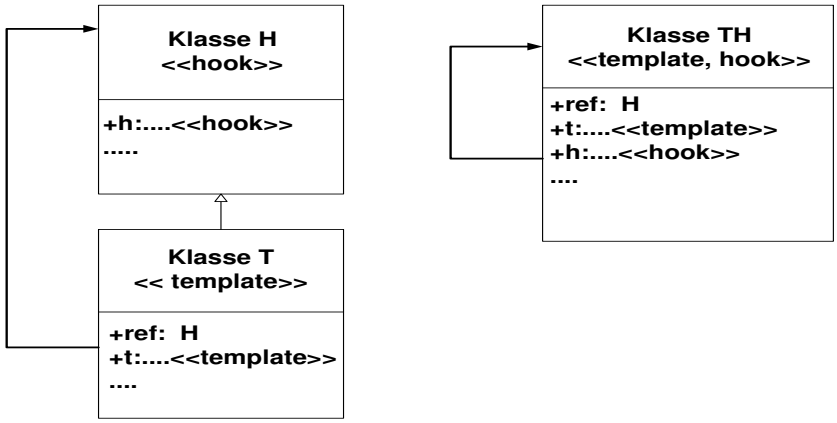

Abbildung 5.12: Rekursive Kombination [37]

objektorientierten Rahmenwerkes auch sinnvoll sein, weitere Schablonenmethoden im Rahmenwerk zu implementieren.

5.4.6 Konstruktionsprinzipien für komponentenbasierte Rahmenwerke

Im Gegensatz zu objektorientierten Rahmenwerken, bei denen mit der Hotspot-getriebenen Vorgehensweise eine Methodik zum Entwurf zur Verfügung steht, fehlt eine entsprechende Vorgehensweise bei komponentenbasierten Rahmenwerken.

Der Entwurf von komponentenbasierten Rahmenwerken kann dadurch verbessert werden, dass Komponenten gleich so entworfen und entwickelt werden, dass sie leicht an spezielle Anforderungen für eine konkrete Anwendung angepasst werden können. Aus derartigen Komponenten können dann im Prinzip leicht komponentenbasierte Rahmenwerke konstruiert werden. Da man aber die Anforderungen an ein komponentenbasiertes Rahmenwerk häufig beim Entwurf einzelner Komponenten noch nicht genau kennt, besteht die Gefahr, dass die einzelnen Komponenten mit zuviel Flexibilität ausgestattet werden.

In der Literatur werden Modifikationen des Hotspot-getriebenen Entwurfes für komponentenbasierte Rahmenwerke diskutiert [38]. Die Untersuchungen befinden sich aber noch in einem frühen Stadium, so dass von einem systematischen Vorgehen nicht gesprochen werden kann.

5.4.7 Beispiele für Rahmenwerke

Wir diskutieren an dieser Stelle ein objektorientiertes und ein agentenbasiertes Rahmenwerk, die zum Entwurf von betrieblichen Anwendungssystemen für die Produktionsdomäne verwendet werden können.

5.4.7.1 DéjàVu

DéjaVu ist ein in der Programmiersprache C++ implementiertes Rahmenwerk zum Bau von Ablaufplanungssystemen [5]. DéjàVu stellt abstrakte Klassen wie

- Auftrag,
- Los,
- Arbeitsgang,
- Arbeitsplan,
- Ressource

zur Verfügung. Arbeitspläne beinhalten die Vorschriften, aus denen Produkte hergestellt werden (vergleiche hierzu die Ausführungen in Abschnitt 6.1.2). Diese Klassen bilden den Kern von DéjàVu. Außerdem enthält DéjàVu Klassen für die Abbildung von zeitlichen Restriktionen. In DéjàVu wurde ein Standardablauf für die Interaktion von Anwendern mit einem Ablaufplanungssystem aufgenommen.

Bewertung von Ablaufplänen

DéjàVu bewertet Ablaufpläne unter Verwendung von sogenannten Softconstraints. Die Ablaufpläne werden dabei durch iterative Verfahren sukzessive verbessert. Ablaufplanungsanwendungen, die auf Basis dieses Rahmenwerks entwickelt werden, erlauben in starkem Maße die Interaktion mit dem Benutzer.

In DéjàVu wird die abstrakte Basisklasse „Constraint“ verwendet. Konkrete Constraints wie die Einschränkung „TardinessConstraint“ oder „SetupConstraint“ werden von dieser Klasse abgeleitet. Jedem Ablaufplan ist eine Liste mit Constraints zugeordnet. Wir betrachten dazu das nachfolgende Beispiel.

Beispiel 5.4.4 (Constraints) *Instanzen der Klasse „TardinessConstraint“ bewerten den durch den Ablaufplan vorgegebenen Fertigstellungstermin eines Loses, indem sie diesen Termin mit dem geplanten Fertigstellungstermin in Beziehung setzen. In Abhängigkeit vom Grad der Verspätung wird dem Ablaufplan eine Zahl aus dem Intervall* $[0, 1]$ *zugeordnet. Wenn ein Los eingeplant wird, erzeugt das Los entsprechende Einschränkungen, die dann in die Einschränkungsliste des Ablaufplans aufgenommen werden. Wenn umgekehrt ein Arbeitsgang auf einer Maschine ausgeführt wird, wird eine Maschinenbelegung durchgeführt. Diese Maschinenbelegung erzeugt dann ebenfalls Einschränkungen, die in die Einschränkungsliste aufgenommen werden.*

Die abstrakte Basisklasse „Constraint“ stellt Methoden zur Verfügung, die in der abstrakten Klasse nicht implementiert sind, aber vom DéjàVu-Kern zur Bewertung von Ablaufplänen benutzt werden. In abgeleiteten Klassen müssen dann die entsprechenden Methoden implementiert werden, damit Objekte instanziiert werden können.

DéjàVu ist in der Lage, Ablaufpläne iterativ zu verbessern. Dabei kommen

lokale Suchverfahren zur Anwendung. Im Folgenden soll das Vorgehen anhand von **Tabu-Search** [35, 4] dargestellt werden. Tabu-Search geht von einem gegebenen initialen zulässigen Ablaufplan aus. In einem ersten Schritt werden zunächst alle Nachbarschaften dieses Ablaufplans untersucht. Wir betrachten dazu das folgende Beispiel.

Tabu-Search

Beispiel 5.4.5 (Nachbarschaft) *Wir erhalten eine spezielle Nachbarschaft, indem zwei Lose auf einer festen Maschine oder Lose zwischen zwei Maschinen im Ablaufplan vertauscht werden.*

Die unterschiedlichen Nachbarschaften werden mit einer vorgegebenen Zielfunktion bewertet. Die Nachbarschaft mit dem besten Zielfunktionswert wird ausgewählt. Der Ablaufplan wird entsprechend der Nachbarschaft modifiziert. Der Übergang von einem Ablaufplan zu einem anderen Ablaufplan entsprechend der Auswahl des besten Nachbarn, auch wenn dieser eine Verschlechterung darstellt, wird als Zug bezeichnet. Zur Vermeidung des Erreichens von lokalen Optima wird eine Tabu-Liste eingeführt. Die Tabu-Liste speichert die letzten n durchgeführten Züge. Nachdem ein bestimmter Zug in der Tabu-Liste gespeichert wurde, kann dieser Zug in den nächsten n Verfahrensiterationen nicht mehr gewählt werden.

Nachbarschaftssuche

Das eben beschriebene Tabu-Search-Verfahren wird als Schablonenmethode implementiert. Die Schablonenmethode verwendet Einschubobjekte der Planhierarchie (Klasse „Schedule“) und der Hierarchie der Operatoren (Klasse „ScheduleTask“). Das nachfolgend skizzierte Verfahren „ImproveSchedule“ deutet das gewählte Vorgehen in einer C++-artigen Notation an [5]:

```
ImproveSchedule(void)
{
        tasklist = schedule->GenerateTasks();
        bestTask = taskList->First();
        while(not taskList->Empty())
        {
            task = taskList->First();
            evaluation = task->Evaluate();
            if (evaluation > bestEvaluation)
            {
               bestTask = task;
               bestEvaluation = evaluation;
            }
            taskList->Remove(task);
        }
        task->Do();
}.
```

Die Methode „GenerateTasks“ ist eine Einschubmethode der Schedule-Hierarchie, während die Methoden „Evaluate“ und „Do“ Einschubmethoden der ScheduleTask-Hierarchie darstellen.

In der Schedule-Hierarchie werden Ablaufplantypen von DéjàVu unterschieden. Unterschiedliche Ablaufplantypen erzeugen durch die ihnen zugeordnete Methode „GenerateTask“ auch unterschiedliche Listen mit auszuführenden Arbeitsgängen. Im nachfolgenden Beispiel werden unterschiedliche Ablaufplantypen für Chargen- und Montageprozesse betrachtet.

Beispiel 5.4.6 (Ablaufplantyp) *In der Chargenfertigung ist es notwendig, festzulegen, welche Lose zu einer Charge zusammengefasst werden sollen. Bei Montageprozessen müssen verschiedene Halbfabrikate termingerecht in den jeweils notwendigen Mengen zur Verfügung stehen.*

Konkrete Operatoren für Tabu-Search werden von der abstrakten Basisklasse „SchedulingTask“ abgeleitet. Diese Basisklasse schreibt eine Schnittstelle vor, die von jeder abgeleiteten Klasse implementiert werden muss. Die Klasse wird unter Verwendung des Command-Design-Patterns [12] entwickelt. Dieses Muster wird verwendet, da die zu implementierenden Operatoren für Tabu-Search vom manipulierten Objekt abhängig sind. Diese Situation liegt bei den Operatoren für Tabu-Search vor, da unterschiedliche Ablaufplantypen verschiedene Operatoren voraussetzen. Die Manipulationsmöglichkeiten der Operatoren sind aber in allen Fällen gleich. Die entsprechende Klasse „SchedulingTask“ sieht in C++-artiger Notation wie folgt aus [5]:

```
class SchedulingTask: public Task
{
        virtual double Evaluate(void);
        virtual bool do(void) = 0;
        virtual bool Redo(void) = 0;
        virtual bool Undo(void) = 0;
        virtual SchedulingTask* InverseTask(void) const = 0;
}.
```

Die beiden Methoden „Evaluate“ und „Do“ werden innerhalb von „Improve-Schedule“ bereits verwendet. Die Methode „Redo“ wiederholt „Do“. „Undo“ macht den zuletzt getätigten Zug rückgängig. Die Methode „InverseTask“ stellt die inverse Operation für eine Instanz von SchedulingTask dar. Wir betrachten dazu das folgende Beispiel.

Beispiel 5.4.7 (InverseTask) *Wenn SchedulingTask eine Verschiebung von Los i auf Maschine m im Ablaufplan um zwei Positionen nach hinten ausführt, ist InverseTask für die Verschiebung von i um zwei Positionen nach vorne verantwortlich.*

Interaktion mit dem Benutzer

DéjàVu regelt weiterhin die Interaktion zwischen Benutzer und Ablaufplan. Benutzer haben die Möglichkeit, durch das Ablaufplanungssystem vorgeschlagene Ablaufpläne interaktiv zu modifizieren. Der Benutzer kann die Vorschläge des Ablaufplanungssystems einerseits jederzeit verwerfen, andererseits kann das Ablaufplanungssystem jederzeit mit einer neuen Optimierung der Ablaufpläne starten und somit insbesondere Änderungen der Benutzer in den Plänen geeignet berücksichtigen.

Zur Unterstützung der Interaktion von Benutzer und vorgeschlagenen Ablaufplänen wird durch die Architektur von DéjàVu eine enge Kopplung zwischen Benutzerschnittstelle und der Darstellung von Ablaufplänen unterstützt. Eine auf Basis von DéjàVu entwickelte Ablaufplanungssoftware zeigt dem Benutzer jede durch den Ablaufplan verletzte Einschränkung an. Außerdem können Ablaufpläne aus Ressourcen- und Losperspektive jeweils betrachtet und entsprechend modifiziert werden. Die Verwendung unterschiedlicher Sichten wird durch das Model-View-Controller-Entwurfsmuster (vergleiche [12] und die Ausführungen in Abschnitt 2.2.3) ermöglicht.

Die Modifikation von Ablaufplänen wird durch das Observer-Pattern [12] unterstützt. Das Muster schlägt vor, zwischen dem Subjekt einer Veränderung und einem Beobachter zu unterscheiden. Objekte melden sich bei einem Subjekt als Beobachter an. Bei einer Veränderung eines Subjekts schickt dieses eine Benachrichtigung an alle Beobachter. Wir betrachten dazu das folgende Beispiel.

Beispiel 5.4.8 (Observer-Pattern) *Das Fenster, in dem die Verletzung von Einschränkungen in einem Ablaufplan angezeigt wird, ist als Beobachter beim entsprechenden Ablaufplanobjekt angemeldet. Wenn der Ablaufplan verändert wird, wird das Fensterobjekt benachrichtigt und kann dann entsprechend den Fensterinhalt verändern.*

Nutzung von Entwurfsmustern

Am Beispiel von DéjàVu ist deutlich zu erkennen, dass Entwurfsmuster wesentlich dazu beitragen, die durch ein objektorientiertes Rahmenwerk zur Verfügung gestellte Architektur zu beschreiben. Auf diese Tatsache wurde bereits in Abschnitt 2.2.1 hingewiesen, indem dort gezeigt wurde, dass Muster strukturelle Integritätsbedingungen für die einzelnen Modellebenen einer Architektur darstellen.

5.4.7.2 ManufAg

agentenbasiertes Rahmenwerk

ManufAg stellt ein agentenbasiertes Rahmenwerk zur Steuerung komplexer Produktionssysteme dar [31]. Komplexe Produktionsprozesse [33] sind durch einen zeitlich veränderlichen Produktmix, das gleichzeitige Auftreten unterschiedlicher Prozesstypen (z. B. Batchprozesse, bei denen mehrere Lose gleichzeitig auf einer Maschine bearbeitet werden), vorgeschriebene Kundenendtermine, eine hohe Komplexität der Arbeitspläne, reihenfolgeabhängige Umrüstzeiten, parallele Maschinen sowie sekundäre Ressourcen gekennzeichnet. Externe Störungen

in Form von Veränderungen von Kundenendterminen und internen Störungen durch Maschinenausfälle sind typisch.

Gegenwärtig existieren verschiedene Systemprototypen, welche die in Abschnitt 2.2.3 eingeführte PROSA-Referenzarchitektur anwenden (vergleiche z.B. das ManAge-System [15, 18]). Eine Unterstützung der Entwicklung von Multi-Agenten-Systemen auf Basis von PROSA steht aber erst durch ManufAg zur Verfügung.

Designkriterien

Wir beschreiben zunächst Kriterien, die für das Design des Rahmenwerks wesentlich sind:

- **Anforderungen, die sich aus der Produktionssteuerungsdomäne ableiten**: Wir benötigen keine generische Architektur, da das vorgeschlagene Rahmenwerk auf die Produktionssteuerungsdomäne abzielt. Es ist notwendig, ein Rahmenwerk zu entwickeln, das die Abbildung unterschiedlicher Organisationsformen von Produktionssystemen zusammen mit verschiedenen Produktionssteuerungsparadigmen ermöglicht.

- **Flexibilität**: Unter Flexibilität verstehen wir die Möglichkeit, dass ein System sich an unterschiedliche Situationen anpassen kann. Ein bestimmter Grad an Flexibilität, im Wesentlichen durch die prinzipielle Möglichkeit autonomer Aktionen verursacht, ist für Multi-Agenten-Systeme typisch [48]. Wir betrachten im Fall von ManufAg speziell Flexibilität im Kontext der Domäne komplexer Produktionssysteme.

- **Skalierbarkeit**: Unter Skalierbarkeit eines Produktionssteuerungssystems, das auf Basis von ManufAg entwickelt wurde, verstehen wir die Möglichkeit, die Anzahl der zu steuernden Maschinen und Lose im Basissystem erhöhen zu können, ohne nennenswerte Laufzeiteinbußen hinnehmen zu müssen.

- **Wiederverwendbarkeit**: Softwarewiederverwendung wird durch die Entwicklung einer generischen Infrastruktur, durch die Bereitstellung einer Menge von Agentenklassen und -rollen sowie von entsprechenden Interaktionsprotokollen erreicht.

- **Kompatibilität mit existierenden Standards**: Das ManufAg-Rahmenwerk muss existierenden Standards im Bereich der Agententechnologie so weit wie möglich folgen (vergleiche dazu [17, 24]). Außerdem müssen zeitgemäße Standards für Middleware eingehalten werden.

Nach der Darstellung der Entwurfskriterien stellen wir Infrastrukturaspekte sowie die Agentenarchitektur vor. Außerdem gehen wir auf die Behandlung unterschiedlicher Organisationsformen von Produktionssystemen ein.

Infrastruktur

Die durch das Rahmenwerk unterstützte Infrastruktur orientiert sich an der von FIPA vorgeschlagenen abstrakten Architektur [24], um Kompatibilität zu

existierenden Standards sicherzustellen. Eine Infrastruktur für Multi-Agenten-Systeme hat die folgenden Aufgaben zu erfüllen:

1. dynamisches Erzeugen von Agenten,
2. dynamisches Zerstören von Agenten,
3. Sicherstellung der Kommunikationsfähigkeit von Agenten,
4. Bereitstellen eines Adressen- und Dienstverzeichnisses für alle Agenten.

Diese Dienste sind dem FIPA-Standard für agentenbasierte Softwaresysteme (vergleiche [34, 24] für eine entsprechende Einordnung) entnommen. Eine Laufzeitumgebung für ein Multi-Agenten-System hat die folgenden Bestandteile:

1. Agent-Communicator,
2. Agent-Service-Directory,
3. Agent-Management-System,
4. Agent-Container.

Diese Bestandteile des Multi-Agenten-Systems stellen Dienste zur Verfügung, die von den Agenten zur Ermittlung von Informationen über andere Agenten und zur Kommunikation und Interaktion mit anderen Agenten benutzt werden können [15].

Der **Agent-Communicator** kapselt Kommunikationsfähigkeiten der Agenten. Jede Kommunikation zwischen Agenten wird über den Agent-Communicator abgewickelt.

Das **Agent-Service-Directory** dient zur Registrierung der Fähigkeiten der Agenten, die sie als Dienste anderen Agenten zur Verfügung stellen. Ein Agent-Service-Directory ist ein Dienste- und Adressenverzeichnis. Agenten fragen beim Agent-Service-Directory der Laufzeitumgebung nach Informationen über Agenten nach, mit denen sie interagieren wollen. Falls die gesuchte Information nicht verfügbar ist, versucht das Agent-Service-Directory mit anderen Agent-Service-Directories der verbleibenden Laufzeitumgebungen Kontakt aufzunehmen und diese Information zu ermitteln. Auf diese Art und Weise wird der Einsatz eines zentralen Agent-Service-Directories als zentraler Informationspunkt im verteilten System vermieden. Das Agent-Service-Directory ist für das Management der hierarchischen und lokalen Organisation der Agenten innerhalb des Multi-Agenten-Systems verantwortlich.

Laufzeitumgebung

Jede Laufzeitumgebung eines Multi-Agenten-Systems benötigt zum Management des Lebenszyklus der Agenten ein **Agent-Management-System (AMS)**. Das Agent-Management-System dient zur Erzeugung und umgekehrt zum Entfernen von Agenten, wenn diese nicht länger benötigt werden.

Ein **Agent-Container** ist der letzte wichtige Bestandteil einer Laufzeitumgebung. Der Container enthält Informationen über alle in der Laufzeitumgebung aktiven Agenten. In Abbildung 5.13 sind die wesentlichen Bestandteile einer Laufzeitumgebung dargestellt.

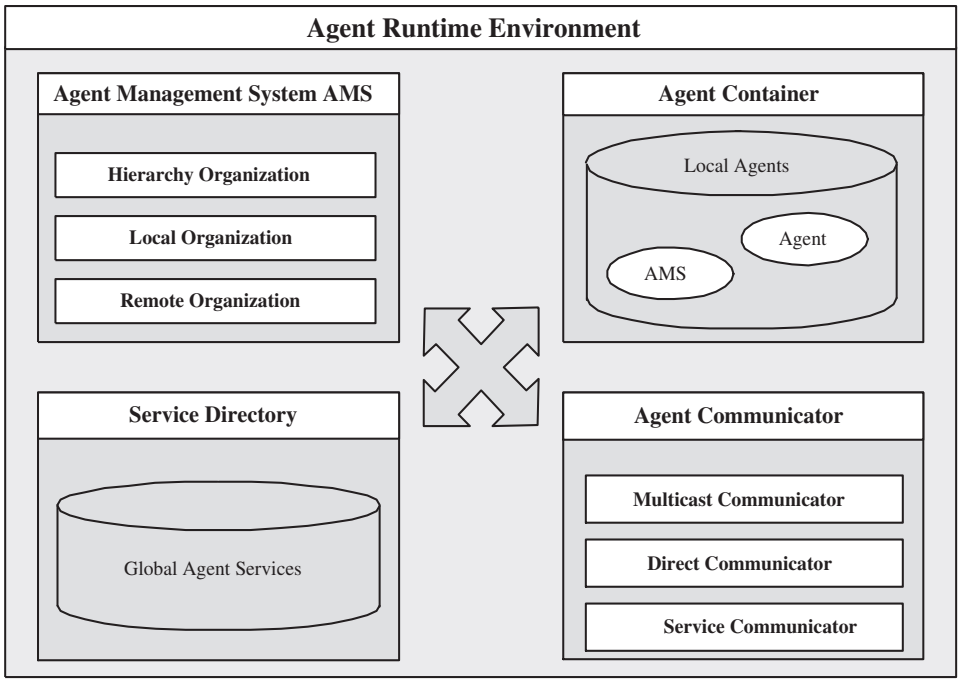

Abbildung 5.13: Multi-Agenten-System-Infrastruktur

Kommunikation zwischen autonomen Agenten ist eine der wichtigen Eigenschaften von Multi-Agenten-Systemen, um Kooperations- und Koodinationsfähigkeiten geeignet abzubilden. Der für die Kommunikation verantwortliche Teil des Rahmenwerks erlaubt es, die abzubildende organisatorische Struktur geeignet zu berücksichtigen. Dazu werden Kommunikationsrechte im Multi-Agenten-System eingeführt, da anstelle des von FIPA vorgeschlagenen globalen Directories als möglichem Ausfallpunkt ein Peer-to-Peer-Announcement-System implementiert ist. Die durch das Rahmenwerk zur Verfügung gestellten Kommunikationsrechte sind in Tabelle 5.9 zusammengetragen.

Die Kommunikationsrechte bilden eine Relation R_{comm} auf der Menge der Agenten. Zwei Agenten A_1 und A_2 stehen miteinander in Beziehung, wenn A_1 mit A_2 kommunizieren kann. R_{comm} ist symmetrisch, d.h., wenn Agent A_1 mit A_2 kommunizieren kann, dann auch Agent A_2 mit A_1. Die Relation R_{comm} ist aber nicht transitiv.

Übungsaufgabe 5.13 (Transitivität) *Begünden Sie die fehlende Transitivität bei der Agentenkommunikation.*

Tabelle 5.9: Kommunikationsrechte der Agenten

Kommunikationsrechte	Beschreibung
öffentlich	Ein Agent kann mit allen öffentlichen Agenten kommunizieren und bietet seine Dienste allen anderen verbundenen Laufzeitumgebungen an.
privat	Der Agent kann ausschließlich mit Agenten, die sich in seiner Laufzeitumgebung befinden, kommunizieren.
hierarchisch	Der Agent ist innerhalb einer Hierarchie organisiert. Er kann lediglich mit seinen Eltern- und Kindagenten kommunizieren.
Hierarchieebenen	Der Agent kann mit anderen Agenten auf seiner Hierarchieebene zusätzlich zu den eigenen Eltern- und Kindagenten kommunizieren.
Cluster	Das Multi-Agenten-System kann in mehrere Cluster zerlegt werden. Die Kommunikation in den einzelnen Clustern kann voneinander entkoppelt werden.
kombinierte Rechte	Verschiedene Kommunikationsrechte können miteinander kombiniert werden. Beispielsweise ist es möglich, Clusterrechte mit hierarchischen Kommunikationsrechten zu kombinieren.

Das Rahmenwerk benutzt den administrativen Teil der FIPA-ACL-Nachrichten und die grundlegenden Sprechakte, die ebenfalls von FIPA vorgeschlagen werden. Das Agenten-Service-Directory ist für die Herstellung von Verbindungen zwischen unterschiedlichen Remote-Laufzeitumgebungen verantwortlich. Falls die Laufzeitumgebungen einander kennen, können die Agenten entsprechend ihrer Kommunikationsrechte auf diesen Laufzeitumgebungen miteinander kommunizieren und kooperieren. Die Kommunikationsrechte werden ebenfalls zur Veröffentlichung der Dienste der Agenten innerhalb des Multi-Agenten-Systems genutzt. Diese Informationen werden in lokalen Blackboards für jede Laufzeitumgebung vorgehalten. Die Agenten können diese Blackboards durchsuchen, um Agenten zu finden, die gewünschte Dienste anbieten.

Ein Agent ist entsprechend Definition 4.1.1 eine autonome Einheit, die bestimmte Aktionen im Auftrag eines anderen Agenten oder Anwenders ausführt. Agenten beziehen außerdem Informationen von ihrer Umwelt und versuchen, implizite oder explizite Ziele zu erreichen. Ein Agent besteht im Wesentlichen aus einer Menge von Verhaltensausprägungen und Aktionen, welche die Reaktionen des Agenten in bestimmten Situationen bestimmen. Wir betrachten das folgende

Beispiel für Situationen.

Beispiel 5.4.9 (Situation) *Eingehende Nachrichten oder bestimmte Umweltereignisse wie Maschinenausfälle haben Einfluss auf die Reaktionen der Agenten.*

Agenten-verhalten

Das Verhalten der Agenten bildet die Basis für die Abbildung von Reaktivität und Proaktivität. Die Interaktionen zwischen Agenten werden hauptsächlich unter Verwendung von Verhaltensausprägungen implementiert. Sie sind für den Nachrichtenaustausch zwischen den Agenten in Abhängigkeit vom Zweck der Interaktion verantwortlich.

Ein Agent stellt Dienste zur Verfügung, die von anderen Agenten genutzt werden können. Ein Dienst stellt eine Hülle für eine Menge von Aktionen dar, die von dem den Dienst anbietenden Agenten ausgeführt werden können. Jeder Agent besitzt Ziele. Die Aktionen des Agenten werden zur Erfüllung der Ziele eingesetzt. Dieser zielorientierte Ansatz wird verwendet, um Proaktivität zu modellieren. Proaktivität bedeutet in diesem Zusammenhang, dass ein Agent selber entscheiden kann, ob er eine bestimmte Aktion zu einem bestimmten Zeitpunkt oder unter bestimmten Umständen ausführen will oder nicht.

Ein Ziel wird als Liste gegenwärtiger oder zukünftiger Aktionen abgebildet. Die Liste wird als **Action-Item-List (AIL)** bezeichnet. Die Liste beschränkt sich auf Aktionen, die zu bestimmten Umweltereignissen zugehörig sind bzw. innerhalb eines bestimmten Zeithorizontes stattfinden. Jeder Agent kann seine eigene AIL zur Laufzeit verändern. Dadurch ist es möglich, die Ausführung einer kritischen Aktion innerhalb des Zeithorizontes zu überwachen bzw. eine Aktion zeit- oder ereignisgesteuert zu initiieren. Die im Rahmenwerk abgebildete Proaktivitätsarchitektur ermöglicht es, Agenten zu implementieren, die auf Ereignisse in der Umwelt geeignet reagieren können. Gleichzeitig sind die Agenten in der Lage, Ereignisse zu interpretieren und selbständig Aktionen der AIL zu initiieren.

Agenten-architektur

In Abbildung 5.14 zeigen wir die Architektur eines einzelnen Agenten als UML-Klassendiagramm.

Um Flexibilität in Multi-Agenten-Systemen, die auf Basis von ManufAg entwickelt werden, zu erhalten, werden den einzelnen Agenten Rollen zugewiesen.

Rollen

Jeder Agent kann im zeitlichen Verlauf unterschiedliche Rollen einnehmen [32]. Wir definieren den Rollenbegriff wie folgt.

Definition 5.4.4 (Rolle) *Klassen zur Kapselung des normativen Verhaltensrepertoires eines Agenten werden als Rolle des Agenten bezeichnet.*

Eine Verhaltensausprägung wird durch Zustände und Zustandstransitionen beschrieben. Eine neue Rolle sorgt dafür, dass der Agent mit neuen Eigenschaften in Form von Diensten, Zielen und Aktionen versorgt wird. Unter Umständen ist es notwendig, dass eine neue Verhaltensausprägung den Agenten zugewiesen wird. Diese Verhaltensausprägung definiert, warum, wann und wie ein Agent

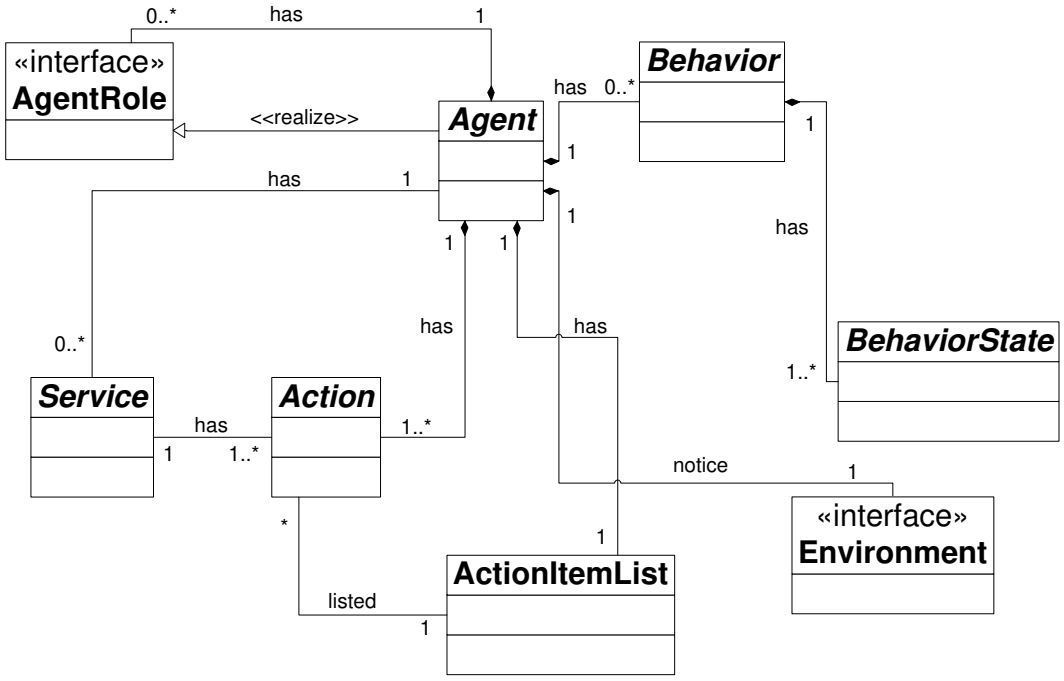

Abbildung 5.14: UML-Klassendiagramm für die Architektur eines einzelnen Agenten

auf spezielle Ereignisse reagiert. Die Flexibilität des verwendeten Rollenkonzepts gibt Multi-Agenten-Systemen, die auf Basis von ManufAg entwickelt wurden, die Möglichkeit, sich an unterschiedliche Situationen innerhalb der Produktionssteuerungsdomäne anzupassen. Das implementierte Rollenkonzept dient der Realisierung des in Abschnitt 5.4.5 eingeführten Separationsprinzips. Durch die unterschiedlichen Rollen werden somit die erforderlichen Einschubmethoden zur Verfügung gestellt.

Um die Online-Verfügbarkeit der aktiven Agenten sicherzustellen, wird ein Multi-Threading-Ansatz für die Implementierung der Agenten und der Verhaltensausprägungen verwendet. Dieser Ansatz erlaubt die gleichzeitige Ausführung mehrerer Agenten und/oder Verhaltensausprägungen. Ein einzelner Agent kann gleichzeitige Anfragen unterschiedlicher Agenten behandeln und ebenso auf gleichzeitige Ereignisse reagieren. Wir betrachten dazu das folgende Beispiel.

Beispiel 5.4.10 (Konkurrierende Ausführung mehrerer Agenten) *Ein Agent kann mit zwei weiteren Agenten im Rahmen eines Kontraktnetzansatzes verhandeln und gleichzeitig einen Dienst eines dritten Agenten nachfragen.*

MAS-Architektur

Die dargestellte Architektur von Einzelagenten bildet die Basis für die Entwicklung einer Multi-Agenten-System-Architektur für die Produktionssteuerungsdomäne. Dabei kommt die bereits in Abschnitt 2.2.3 dargestellte PROSA-Referenzarchitektur [47] zur Anwendung. Das Rahmenwerk muss die Abbildung

von unterschiedlichen Organisationstypen wie Fließ- und Werkstattfertigung sowie die Realisierung von verschiedenen Produktionssteuerungsparadigmen ermöglichen. Wir betrachten das nachfolgende Beispiel für mögliche Produktionssteuerungsverfahren.

Beispiel 5.4.11 (Verfahren zur Produktionssteuerung) *Ansätze, die auf Prioritätsregeln basieren [18], verteilte verhandlungsbasierte Verfahren zur Ressourcenallokation [44] oder stärker zentralisierte Ablaufplanungsverfahren wie die Shifting-Bottleneck-Heuristik [30] können als Produktionssteuerungsverfahren im Rahmen von Multi-Agenten-Systemen herangezogen werden.*

Agentenhierarchie

Wir verwenden die PROSA-Referenzarchitektur, um diese Abbildung zu ermöglichen. Als Ergebnis des Entwurfsprozesses erhalten wir eine Architektur, welche die Abbildung von Agentenhierarchien ermöglicht. Durch die Hierarchie können unterschiedliche Typen von Produktionssystemen geeignet abgebildet werden. PROSA schlägt die Verwendung von Entscheider- und Dienstagenten vor. Entscheider- und Dienstagenten können als Ausgangspunkt für eine weitere Anpassung an spezielle Produktionssteuerungsprobleme herangezogen werden. Dienstagenten dienen der Implementierung von unterschiedlichen Produktionssteuerungsverfahren.

Basierend auf der beschriebenen Architektur für Einzelagenten und auf PROSA schlägt ManufAg eine hybride Architektur vor, die sowohl reaktive als auch hybride Eigenschaften aufweist. Die bekannte hybride Architektur InteRRaP [29] verwendet drei Schichten innerhalb jedes Agenten. Die erste Schicht behandelt reaktive Situationen. Die zweite Schicht ist eine Planungsschicht, während die dritte Schicht eine Kooperationsschicht für Agenteninteraktionen darstellt. Der in ManufAg implementierte Entscheider- und Dienstagentenansatz von PROSA realisiert eine hybride Schichtenarchitektur, die durch die Dienstagenten eine separate Planungsschicht und durch die Entscheideragenten eine reaktive und proaktive Schicht zur Verfügung stellt.

PROSA und der vorgeschlagene Schichtenansatz sind der Schlüssel, um Skalierbarkeit von auf ManufAg basierenden Produktionssteuerungsanwendungen sicherzustellen. Rechenintensive Ablaufplanungsverfahren können unterschiedlichen Dienstagenten zugeordnet werden, die wiederum auf Computern unterschiedlicher Leistungsfähigkeit ausgeführt werden können. Unter Verwendung dieses Ansatzes ist es leichter, das Produktionssteuerungssystem an unterschiedliche Lastsituationen anzupassen. Eine gute Laufzeitperformance ist die Folge des gewählten Vorgehens.

Entscheideragenten

Wie in PROSA vorgeschlagen, wird auch in ManufAg zwischen Entscheider- und Dienstagenten unterschieden. Entscheider- und Dienstagenten sind unabhängig von der Organisationsform des Multi-Agenten-Systems, d.h., die beiden Agententypen sind in sowohl heterarchisch als auch hierarchisch organisierten Multi-Agenten-Systemen zu betrachten.

Entscheideragenten stellen die nachfolgende Funktionalität zur Verfügung:

Tabelle 5.10: Verhalten von Entscheideragenten

Verhalten	Beschreibung
Prepare_Decision_Making	Modellierung der Vorbereitungsphase einer bestimmten Entscheidung
Make_Decision	Modellierung der Entscheidungsfindungsphase
Inform_DM_Agent	Information anderer Entscheideragenten über bestimmte Entscheidungen
Start_Staff_Agent	Aktivierung eines anderen Dienstagenten durch den Entscheideragenten
Get_Staff_Agent_Result	Entscheideragent fordert Dienstagenten auf, Ergebnisse von Berechnungen zu übertragen
Request_DM_Agent_Service	Entscheideragent fragt Dienste eines anderen Entscheideragenten nach

1. Vorbereiten von Entscheidungen wie situationsabhängige Wahl von Parametern für die Nutzung von Dienstagentenfunktionalität, Auswahl einer geeigneten Dienstagentenfunktionalität,
2. eigentliche Entscheidungsfindung,
3. Information anderer Entscheideragenten über das Ergebnis der Entscheidung,
4. Aktivierung anderer Entscheideragenten,
5. zeitgetriebene Abfrage der Ergebnisse von Dienstagenten,
6. Anfrage bei anderen Entscheideragenten bezüglich bestimmter Dienste,
7. sachgerechte Behandlung von Problemen und Ausnahmen, die während des Entscheidungsfindungsprozesses auftreten.
8. Überprüfung, ob in der AIL relevante Ereignisse enthalten sind.

Aus der zu realisierenden Funktionalität leiten sich Verhaltensklassen für Entscheideragenten ab. Die zugehörigen Verhaltensausprägungen sind in Tabelle 5.10 dargestellt.

Dienstagenten

Dienstagenten haben die nachfolgenden Aufgaben zu lösen:

1. Problemlösung vorbereiten durch Berechnung interner Parameter des Lösungsverfahrens, Versorgung des Lösungsverfahrens mit Daten und Parametern,

Tabelle 5.11: Verhalten von Dienstagenten

Verhalten	Beschreibung
Prepare_Solution	Modellierung der Vorbereitungsphase einer bestimmten Lösungsaktivität
Parameterize_Algorithm	Parametrisierung eines Lösungsalgorithmus mit intern oder extern bestimmten Parametern
Solve_or_Interrupt	Ermittlung von zulässigen Lösungen für das Problem, an dem der Dienstagent arbeitet, Beendigung der Lösungssuche durch Entscheideragenten möglich
Communicate_Solution	Information des Entscheideragenten, der die Problemlösung initiiert hat, über die ermittelte Lösung

2. Ermittlung interner Parameter der Problemlösungsmethode,
3. Versorgung der Problemlösungsmethode mit Daten und Parametern,
4. Lösung des Problems,
5. Abbruch eines Lösungsvorganges zu bestimmten Zeitpunkten oder infolge bestimmter Ereignisse,
6. Information anderer Agenten über das Ergebnis der Berechnung,
7. Behandlung von Ausnahmen während der Problemlösungsvorbereitung und der eigentlichen Problemlösung.

Die in Tabelle 5.11 dargestellten Verhaltensausprägungen für Dienstagenten lassen sich aus der zu realisierenden Funktionalität ableiten.

Interaktionen zwischen Entscheider- und Dienstagenten

Die Interaktionen zwischen Entscheider- und Dienstagenten können generisch beschrieben werden. Die Aktionen eines einzelnen Agenten können durch Verhaltenszustände beschrieben werden. Übergänge von einem Zustand zum nächsten sind notwendig, um eine bestimmte Aktion auszuführen. Die Zustandsübergänge werden durch Ereignisse ausgelöst. Die einzelnen Zustände werden nicht im Detail erläutert. Die Interaktionen werden lediglich durch UML-Sequenzdiagramme dargestellt.

In Abbildung 5.15 und 5.16 sind die Interaktionen zwischen Entscheider- und Dienstagenten aus Entscheideragentensicht dargestellt. Abbildung 5.16 dient der Fortsetzung von Abbildung 5.15. Die Statusänderungen der Verhaltensobjekte werden durch Selbstaufrufe der Verhaltensobjekte abgebildet. Wir zeigen ebenfalls die verwendeten Sprechakttypen der FIPA-ACL.

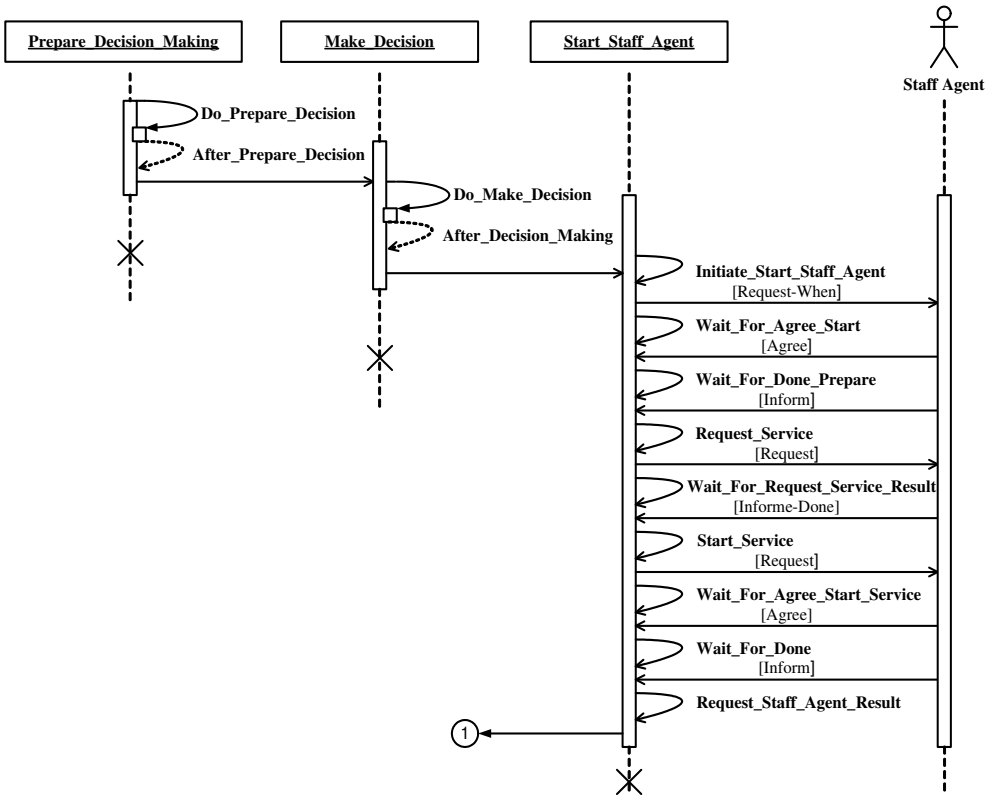

Abbildung 5.15: Sequenzdiagramm für das Zusammenwirken von Entscheider- und Dienstagenten aus Entscheideragentensicht (Teil I)

In Abbildung 5.17 ist die Interaktion zwischen Entscheider- und Dienstagenten aus Sicht eines Dienstagenten als UML-Sequenzdiagramm dargestellt.

Die bereitgestellten Muster für die Agenteninteraktionen stellen weitere Beispiele für Schablonenmethoden dar. Durch spezielle Einschubmethoden kann dann erreicht werden, dass unterschiedliche Lösungsverfahren innerhalb der auf ManufAg basierenden Multi-Agenten-Systeme Anwendung finden.

Organisationsformen

Wir beschreiben an dieser Stelle, wie unter Verwendung von ManufAg erstellte Multi-Agenten-Systeme organisiert sein können. Es werden

- föderierte,
- hierarchische,
- clusterartige,
- hybride

Organisationsformen betrachtet. **Föderierte Organisationsformen** in Multi-Agenten-Systemen sind dadurch gekennzeichnet, dass keinerlei zentrale

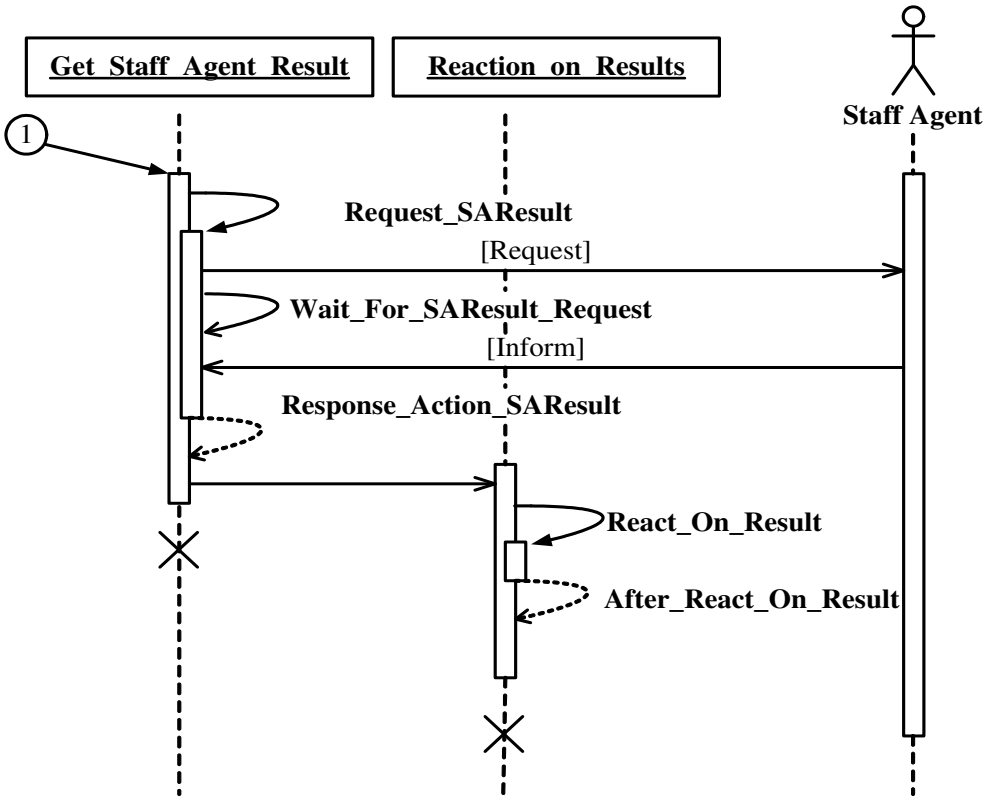

Abbildung 5.16: Sequenzdiagramm für das Zusammenwirken von Entscheider- und Dienstagenten aus Entscheideragentensicht (Teil II)

Steuerung vorliegt. Verhandlungen auf Basis des Kontraktnetzes sowie unter Verwendung von Auktionen sind häufig angewandte Problemlösungsstrategien in Multi-Agenten-Systemen, die keinerlei zentrale Steuerung voraussetzen. Die Unterstützung von föderierten Organisationsformen ist somit wesentlich für ein agentenbasiertes Rahmenwerk. Typischerweise ist das Verhalten von Systemen, die aus vielen Agenten mit einfacher Funktionalität bestehen und keine zentrale Steuerung beinhalten, schwer vorauszusagen. Aus diesem Grund ist es häufig nicht sinnvoll, ausschließlich föderierte Organisationsformen innerhalb eines Multi-Agenten-Systems zuzulassen.

Um ein vorhersagbares, unter globalen Gesichtspunkten günstiges Systemverhalten zu erreichen, werden **hierarchische Agentenorganisationen** eingeführt [28, 41]. Auf Basis der PROSA-Referenzarchitektur können baumartige Agentenhierarchien aufgebaut werden. Wenn eine hierarchische Organisation der Agenten verwendet wird, ist diese nicht notwendigerweise zwingend für alle Agenten innerhalb des Multi-Agenten-Systems erforderlich. Ein hierarchisch aufgebautes Multi-Agenten-System besteht typischerweise aus zwei Klassen von Agenten. **Hierarchische Agenten** werden entweder

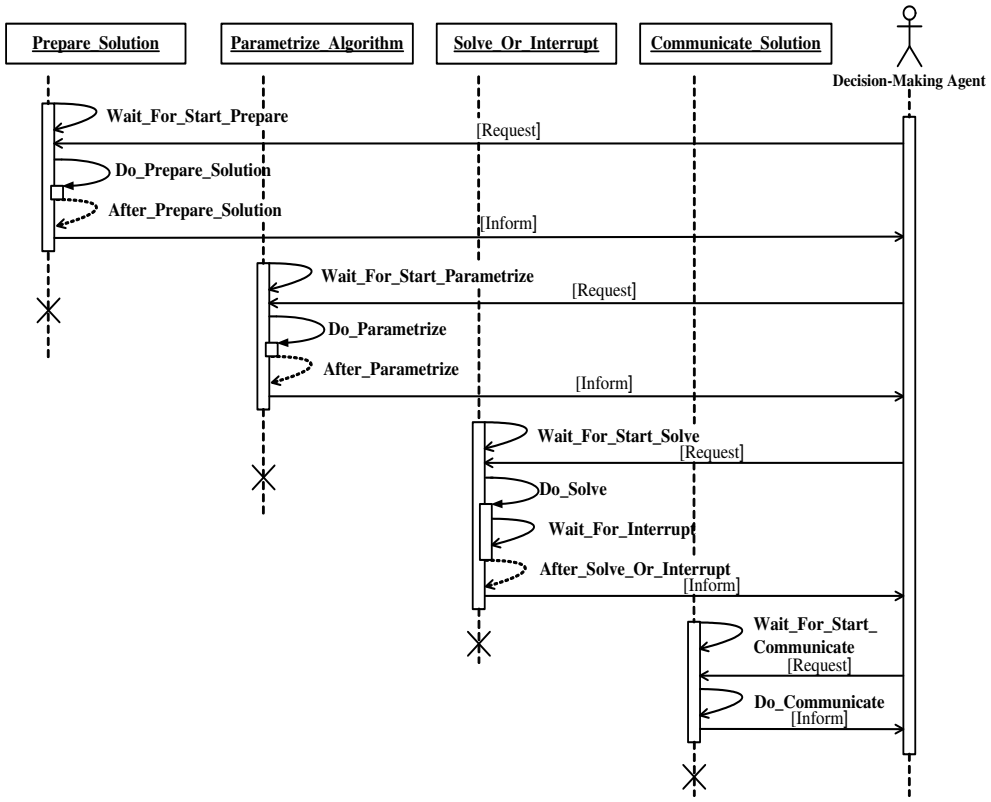

Abbildung 5.17: Sequenzdiagramm für das Zusammenwirken von Entscheider- und Dienstagenten aus Dienstagentensicht

als Eltern- oder als Kindagenten oder in beiden Funktionen innerhalb einer Agentenhierarchie verwendet. Elternagenten modifizieren gegebenenfalls ihre Entscheidungen, wenn sie im Rahmen von Rückkopplungsprozessen bzw. iterativen Abstimmungsprozessen die Lösungen der zu ihnen zugehörigen Kindagenten berücksichtigen. **Hierarchiefreie Agenten** sind nicht Bestandteil irgendeiner Hierarchie. Hierarchiefreie Agenten können mit Agenten auf einer beliebigen Hierarchieebene kommunizieren. Wir betrachten dazu das nachfolgende Beispiel.

Beispiel 5.4.12 (Hierarchiefreie Agenten) *Agenten, die physische Entitäten vertreten, die sich durch das Produktionssystem bewegen, sind typischerweise hierarchiefrei. So sind Lose in einem Produktionssystem hierarchiefrei.*

Innerhalb von ManufAg wird ein Schlüssel, **Hierarchie-Identifier (HI)** genannt, vorgeschlagen. Ein HI versieht einen Agenten mit einem systemweiten Namen, der unabhängig von der Laufzeitumgebung ist, auf der sich der jeweilige Agent befindet. Der HI eines hierarchischen Agenten dient der Speicherung von Informationen bezüglich der Eltern- sowie der Kindagenten dieses Agenten.

Durch den HI eines Agenten wird weiterhin verwaltet, welche Agenten die Dienste des Agenten nutzen können.

In bestimmten Situationen ist es sinnvoll, Agenten in **Clustern** zu organisieren. Diese Form der Organisation kann dazu verwendet werden, eine große Anzahl von Agenten auf verschiedene Unterorganisationen („Cluster“) aufzuteilen. Jedes Cluster erhält einen Agenten als Clusterrepräsentanten. Das Clusterkonzept ist ähnlich dem Konzept von Holonen (vergleiche [10, 24] bzw. die Ausführungen in Abschnitt 2.2.3). Eine Clusterorganisation kommt zum Einsatz, wenn ein Zusammenfassen von Agenten zu größeren Einheiten aus der Problemstellung heraus sinnvoll ist, gleichzeitig aber keine hierarchische Strukturierung möglich ist.

Neben den bisher betrachteten föderierten, hierarchischen sowie clusterartigen Organisationsformen existieren **hybride Ansätze**. Wir betrachten dazu das nachfolgende Beispiel.

Beispiel 5.4.13 (Hybride Organisationsformen) *In einem hierarchisch organisierten Multi-Agenten-System sind zwischen den Agenten auf einer Hierarchieebene Verhandlungen möglich, die nicht zentral gesteuert werden. Außerdem können Agenten auf einer Hierarchieebene zu Clustern zusammengefasst werden.*

Ontologien

Ontologien und Contentsprachen sind für die Kommunikation von Agenten wesentlich. Aufgrund der unterschiedlichen Produktionssteuerungsprobleme sind problemspezifische Ontologien erforderlich. Aus diesem Grund stellt ManufAg keine eigene Ontologie für die Produktionssteuerungsdomäne zur Verfügung. ManufAg bietet lediglich die Infrastruktur für die Integration von Ontologien an. Von der Ontologie wird nur gefordert, dass sie in .NET kompatible Klassenstrukturen transformiert werden kann, die von den Agenten interpretiert werden können. Jede Ontologie erhält einen eindeutigen Bezeichner für den Zugriff der Agenten.

Für die Behandlung von Contentsprachen wird ein ähnlicher Ansatz gewählt. ManufAg enthält ein Codec-Element, welches das Kodieren und Dekodieren der Nachrichten in der Contentsprache erlaubt. Das Codec-Element muss an die jeweilige Contentsprache angepasst werden. Jeder Agent kann gleichzeitig mehrere Ontologien und Contentsprachen durch einen geeigneten Content-Handler unterstützen. Bei den einzelnen Konversationen muss sowohl die Ontologie als auch die verwendete Contentsprache spezifiziert werden.

Wir diskutieren abschließend das Rahmenwerk unter Softwareengineering-Gesichtspunkten. ManufAg ist ein Grey-Box-Rahmenwerk. Wir verwenden Black-Box-Konstrukte zur Abbildung von konstanten Eigenschaften der Domäne. Wir betrachten dazu das folgende Beispiel.

Beispiel 5.4.14 (Konstante Eigenschaften der Domäne) *Jedes Produktionssteuerungssystem muss Lose und Ressourcen geeignet abbilden. Außerdem*

muss die Verwaltung von Ablaufplänen und Entscheidungseinheiten zur Berechnung von Ablaufplänen unterstützt werden. Die Beziehung zwischen diesen Elementen ist durch PROSA vorgegeben. Rollen und Interaktionsprotokolle können auf diese Weise als Blackbox-Konstrukte abgebildet werden.

White-Box-Techniken werden zur Abbildung von flexiblen Eigenschaften der Produktionssteuerungsdomäne herangezogen. Das nachfolgende Beispiel dient zur Illustrierung von flexiblen Eigenschaften einer Domäne.

Beispiel 5.4.15 (Flexible Eigenschaften der Domäne) *Die durch die Steuerung eines Produktionssystems zu erreichenden Ziele sind zu unterschiedlichen Zeitpunkten verschieden. So kann zwischen termin- und durchsatzorientierten Zielen unterschieden werden. Ziele werden durch Aktivitäten erreicht. Die zukünftigen Aktivitäten werden in der AIL vorgehalten. Die AIL wird als abstrakte Klasse vorgegeben. Durch Überschreibung dieser Klasse können bestimmte Ziele modelliert werden. Ein weiteres Beispiel für flexible Eigenschaften ist durch die Möglichkeiten zur Batchbearbeitung gegeben. Unter einem Batch verstehen wir dabei die gleichzeitige Bearbeitung mehrerer Lose auf einer Maschine. Die Batchbearbeitung wird durch die neue Rolle „Batch“ für Losagenten ermöglicht. Die Batchrolle wird durch Überschreibung der abstrakten Klasse „Rolle“ erreicht.*

Unter Verwendung von ManufAg wurde das Multi-Agenten-System FABMAS, das zur Steuerung von Halbleiterfertigungsprozessen verwendet wird, konstruiert. FABMAS wird in Abschnitt 7.1 dieses Kurses beschrieben.

Lösungen zu den Übungsaufgaben

Übungsaufgabe 5.1
Im Rahmen der Nummernsysteme erfolgt prinzipiell eine Unterscheidung zwischen manueller und automatischer Nummernvergabe. Bei der automatischen Nummernvergabe schlägt das System eine Nummer vor, während bei der manuellen Vergabe der Nutzer die neue Nummer vergibt.
Weiterhin werden Nummernkreise zur Einschränkung der möglichen Nummern verwendet. So ist es zum Beispiel möglich, jedes Jahr einen anderen Nummernkreis zu verwenden (z.B. 2008-XXXXXXX, wobei 2008 das Jahr repräsentiert). Dadurch lässt sich allein durch die Nummer des Geschäftsobjektes ermitteln, in welchem Jahr es angelegt wurde. Das ist besonders bei allen Arten von Aufträgen (Kundenaufträge, Fertigungsaufträge, Rechnungen usw.) sinnvoll.
Man kann jedoch auch andere semantische Informationen an die Nummer binden. So ist es zum Beispiel im Einzelhandel üblich, die Produktgruppe mit in die Nummer des Artikels einzubeziehen (z.B. Produktgruppennummer dreistellig - Produktnummer fünfstellig).

Übungsaufgabe 5.2
In der äußeren Schale des Kern-Schalen-Modells sind die unternehmensindividuellen Funktionen angesiedelt. Im Bereich der Halbleiterindustrie werden beispielsweise spezielle Verfahren zur Prognose, Terminierung und Ablaufplanung benötigt. Prognoseverfahren werden eingesetzt, da der Markt durch stark schwankende Nachfrage gekennzeichnet ist. Gleichzeitig sind die Kapitalbindungskosten sehr hoch. Verfahren zur Terminierung und Ablaufplanung werden benötigt, da es sich um ein komplexes Produktionssystem handelt und aufgrund des Nachfragermarktes die Termintreue einen hohen Stellenwert einnimmt.

Übungsaufgabe 5.3
Plattformunabhängigkeit wird durch Standards sichergestellt. Beispiele dafür sind der Posix-Standard für Betriebssysteme und die Java-Laufzeitumgebung. Auf den einzelnen Systemen sind diese Standards allerdings häufig unterschiedlich implementiert. So gibt es zwischen den unterschiedlichen Posix-Implementierungen minimale Abweichungen bezüglich Parametertypen und Funktionsrückgabewerten. Bei der Java-Laufzeitumgebung sind die GUI und das Threadmanagement bei den einzelnen Plattformen unterschiedlich umgesetzt. Dies macht sich im Aussehen der GUIs und im unterschiedlichen Locking-Verhalten der Threads bemerkbar.

Übungsaufgabe 5.4
Prinzipiell muss zwischen der Verfügbarkeit des Systems (tolerierbare Ausfallzeiten) und dem Umfang sowie den Kosten des Sicherungsaufwandes abgewogen werden. Mit zu berücksichtigen sind dabei:

- Zeiten für die Wiederherstellung,
- Testbarkeit des Sicherungskonzeptes,
- Umfang der Sicherung,
- rechtliche Anforderungen,
- Zeitpunkte der Sicherung.

Bei den Zeiten, die für die Wiederherstellung des Systems berücksichtigt werden müssen, ist beispielsweise die Dauer eines Desaster-Recovery zu betrachten, das mehrere Tage in Anspruch nehmen kann. Das Sicherungskonzept muss entsprechend getestet werden. Die erfolgreiche Sicherung ist zu verifizieren. Beim Umfang der Sicherung wird zwischen inkrementeller Sicherung und Vollsicherung unterschieden. Systeminstallationen müssen beispielsweise nicht jedes Mal mit gesichert werden. Die rechtlichen Anforderungen sind bei der Sicherung zu berücksichtigen, da beispielsweise geschäftsrelevante E-Mails aus steuerrechtlichen Gründen mehrere Jahre aufbewahrt werden müssen. Aus vertragsrechtlichen Gründen kann auch eine Chargenrückverfolgung für mehrere Jahrzehnte vorgeschrieben sein. Der Zeitpunkt der Sicherung ist so zu wählen, dass eine hohe Belastung des Produktivsystems vermieden wird. Dabei muss festgelegt werden, welcher Datenverlust tolerierbar ist.

Übungsaufgabe 5.5
Ausgehend von der Drei-Schichten-Architektur (vgl. Abschnitt 2.6.2) müssen folgende Punkte näher spezifiziert werden:

- Datenhaltungsschicht: Flexibilität des Datenmodells (Erweiterung einer betriebswirtschaftlichen Entität um zusätzliche Attribute); Möglichkeiten des direkten Zugriffs auf das Datenmodell (direkt auf die Datenbank, über Zugriffsmethoden),
- Anwendungsschicht: Schnittstellen zu Fremdsystemen; Plattformunabhängigkeit; Möglichkeit, eigene Logik zu implementieren bzw. anzupassen (welche Programmiersprache?),
- Präsentationsschicht: Anpassbarkeit der GUI (firmeninterne Dialoge und Logos); Integrationsmöglichkeit der GUI in andere Anwendung bzw. Einbindung anderer Anwendungen (Integration des Outlook-Adressbuchs in das Customer-Relationship-Management).

Übungsaufgabe 5.6
Die Fachabteilung kennt die Geschäftsprozesse des Unternehmens und somit die fachlichen Anforderungen an das betriebswirtschaftliche Standardsoftwaresystem. Den Aufwand der Anpassungen können die DV-Verantwortlichen jedoch

besser einschätzen. Außerdem sind sie für die Einführung und den Betrieb des Systems verantwortlich. Damit das einzuführende System auch von allen Abteilungen akzeptiert wird, müssen alle Beteiligten beim Auswahlprozess vertreten sein.

Übungsaufgabe 5.7
Im Falle des internen ASP übernimmt eine DV-Abteilung den Rechenzentrumsbetrieb und stellt den Fachabteilungen die entsprechenden Anwendungen zur Verfügung. Häufig erfolgt hier eine interne Abrechnung zwischen den Abteilungen auf Basis eines Profit-Center-Ansatzes. Beim externen ASP ist der Rechenzentrumsbetrieb an einen externen Anbieter ausgelagert.

Übungsaufgabe 5.8
Die Projektvorbereitung entspricht im Wesentlichen der ersten Phase des allgemeinen Modells. Der Business-Blueprint umfasst den ersten Teil der Feinspezifikation des allgemeinen Modells. Der zweite Teil der Feinspezifikation wird durch die Realisierung der Implementation-Roadmap abgedeckt. Die Realisierung umfasst weiterhin einen Teil der Prototypphase. Der andere Teil der Prototypphase des allgemeinen Modells wird durch die Produktionsvorbereitung abgedeckt. Die Produktionsvorbereitung beinhaltet weiterhin die Pilotphase. Go-Live-and-Support entspricht dem Produktivbetrieb des allgemeinen Modells. Die Zuordnung der Implementation-Roadmap zum allgemeinen Modell ist in Abbildung 5.18 graphisch dargestellt.

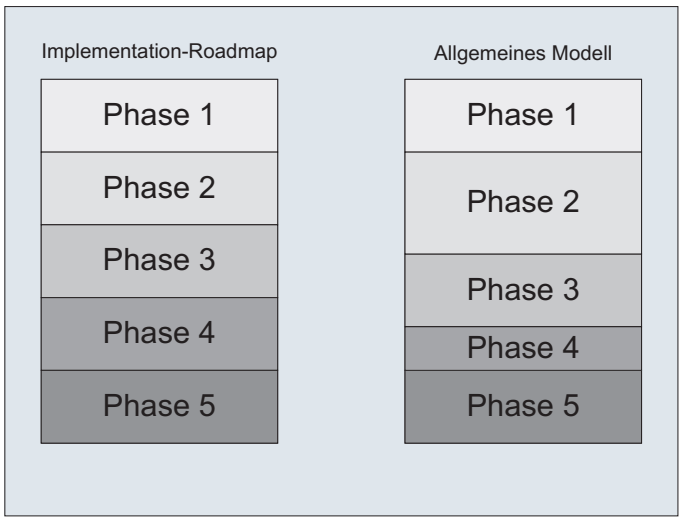

Abbildung 5.18: Implementation-Roadmap vs. allgemeines Phasen-Modell

Übungsaufgabe 5.9
Ein Buchungskreis innerhalb des SAP-Systems stellt ein rechtlich selbständiges

Unternehmen dar, das eigenständig steuerrechtlich bilanziert. Demgegenüber kann ein Mandant mehrere rechtlich selbständige Unternehmen umfassen, die auf einem gemeinsamen Datenbestand arbeiten. Typischerweise werden Holdings oder Konzerne als Mandanten und die dazugehörigen Unternehmen als Buchungskreise abgebildet. Der Geschäftsbereich dient zur externen segmentbezogenen Berichterstattung (beispielsweise für die Kapitalmärkte oder Investoren). Bei einem SAP-System müssen mindestens ein Mandant und ein Buchungskreis angelegt werden. In Abbildung 5.19 sind die Organigramme für die Kunststoff AG und das Transportunternehmen mit den SAP R/3-Organisationsstrukturen dargestellt.

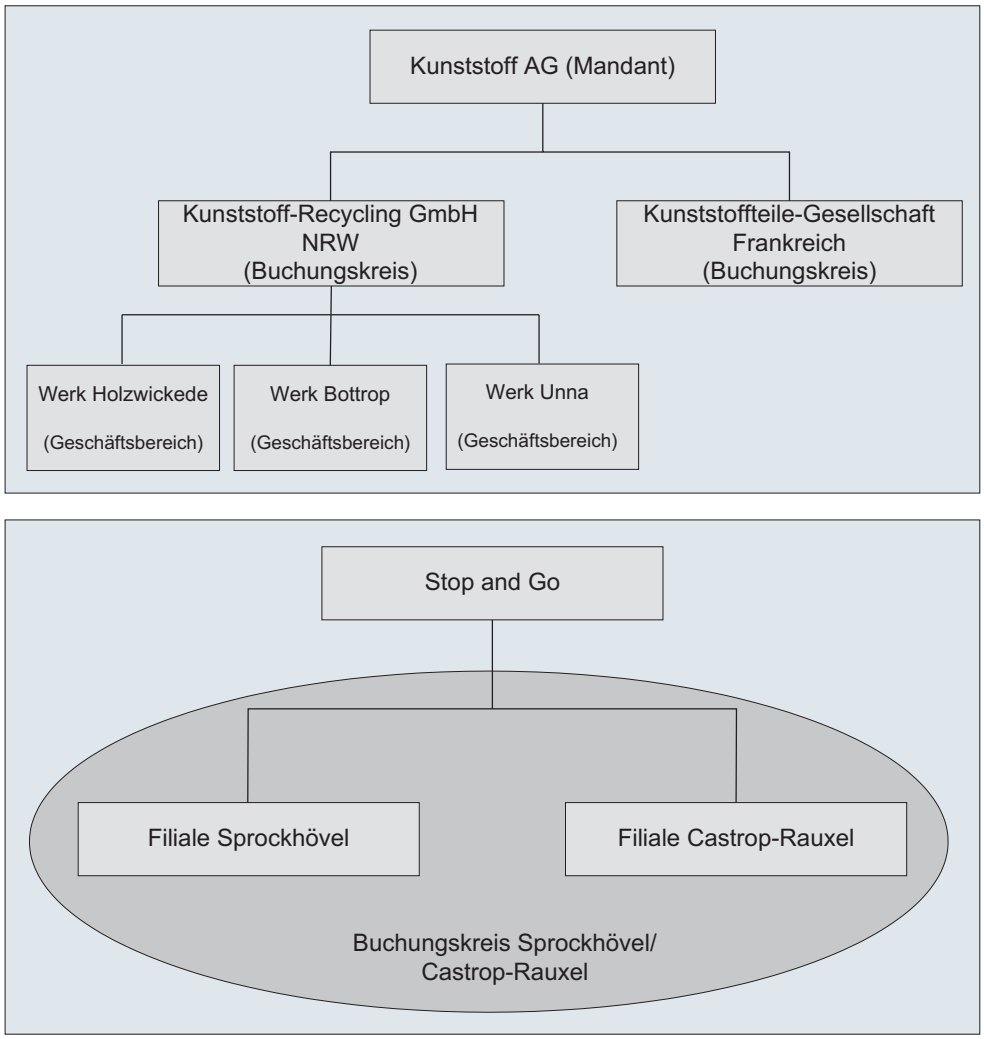

Abbildung 5.19: Organigramme für die Kunstoff AG und „Stop and Go“

Übungsaufgabe 5.10
Eine ereignisorientierte Kopplungsarchitektur lässt sich nur sinnvoll umsetzen, wenn das Zielanwendungssystem die BAPI-Schnittstelle bereitstellt. Diese Schnittstelle muss vom Quellanwendungssystem bei den jeweils relevanten Ereignissen angesprochen werden. Die Funktionen von Ereignis-, Empfänger- und Heterogenitätssubsystem sind geeignet zu ergänzen (vergleiche Abbildung 3.10).
Handelt es sich bei dem Quellanwendungssystem um ein SAP-System, wird das Ereignissubsystem mittels der ALE-Technologie realisiert. BAPIs lassen sich hier nicht verwenden, da sie als einfache Funktionsbausteine implementiert und somit passiv sind.

Übungsaufgabe 5.11
Im Wesentlichen kann den Argumenten aus Aufgabe 5.10 gefolgt werden. Weil BAPIs passiv sind, muss die Propagation von Datenänderungen nachimplementiert werden.

Übungsaufgabe 5.12
Ein zentrales Rechenzentrum, das weltweit angesprochen wird, birgt die folgenden Probleme.

- Wegen der unterschiedlichen Zeitzonen ist es schwierig, Wartungsintervalle (z.B. für die Datenbank) festzulegen.
- Die Latenzzeiten sind schlechter, wenn die Systeme in den einzelnen Ländern nicht vor Ort vorgehalten werden.
- Die Integration und Konsolidierung gewachsener Systeme in den Niederlassungen verursacht hohe Kosten.
- Häufig gibt es Probleme bzgl. unterschiedlicher Rechtssysteme (z.B. hat SWIFT wegen unterschiedlicher Datenschutzrechte die Verlagerung von Teilen des Rechenzentrums aus den USA nach Europa durchgeführt).
- Im Falle von Zerstörungen aufgrund höherer Gewalt (Krieg, Naturkatastrophen) birgt ein zentrales Rechenzentrum das Risiko des Totalausfalls des Systems.

Übungsaufgabe 5.13
Wenn in einem Agentensystem ein Agent A mit einem Agenten B direkt kommunizieren kann und Agent B mit einem Agenten C, dann kann nicht notwendigerweise Agent A direkt mit Agent C kommunizieren.
Das wird am Beispiel der hierarchischen Kommunikationsrechte deutlich. Ein Agent kann mit anderen Agenten auf der übergeordneten Ebene (Elternagenten) und auf der untergeordneten Ebene (Kindagenten) kommunizieren. Die Agenten A, B und C befinden sich jeweils auf einer Ebene einer dreistufigen Hierarchie.

Agent A ist der obersten und Agent C der untersten Ebene zugeordnet. Somit kann Agent A mit Agent B kommunizieren und Agent B mit Agent C. Agent A kann aber nicht mit Agent C kommunizieren.