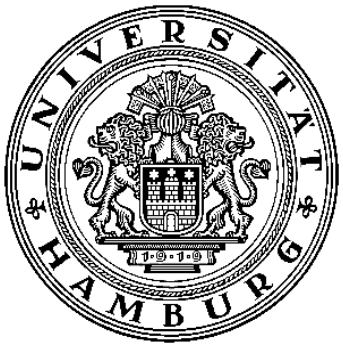

Universität Hamburg
FB Informatik

Konzepte und Techniken zur Adaption von betrieblicher Standardsoftware

Diplomarbeit

eingereicht bei

- Erstbetreuer -
Prof. Dr. J. W. Schmidt
Arbeitsbereich Softwaresysteme
Technische Universität Hamburg-Harburg

und

- Zweitbetreuer -
Prof. Dr. K. Brunnstein
Arbeitsbereich Angewandte Informatik
Universität Hamburg

von
André Wittenburg

9. März 2000

Zusammenfassung

Betriebliche Standardsoftwaresysteme liegen grundsätzlich in einem Initialzustand, d.h. als unternehmensneutrales System vor und werden durch *Adaption* in ein produktives System überführt. Für diese Transformation bieten die Hersteller von betrieblicher Standardsoftware verschiedene Techniken und Konzepte an, um die gestellten Anforderungen des Unternehmens in das Softwaresystem abzubilden. Die Auswahl der adäquaten Mittel zur Adaption der betrieblichen Standardsoftware aus den verschiedenen Möglichkeiten wird jedoch oftmals nur rudimentär unterstützt.

Die verschiedenen Konzepte und Techniken zur Adaption von betrieblicher Standardsoftware sind Gegenstand dieser Arbeit. Die einzelnen Techniken und Konzepte werden am Beispiel des R/3 Systems der SAP AG vorgestellt, eingeordnet und bewertet. Es wird eine Taxonomie entwickelt, in der die verschiedenen Konzepte und Techniken eingeordnet werden.

Danksagung

An dieser Stelle möchte ich mich bei allen Menschen bedanken, die mir diese Arbeit ermöglicht und mich während der Durchführung unterstützt haben.

Mein spezieller Dank gilt Herrn Prof. Dr. Joachim W. Schmidt für die Betreuung dieser Arbeit und die Bereitstellung der Mittel, um diese Arbeit durchzuführen. Die exzellenten Arbeitsbedingungen am Arbeitsbereich Softwaresysteme haben maßgeblich zum Gelingen dieser Arbeit beigetragen.

Herrn Prof. Dr. Klaus Brunnstein danke ich für die zusätzliche Betreuung und Unterstützung der Arbeit. Die Möglichkeit, im Oberseminar seines Arbeitsbereiches AGN einen Vortrag halten zu dürfen, hat mir einen neuen Sichtwinkel gegeben, der dieser Arbeit gut getan hat.

Herrn Dr. Gerald Schröder danke ich für die stetigen Ermutigungen, Anregungen und Diskussionen, die mich während meiner Arbeit stets vorangebracht haben. Ohne diese Zusammenarbeit mit Herrn Dr. Schröder, die sich inzwischen über mehrere Jahre erstreckt, wären mir viele Sichtweisen und Abstraktionen der Informatik verborgen geblieben.

Frau Andrea Willms danke ich für das Korrekturlesen dieser Arbeit.

Des weiteren danke ich allen Mitarbeitern des Arbeitsbereich Softwaresysteme der Technischen Universität Hamburg-Harburg, die mir während der Arbeit geholfen, durch Anregungen und Kritik neue Perspektiven eröffnet haben.

Mein besonderer Dank gilt meinen Eltern: Ohne ihr Zutun wäre dieses Studium nicht möglich gewesen.

Hamburg, den 9. März 2000 André Wittenburg

Inhaltsverzeichnis

1 Einleitung und Motivation **1**
1.1 Adaption von betrieblicher Standardsoftware 2
1.2 Gliederung der Arbeit 4

2 Adaption von Standardsoftware - Überblick **7**
2.1 Vom Business (Re-)Engineering zur Adaption von Standardsoftware 8
2.1.1 BPR - Business (Process) Reengineering 8
2.1.2 BE - (Continuous) Business Engineering 10
2.2 Adaption vs. Customizing 13
2.3 Vorgehen bei der Einführung von betrieblichen Standardsoftwaresystemen . . 15
2.3.1 Die Phasen der strukturierten Einführung mit ASAP 16
2.3.2 Werkzeuge zur Unterstützung des Einführungsprozesses 20

3 Architekturelle Klassifikation von Adaptionskonzepten **23**
3.1 Architektur betrieblicher Standardsoftwaresysteme 23
3.1.1 Persistente Datenhaltung - Datenbankserver 24
3.1.2 Applikationsebene - Applikationsserver 25
3.1.3 Präsentationsebene - Präsentationsserver 27
3.2 Ansätze zur Adaption in der Systemarchitektur 28
3.3 Charakterisierung und Kategorisierung der Adaptionsmechanismen 29

4 Konfiguration **33**
4.1 Anpassen an kundenspezifische Anforderungen 33
4.2 Formularwesen - Generierung und Modifikation 36
4.3 Berichtswesen - Reports 39
4.4 Benutzerrechte und Benutzerverwaltung 43
4.4.1 Standardbenutzer 44
4.4.2 Berechtigungsobjekte und Berechtigungen 44
4.4.3 Profile und Sammelprofile 45

5 Erweiterung **47**
5.1 Workflow-Systeme in betrieblicher Standardsoftware 48
5.1.1 Die Architektur der Workflow-Komponente 48
5.1.2 Werkzeuge zur Erstellung und Modifikation von Workflows 51
5.1.3 Aufbau und Semantik von Business-Objekten 52
5.2 Erweiterung der Standardfunktionalitäten 54
5.3 Regeln zur Validierung und Substitution 58

5.3.1 Überprüfung von Eingaben durch Validierung . . . 58
5.3.2 Ersetzen von Eingaben durch Substitution . . . 61
5.4 Personalisierung der Standardsoftware . . . 63
5.5 Anbindung von Standardsoftwaresystemen an das Internet . . . 64
5.5.1 Architektur eines Servers zur Anbindung an das Internet . . . 64
5.5.2 Eine Erweiterung der HyperText Markup Language (HTML) . . . 69
5.5.3 Internet Anwendungen und Komponenten . . . 70

6 Kopplung **75**
6.1 Massendatenverarbeitung . . . 76
6.2 Programmierschnittstellen einer Standardsoftware . . . 78
6.2.1 CPI-C - Common Programming Interface for Communication . . . 78
6.2.2 RFC - Remote Function Call . . . 79
6.3 Applikations-Kopplung . . . 84
6.3.1 Aufbau von IDocs . . . 86
6.3.2 ALE Ausgangsverarbeitung . . . 87
6.3.3 ALE Eingangsverarbeitung . . . 88
6.3.4 Erzeugen eines IDoc von DEBMAS03 . . . 89
6.4 Interprogramm Kommunikation über Objektmodelle . . . 90
6.4.1 BAPI - Business Application Programming Interface . . . 91
6.4.2 Nutzung von OLE, ActiveX, COM und DCOM mit Standardsoftwaresystemen . . . 93

7 Zusammenfassung und Ausblick **97**
7.1 Zusammenfassung . . . 97
7.2 Taxonomie der Techniken und Konzepte . . . 98
7.3 Bewertung und Ausblick . . . 98

A Quellcode eines Java RFC **101**

B Quellcode eines BAPI Aufrufes unter Java **107**

C Quellcode eines RFC unter VisualBasic als Excel-Makro **111**

Glossar **113**

Literaturverzeichnis **119**

Abbildungsverzeichnis

2.1 Kommunikation zwischen Geschäftswelt und IT-Welt 10
2.2 Wertschöpfungskette für einen Neuwagenverkauf [CK99] 11
2.3 Zusammenhang von Wertschöpfungsketten und EPKs 12
2.4 Ereignisgesteuerte Prozeßketten und ihre vier Sichten 13
2.5 Elemente der ereignisgesteuerten Prozeßketten 14
2.6 Die ASAP Roadmap [SAP99b] 17
2.7 Der ASAP „Implementation Assistant“ [SAP99b] 20
2.8 Die ASAP „Question and Answer database“ [SAP99c] 21
2.9 Das ASAP „Concept Check Tool“ [SAP99a] 22

3.1 Client/Server-Architektur des R/3 Systems 24
3.2 Open vs. Native SQL 25
3.3 Zusammenhang der Kategorien zur Adaption von Standardsoftware 30

4.1 Der Einführungsleitfaden des R/3 Systems 35
4.2 Seitenfenster eines SAPscript Formulars 38
4.3 Quellcode des Reports - Druckprogramm zu SAPscript 40
4.4 Ausgabe von SAPscript 41

5.1 Architektur des SAP Business Workflows [BMS99] 48
5.2 Grafische Visualisierung einer Workflow-Definition 52
5.3 Aufbau eines Business-Objektes [SAP98d] 53
5.4 ABAP-Quellcode eines Funktionsexits 58
5.5 Ablaufdiagramm einer Validierung 59
5.6 Ablaufdiagramm einer Substitution 62
5.7 Der SAP Session Manager 63
5.8 Architektur des Internet Transaction Servers 65
5.9 Globale Service-Beschreibung eines Internet Transaction Servers 67
5.10 Ablaufdiagramm einer ITS-Sitzung 68
5.11 ABAP-Quellcode eines WebRFCs 72
5.12 ABAP-Quellcode eines Web-Reports 73

6.1 Programmierschnittstellen des R/3 Systems 76
6.2 Batch-Input Verfahren 77
6.3 Quellcode eines ABAP Reports - RFC zwischen zwei R/3 Systemen 82
6.4 RFC über Java 83
6.5 Kommunikation von zwei R/3 Systemen über ALE 85
6.6 Aufbau eines IDoc-Typ (DEBMAS03) 87

6.7 Nachrichtenaustausch über ALE 89
6.8 IDoc vom IDoc-Typ DEBMAS03 90

Kapitel 1

Einleitung und Motivation

Softwaresysteme werden in Unternehmen zur Unterstützung der verschiedensten Aufgaben und Tätigkeiten eingesetzt. Beispiele sind Softwareprogramme, die Roboter bei einem Automobilhersteller steuern, oder Betriebssysteme, welche die Grundlage für die verschiedensten Anwendungen wie CAD-Werkzeuge, Textverarbeitungen oder Datenbanken bilden. Eine Klassifizierung dieser Systeme kann entweder anhand ihrer Funktionen bzw. des Einsatzbereiches oder anhand der Entwicklung geschehen. Wird anhand der Entwicklung klassifiziert, so können folgende Klassen differenziert werden:

- Individualsoftware, die von dem Unternehmen oder für das Unternehmen speziell entwickelt wird.
- Standardsoftware, die von einem Softwarehaus vertrieben wird und in mehreren Unternehmen einsetzbar ist.

Scheer definiert Standardanwendungssoftware wie folgt:

> Standardanwendungssoftware sind von Softwarehäusern und Hardwareherstellern für einen anonymen Markt entwickelte Programmsysteme zur Lösung von Anwendungsproblemen [Sch90].

Standardanwendungssoftware wird demnach für einen anonymen Markt entwickelt. Dies bedeutet zum einen, daß ein Auftraggeber zur Erstellung dieser Software nicht unbedingt notwendig ist, und zum anderen, daß die Software nicht exakt auf ein Unternehmen „zugeschnitten“, sondern prinzipiell für beliebige Unternehmen verwandter Art entwickelt ist. Bei Balzert findet sich eine ähnliche Definition für Standardsoftware und zusätzlich eine Definition für Individualsoftware:

> Standardsoftware ist ein Softwareprodukt, das für einen Markt entwickelt wird und das industriell vertrieben wird.
> ...
> Individualsoftware ist das Ergebnis eines Projektes für einen bekannten Auftraggeber [Bal97].

Beide Definitionen beschäftigen sich im weitesten Sinne mit Standardsoftware. Für *betriebswirtschaftliche*[1] Standardsoftware werden die obigen Definitionen wie folgt eingeschränkt:

Betriebswirtschaftliche Individualsoftware wird für ein Unternehmen erstellt und nur bei diesem Unternehmen eingesetzt. Sie ist auf dieses Unternehmen und seine Geschäftsprozesse ausgerichtet, so daß die spezifischen Geschäftsprozesse dieses Unternehmens optimal unterstützt werden. Anpassungen an veränderte Geschäftsprozesse und an neue Umfeldanforderungen (zum Beispiel Euro, Mehrwertsteueränderungen, etc.) lassen sich nur durch Änderung der internen Programmabläufe realisieren. Das bedeutet einen beständigen Entwicklungs- und Pflegeaufwand, der zu hohen Kosten führt.
Vorteil: optimale Unterstützung der Unternehmensprozesse,
Nachteil: hohe Entwicklungs- und Pflegekosten.

Betriebswirtschaftliche Standardsoftware, wie SAP R/3, Oracle Financials oder BaanERP, bildet im Kern Standardgeschäftsprozesse und -daten vieler Unternehmen ab und ist für einen Markt entwickelt worden. Viele Unternehmen setzen die Standardsoftware ein, so daß sich die Entwicklungskosten aufteilen und damit reduzieren. Die Pflege der Standardsoftware wird zentral für alle Unternehmen vorgenommen und durch *Release-Wechsel* oder *Updates* an alle Unternehmen weitergegeben; dieser Prozeß ist somit ebenfalls relativ kostengünstig. Standardsoftware dient auch dem *Know-how-Einkauf*, da Unternehmen die Prozesse der Standardsoftware nutzen, um ihre eigenen Abläufe zu optimieren. Die Unternehmen finden allerdings keine Unterstützung von der Standardsoftware für spezifische Geschäftsprozesse, die sie gegenüber Mitbewerbern abheben.
Vorteil: niedrige Entwicklungs- und Pflegekosten,
Nachteil: keine Unterstützung spezifischer Geschäftsprozesse.

Da die Entwicklung und Pflege individueller Software für alle Geschäftsprozesse eines Unternehmens zu hohe Kosten verursachen, wird zunehmend betriebliche Standardsoftware eingesetzt. In einem Unternehmen, in dem diese Standardsoftware eingesetzt werden soll, sind normalerweise nicht exakt diese Standardgeschäftsprozesse und -daten vorzufinden, sondern Variationen davon.

Bei der Einführung der betriebswirtschaftlichen Standardsoftware im Unternehmen sind daher entweder die Unternehmensprozesse an die Software oder die betriebswirtschaftliche Standardsoftware an das Unternehmensumfeld anzupassen. Wenn spezifische Geschäftsprozesse einen Wettbewerbsvorteil versprechen oder für die jeweilige Unternehmung unerläßlich sind, ist die Anpassung oder sogar individuelle Erweiterung der Standardsoftware zur Sicherung des Unternehmenserfolgs unumgänglich.

1.1 Adaption von betrieblicher Standardsoftware

Die Anbieter von betriebswirtschaftlicher Standardsoftware stellen zahlreiche Anpassungs- und Erweiterungsschnittstellen zur Verfügung. Die Funktionalitäten dieser Schnittstellen werden den unterschiedlichen Änderungsanforderungen der Unternehmen gerecht. Dabei besteht

[1] Wird im Folgenden von *betrieblicher* Standardsoftware gesprochen, so ist dies in dieser Arbeit mit *betriebswirtschaftlicher* Standardsoftware gleichzusetzen.

allerdings eine Gefahr: Je größer die Änderungen ausfallen, desto mehr steigt der Aufwand auf Seiten des Unternehmens, insbesondere der Pflegeaufwand bei *Release-Wechseln* der Standardsoftware, welche die Änderungen invalidieren können. Daher ist die Anpassung und Erweiterung ebenso sorgfältig zu planen und durchzuführen wie eine individuelle Eigenentwicklung von Software. Insbesondere ist auf Seiten des Unternehmens abzuschätzen, welche Adaptionsmechanismen für die spezifischen Änderungsanforderungen adäquat sind, also eine Realisierung der Anforderungen erlauben und dabei trotzdem den Entwicklungs- und Pflegeaufwand minimieren.

Zu beachten ist auch die Größe und Komplexität der Standardsoftwaresysteme. Oftmals wird viel Funktionalität angeboten, die das Unternehmen zwar miterwerben muß, die aber nicht benötigt bzw. gewünscht wird. Dieser Faktor kann eine Standardsoftware teurer als eine Individualsoftware werden lassen, da ungenutzte Funktionalität gekauft werden muß und die Adaption der Software wesentlicher komplexer wird. Oftmals muß auch die nicht genutzte Funktionalität bei der Adaption beachtet werden, um Fehlverhalten zu vermeiden.

In dieser Arbeit sollen daher die verschiedenen Änderungsanforderungen klassifiziert und die Möglichkeiten zur Adaption und Erweiterung von betriebswirtschaftlicher Standardsoftware konzeptuell eingeordnet werden. Die Adaptionstechniken werden am Beispiel des R/3 Systems der SAP AG erläutert.

Änderungsanforderungen werden beispielsweise klassifiziert nach:

- Art der Änderung (wegstreichen, hinzufügen, verändern)
- Einordnung der Änderung (Daten, Funktionalität, Ablauf)
- Umfang der Änderung (zu berücksichtigende Abhängigkeiten, Anzahl und Umfang der Eingriffe)
- Persistenz der Änderung (Releaseabhängigkeit, Verfügbarkeit eingesetzter Mechanismen und Werkzeuge)

Konzeptuelle Eckpunkte bei der Adaption sind:

- Ansatzpunkte in der Systemarchitektur (wo wird geändert?)
- Sprachmittel und Werkzeuge (womit wird geändert?)
- Flexibilität (wieviel kann geändert werden?)
- Stabilität (welche Fehlerquellen sind möglich?)
- Langfristigkeit (welche Auswirkungen haben die Änderungen, insbesondere auf neue Releases?)

Zu untersuchen, einzuordnen und zu bewerten sind im Einzelnen die angebotenen Mechanismen und Werkzeuge zur Adaption am Beispiel des R/3 Systems.

1.2 Gliederung der Arbeit

Im Anschluß an diese Einleitung wird in Kapitel 2 ein Überblick über die Adaption von Standardsoftware gegeben. Ausgangspunkt für die Adaption von Standardsoftware sind Modelle für Geschäftsprozesse und die mit den Modellen verbundenen Methoden zur Modifikation, Beschreibung und zum Verstehen von Geschäftsprozessen. Die Methoden des *Business Process Reengineering* und des *Continuous Business Engineering* werden auf ihre Verbindung zur Adaption hin untersucht, und die Modelle der SAP AG zur Visualisierung von Geschäftsprozessen aufgezeigt (siehe Kapitel 2.1).

Die beiden Begriffe *Adaption* und *Customizing* werden im Zusammenhang mit Standardsoftware oftmals synonym verwendet, beide beschreiben den Prozeß des Einstellens und Änderns der Standardsoftware. Da die SAP AG eine weitestgehend eigene Begriffswelt hat, werden in Kapitel 2.2 die beiden Begriffe *Adaption* und *Customizing* gegeneinander abgegrenzt und ihre weitere Bedeutung innerhalb dieser Arbeit definiert.

Bei der Einführung von Standardsoftware werden von den Herstellern verschiedene Werkzeuge und „Projekthilfen“ zur Verfügung gestellt. Die SAP AG bietet seit 1998 hierzu die Methode *AcceleratedSAP* (ASAP) an, die Gegenstand des Kapitels 2.3 ist. *AcceleratedSAP* soll die Einführung beschleunigen und fast alle hierzu notwendigen Schritte der Einführung in einem Paket zusammen.

In Kapitel 3 werden die verschiedenen Adaptionstechniken klassifiziert. Hierzu wird zunächst eine Einführung in die Systemarchitektur von Standardsoftware (am Beispiel des R/3 Systems) in Kapitel 3.1 gegeben, um anschließend in Kapitel 3.2 verschiedene Adaptionstechniken entlang der Systemarchitektur einzuordnen. Daran anschließend folgt in Kapitel 3.3 eine Charakterisierung der Techniken, die zur weiteren Gliederung dieser Arbeit führt.

Kapitel 4 mit dem Titel *Konfiguration* enthält die Adaptionsmechanismen und -techniken, die bei der Einführung einer Standardsoftware unerläßlich sind. Dahinter steht die Frage: „Welche Einstellungen müssen gemacht werden?“ Zu den Antworten zählen:

- *Customizing*[2] zur Einstellung der unternehmensneutral ausgelieferten Funktionalität an die spezifischen betriebswirtschaftlichen Anforderungen des Unternehmens;
- Text-Managementsystem zur Erstellung von Rechnungen, EDI-Formularen, etc. (R/3: SAPscript);
- Berichtswesen zur Erstellung von Berichten und Auswertungen (R/3: Reports);
- Benutzerrechte und Benutzerverwaltung, um Benutzer auf Betriebssystem- und Datenbankebene sowie im Standardsoftwaresystem einzurichten und zu verwalten.

Standardsoftwarepakete bieten außerhalb der Standardfunktionalität weitere optionale Module und Werkzeuge. Diese Module und Werkzeuge sind Gegenstand des Kapitels 5 (Erweiterung):

- Workflow-Systeme, zur Automatisierung von Abläufen;

[2] Customizing im Sinn der SAP (vgl. Kapitel 2.2)

- Erweiterungen beziehungsweise Änderungen der Standardfunktionalität (R/3: *User-* und *Customer-Exits*);
- Validierungen und Substitutionen, um Eingaben (hierbei kann es sich um Benutzereingaben oder Eingabe durch ein System handeln) mit Regeln zu hinterlegen (R/3: Validierung und Substitution);
- Systeme zur Anbindung des Standardsoftwaresystems an das Internet (R/3: *Internet Transaction Server*).

Die zu beantwortende Frage dieses Kapitels 5 ist:
„Welche Einstellungen können zusätzlich vorgenommen werden?“

In Kapitel 6 werden die Möglichkeiten der Kopplung von Standardsoftwaresystemen mit externen Softwaresystemen analysiert. Die Frage „Wie können externe Systeme angebunden werden?“ soll beantwortet werden. Die zu analysierenden Techniken sind:

- Massendaten-Import/-Export und Hintergrundverarbeitung (R/3: *Batch-Input*);
- Kommunikationsprotokolle der Standardsoftware in Netzwerken (R/3: z.B. *Common Programming Interface for Communication (CPI-C)*);
- Funktionsaufrufe auf entfernten Rechnern bzw. *Remote Procedure Calls (RPC)* (R/3: *Remote Function Call (RFC)*);
- Möglichkeiten für den Nachrichtenaustausch mit anderen Systemen (R/3: *Application Link Enabling (ALE) mit IDocs*);
- Interprogramm-Kommunikation über Objektmodelle (R/3: *Business Application Programming Interface (BAPI)* und Microsoft: *(Distributed) Component Object Model ((D)COM)*).

In der Zusammenfassung (Kapitel 7) wird „ein Schritt zurück gemacht“, um die verschiedenen Adaptionstechniken global zu betrachten und zu bewerten. Eine Taxonomie klassifiziert und verallgemeinert die Konzepte und Techniken. Hierbei wird von einem speziellen Standardsoftwaresystem abstrahiert und eine allgemeingültige Einordnung der Adaptionstechniken für Standardsoftware getroffen.

Kapitel 2

Adaption von Standardsoftware - Überblick

Eine Standardsoftware bildet im Kern Standardgeschäftsprozesse ab und ist nicht auf ein Unternehmen zugeschnitten, sondern wird für einen Markt mit einer Vielzahl von Unternehmen entwickelt. Bei der Einführung einer Standardsoftware müssen Standardgeschäftsprozesse entweder an die im Unternehmen vorliegenden Geschäftsprozesse angepaßt werden oder die im Unternehmen existenten Geschäftsprozesse werden an die Standardgeschäftsprozesse angepaßt.

Wenn das Anpassen von Geschäftsprozessen von Seiten des Unternehmens als vorteilhaft erkannt wird oder ein Überdenken der Geschäftsprozesse in Verbindung mit dem *Know-how* der Standardsoftware geplant ist, so bieten sich verschiedene Modelle und Methoden an, die ihren Ursprung sowohl in der *Betriebswirtschaftslehre* als auch in der *Informatik* haben (siehe Kapitel 2.1).

Entscheidet sich ein Unternehmen für die Anpassung der Standardgeschäftsprozesse, so sprechen die Hersteller von Standardsoftwaresystemen oftmals (z.B. SAP) von „*Customizing*“. Eine Abgrenzung vom Begriff *Adaption*, der in dieser Arbeit als Oberbegriff für Änderungen, Einstellungen, etc. genutzt wird, zu *Customizing* folgt in Kapitel 2.2.

Die Einführungsprojekte von Standardsoftwaresystemen unterscheiden sich im Projektaufbau nach einzuführenden Softwarekomponenten und den unternehmensspezifischen Gegebenheiten. Beratende Unternehmen setzen des weiteren oftmals eigene Methoden ein, um die Einführung zu planen und durchzuführen. Dies erschwert das nachträgliche „Einsteigen“ in das Einführungsprojekt einer Standardsoftware für Berater, die keine Kenntnis über die verwendeten Methoden und Modelle haben. Zur Erleichterung dieses „Einstieges“ und zur einheitlichen Strukturierung von Einführungsprojekten wurden, um dieses Vorgehen zu standardisieren, von den Herstellern von Standardsoftware verschiedene Modelle, Methoden und Hilfsmittel entwickelt. Ein Beispiel für solch ein Modell findet sich in Kapitel 2.3.

2.1 Vom Business (Re-)Engineering zur Adaption von Standardsoftware

Entscheidet sich ein Unternehmen für die Einführung einer Standardsoftware, so kann dies mehrere Gründe haben:

- Die bisher eingesetzte Software ist zu heterogen und eine Kopplung der Systeme ist nicht möglich.
- Das Unternehmen befindet sich in einem Stadium des Überdenkens, in dem es versucht, bisherige Strukturen neu zu definieren und Prozesse zu ändern [CK99].
- Die eingesetzte Software deckt nur teilweise die Bedürfnisse des Unternehmens, und es müßte eine neue Software eingeführt werden, um fehlende Bereiche abzudecken. Die Möglichkeit eine Standardsoftware einzuführen, die gleichzeitig das Altsystem ersetzt, bietet sich an [CK99].
- Die Fusion mit einem anderen Unternehmen erzwingt die Entscheidung für ein gemeinsames Softwaresystem, wenn keines der alten Systeme die Bedürfnisse abdeckt und insbesondere die erhöhte Benutzerzahl bewältigen kann.
- Eine totale Umstrukturierung des Unternehmens ist geplant. Als Unterstützung soll eine Standardsoftware eingeführt werden, wobei die Prozesse der Standardsoftware als Vorlage dienen sollen [CK99].
- Das bisherige Softwaresystem ist nur unzureichend dokumentiert und eine Erweiterung des Systems ist nicht möglich. Durch Einsatz eines Standardsoftwaresystems wird eine ausführliche Dokumentation miterworben und die Skalierbarkeit des Systems wird durch den Hersteller der Standardsoftware gewährleistet.

Diese Aufzählung kann durch viele andere Szenarien erweitert werden, in denen sich die Einführung einer Standardsoftware als Möglichkeit bietet. Entscheidet sich ein Unternehmen für die Einführung des R/3 Systems, so geht dies zumeist mit einer Reorganisation im Unternehmen einher. Hierbei sollen auch die Geschäftsprozesse des Unternehmens kritisch hinterfragt werden.

Im Folgenden sollen zwei Methoden untersucht werden, die sich für das Untersuchen, Überdenken bzw. Ändern von Geschäftsprozessen eignen. Das in der Mitte der 80er Jahre entwickelte *Business Process Reengineering* wird in Kapitel 2.1.2 beschrieben. Anschließend wird das *(Continuous) Business Engineering* behandelt, welches in den 90er Jahren mehr und mehr an Popularität gewonnen hat [CK99]. In Kapitel 2.1.2 werden inbesondere die Modelle des R/3 Systems für *Business Engineering* vorgestellt.

2.1.1 BPR - Business (Process) Reengineering

Wird vom Business Reengineering gesprochen, so wird dies häufig mit dem Buch von Michael Hammer und James Champy [HC95] in Verbindung gebracht.

Die Definition von Hammer und Champy für Business Reengineering lautet wie folgt:

> "Reengineering,"probably, is "the fundamental rethinking and radical redesign of business processes to achieve dramatic improvements in critical contemporary measures of performance, such as cost, quality, service, and speed."[1]

Aus dieser Definition extrahieren Hammer und Champy vier Schlüsselwörter, die sie wie folgt beschreiben:

FUNDAMENTAL: Die entscheidende Frage, die beim Reengineering gestellt werden muß, lautet: *"Why do we do what we do? And why do we do it the way we do?"*
Diese fundamentale Frage soll ein Unternehmen zum Überdenken des eigenen Geschäftsbereiches anregen und eine Untersuchung erzwingen, warum bestimmte Wege eingeschlagen werden. Des weiteren beginnt Reengineering beim *Nullpunkt*. Reengineering nimmt nichts als vorhanden an, sondern will einen Neubeginn von Grund auf.

RADICAL: An dieser Stelle ist nicht das umgangssprachliche Wort *radical (engl.)* bzw. *radikal (deutsch)* gemeint, sondern das vom Lateinischen abgeleitete Wort *radix*, welches ins Deutsche übersetzt „Wurzel, Ursprung und Quelle" meint. Business Reengineering beschäftigt sich nicht mit Vorhandenem, sondern soll an der „Wurzel" beginnen.

DRAMATIC: Es ist nicht das Ziel von Business Reengineering nur marginale Veränderungen zu machen. Das Ziel ist eine „dramatische" Veränderung, die Altes verwirft und Neues aufbaut.

PROCESSES: Business Reengineering arbeitet nicht auf der Ebene der (Einzelschritt-) Aufgaben, sondern auf der Prozeßebene.

Business Reengineering bezieht sich nicht auf die Dekomposition der Arbeitsabläufe in elementare Arbeitsschritte, sondern fokussiert den Gesamtablauf, also den Prozeß.

Die Unternehmen, die sich dem Business Reengineering unterzogen haben, waren zumeist sehr erfolgreich, jedoch kann Business Reengineering immense Kosten verursachen, da eine totale Umstrukturierung immer mit hohem Aufwand verbunden ist [CK99]. Des weiteren muß eine große Risikobereitschaft vorhanden sein, um einen „Neuanfang" zu wagen, der sich aus dem Reengineering ergibt. Die Unternehmen, die sich diesen Risiken stellten, entschlossen sich zumeist direkt aus der Führungsebene zu diesem Schritt, da nur die Geschäftsleitung die nötigen Mittel hat, um fundamentale Änderungen durchzuführen.

Ein erfolgreiches Beispiel für das *Reengineering* stellt Michael Hammer in [Ham90] vor. Der Automobilhersteller *Ford Motor Company* hat sich Anfang der 80er Jahre dem *Reengineering* unterzogen und seinen Kreditorenbereich neu strukturiert:

> Ford didn't settle for the modest increase it first envisioned. It opted for radical change — and achieved dramatic improvement. Where it has instituted this new process, Ford has achieved a 75% reduction in head count, not the 20% it would have gotten with a conventional program. And since there are no discrepancies

[1] Hervorhebung wie in Hammer und Champy [HC95]

> between the financial record and the physical record, material control is simpler and financial information is more accurate.

Insbesondere der zu hohe Kapitalaufwand und das hohe Risiko des *BPR* haben zu anderen Ansätzen geführt, die versuchen, das Risiko und das eingesetzte Kapital zu mindern. Dies führt unter anderem zum *(Continuous) Business Engineering*, welches die Möglichkeiten der Informationstechnologie mit den Methoden des *Business (Process) Reengineering* verbindet.

2.1.2 BE - (Continuous) Business Engineering

Die Verschmelzung von *Business Process Reengineering* und den Techniken der Informationstechnologie hat das *Business Engineering* hervorgebracht [CK99]. Die Zielsetzung des *Business Engineering* ist die Optimierung von Geschäftsprozessen. Um dieses Ziel zu erreichen, werden dem *Business Engineering* Modelle und Hilfsmittel zur Seite gestellt.

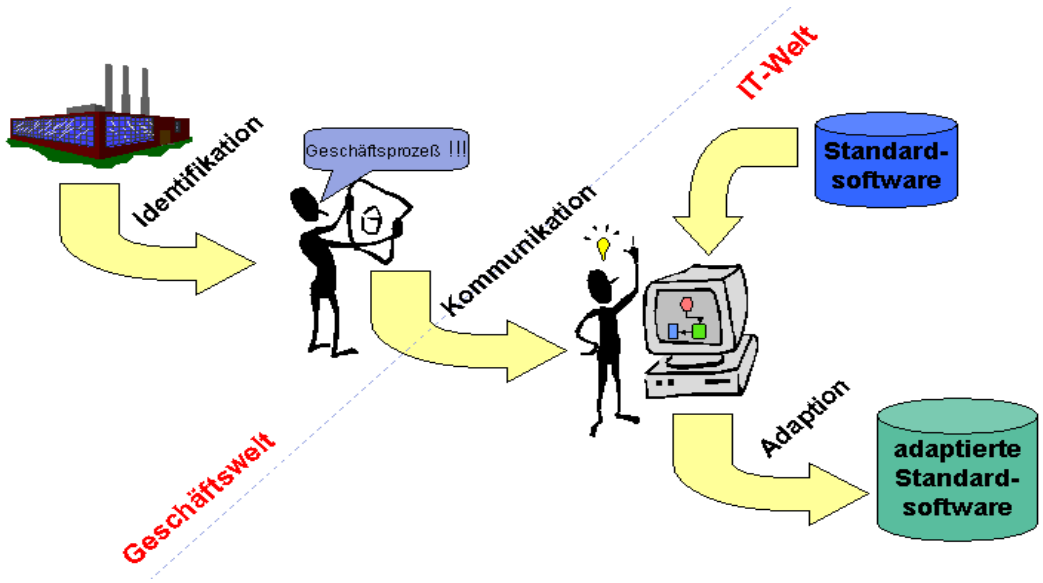

Abbildung 2.1: Kommunikation zwischen Geschäftswelt und IT-Welt

Bei der SAP AG ist dies der *Business Engineer* mit den Modellen der *Wertschöpfungsketten* und den *ereignisgesteuerten Prozeßketten (EPKs)*. Zusätzlich dient der *Business Blueprint*, der im SAP Sprachgebrauch auch als R/3-Referenzmodell bezeichnet wird, als Sammlung aller Geschäftsprozesse des Standardsoftwaresystems als Ausgangspunkt für das *Business Engineering* gebildet. Gerade dieser *Business Blueprint* mit seinen Geschäftsprozessen, in denen die Erfahrungen der Standardsoftwarehersteller gebündelt sind, stellt eine solide Plattform für ein erfolgreiches *Business Engineering* dar.

Business Engineering ermöglicht die Kommunikation der *Geschäftswelt* und der *IT-Welt* unter Zuhilfenahme geeigneter Methoden und Modelle. Die Visualisierung dieses Prozesses ist in Abbildung 2.1 skizziert. Hierbei werden zunächst die Geschäftsprozesse eines Unternehmen identifiziert und durch Modelle repräsentiert. Durch Kommunikation der beiden Welten

erfolgt eine Abbildung der Geschäftsprozesse aus der *Geschäftswelt* in die *IT-Welt* hinein. Innerhalb der *IT-Welt* werden diese Geschäftsprozesse abschließend in ein Standardsoftwaresystem integriert.

Wertschöpfungsketten als Modell des Business Engineerings

Wertschöpfungsketten repräsentieren die höchste Abstraktionsschicht der Modelle des *Business Engineering*. Das Konzept der Wertschöpfungskette beschreibt nach Porter [Por98] den Wettbewerbsvorteil eines Unternehmens und gliedert diesen in strategisch relevante Kategorien.

> "The value chain disaggregates a firm into its strategically relevant activities in order to understand the behavior of costs and the existing and potential sources of differentiation." [Por98]

Abbildung 2.2: Wertschöpfungskette für einen Neuwagenverkauf [CK99]

Keller und Curran [CK99] kategorisieren die Wertschöpfungsaktivitäten zunächst in die allgemeinen Kategorien primäre Aktivitäten und unterstützende Aktivitäten. Zu den primären Aktivitäten gehören Eingangslogistik, Ausgangslogistik, Operationen, Marketing&Vertrieb und Kundendienst. Die unterstützenden Aktivitäten sind die Unternehmensinfrastruktur, die Personalwirtschaft, die Technologieentwicklung und die Beschaffung.

Ein Beispiel für eine Wertschöpfungskette mit verschiedenen Aktivitäten ist in Abbildung 2.2 gegeben. Die einzelnen Wertschöpfungsaktivitäten werde durch ein *Chevron* beziehungsweise einen *Fisch* symbolisiert. Die einzelnen Wertschöpfungsaktivitäten, die im R/3 Umfeld auch als *Szenarien* bezeichnet werden, sind miteinander verkettet. Im Beispiel stellen sie den Verkauf eines Neuwagens mit seinen einzelnen Aktivitäten dar.

Hinter den einzelnen Aktivitäten bzw. Szenarien verbergen sich detaillierte Prozesse, die wiederum durch Prozeßmodelle visualisiert werden. Greift man das Szenario *Rechnung* aus der Wertschöpfungskette heraus, so kann der dahinterliegende Szenarioprozeß wie in Abbildung 2.3 visualisiert werden. Dieses Modell führt zu den Geschäftsprozeßmodellen. Je nach Standardsoftwaresystem werden unterschiedliche Geschäftsprozeßmodelle verwendet (z.B.: *ereignisgesteuerten Prozeßketten (EPK)* [CK99], *Aktivitätsdiagramme* aus UML [BRJ99] oder *Petri-Netze* [JV87]). Das R/3 System der SAP AG verknüpft diese beiden Modelle im *Business Engineer* und erlaubt die Navigation zwischen den einzelnen Modellen.

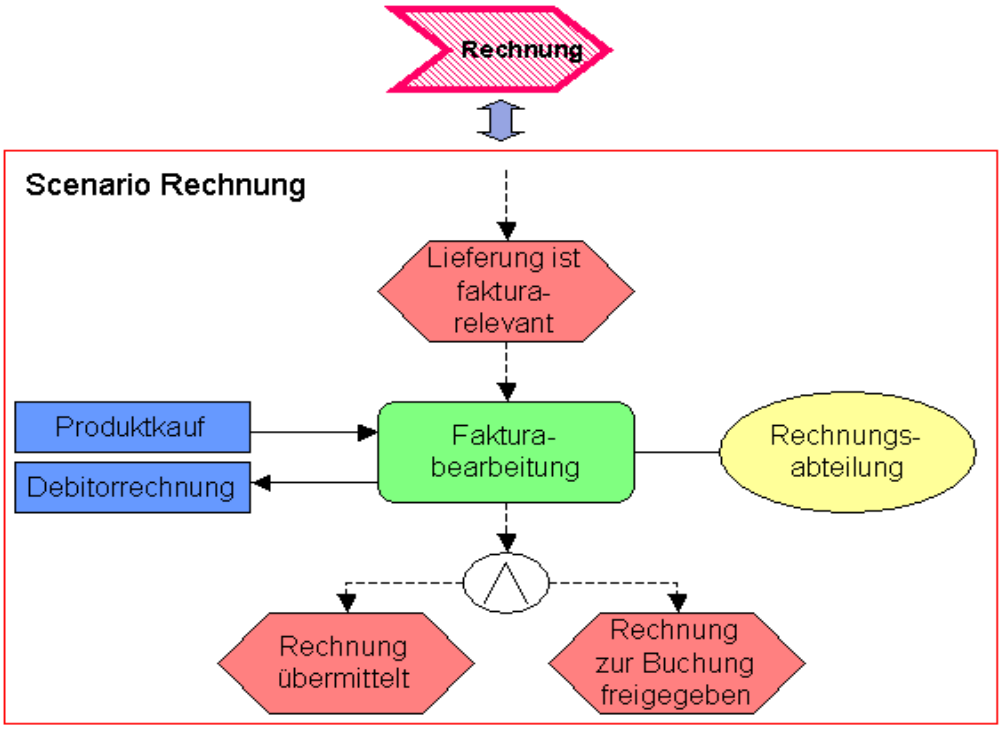

Abbildung 2.3: Zusammenhang von Wertschöpfungsketten und EPKs

Modellierung von Geschäftsszenarien durch ereignisgesteuerte Prozeßketten

Die *ereignisgesteuerten Prozeßketten (EPKs)* bestehen zur Beschreibung von Geschäftsszenarien aus vier Sichten (siehe Abbildung 2.4):

Komponentenmodell zur Visualisierung von Funktionsbäumen: „Was soll getan werden?“

Organisationsmodell zur Visualisierung der Organisationseinheiten: „Wer soll es tun?“

Informations-/Datenmodell zur Visualisierung der Informationen und den zugehörigen Daten: „Welche Informationen/Daten werden benötigt?“

Interaktionsmodell zur Visualisierung der Interaktion und des Informationsflusses zwischen den einzelnen Komponenten: „Wie ist die Interaktion bzw. der Informationsfluß?“

Als fünfte Komponente wird mit den bisherigen Modellen das *Ereignis* („Wann soll etwas getan werden?“) vermischt. Aus diesen vier Modellen und dem *Ereignis* ergibt sich das Modell der *ereignisgesteuerten Prozeßkette (EPK)*.

In Abbildung 2.5 sind alle Elemente der EPKs aufgelistet. Die logischen Operatoren ermöglichen das Verzweigen innerhalb eines Prozesses. Der *Prozeßwegweiser* deutet an, daß sich hinter diesem Szenario weitere Szenarien verbergen. Die *Pfeile* bzw. *Linien* für den Kontrollfluß, den Informations-/Materialfluß und die Zuordnung von Ressourcen/Organisationseinheiten

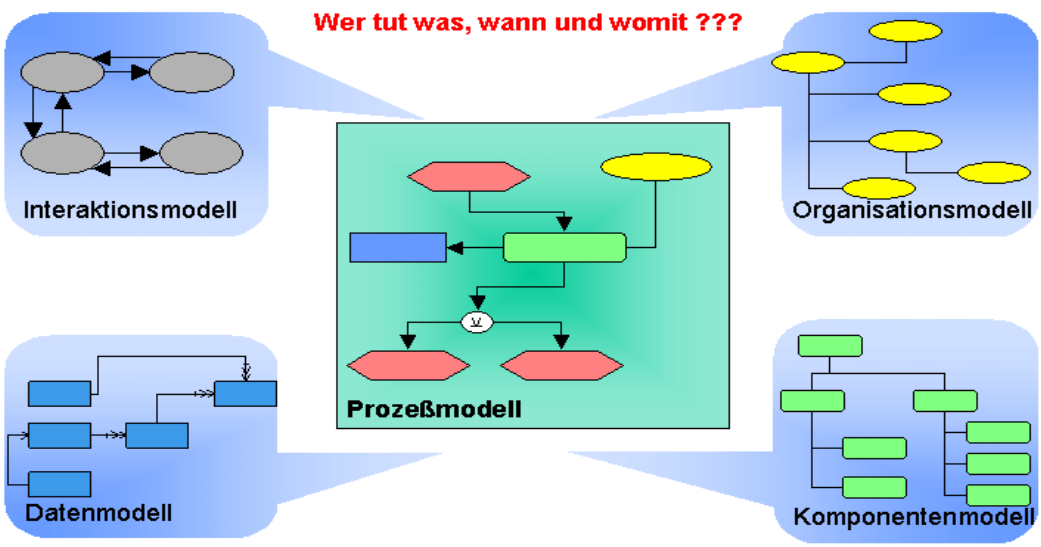

Abbildung 2.4: Ereignisgesteuerte Prozeßketten und ihre vier Sichten

unterscheiden sich durch die Art der Komponenten, die sie miteinander verbinden. Der Kontrollfluß existiert zwischen einem *Ereignis* und einer *Funktion*. Der Informations-/Materialfluß verbindet *Input/Output Objekte* mit einer *Funktion*. Die Zuordnung von Ressourcen/Organisationseinheiten verbindet eine *Organisationseinheit* entweder mit einer *Funktion* oder einem *Ereignis*[2].

Innerhalb der EPKs wird des weiteren zwischen einem Szenarioprozeß und einem Prozeß unterschieden:

- Ein Szenarioprozeß ist eine konkrete betriebswirtschaftliche Aufgabenstellung, die in einer zeitlich-logischen Form in einem EPK repräsentiert wird [SAP98d].
- Ein Prozeß beschreibt eine systemnahe Funktion des R/3 Systems und ihre Reihenfolge [SAP98d].

2.2 Adaption vs. Customizing

Die Hersteller von Standardsoftwaresystemen benutzen zumeist eine eigene Begriffswelt, die auf ihr spezifisches Produkt zugeschnitten ist. Die SAP AG definiert in ihrem Glossar [SAP98e] beispielsweise den Begriff *Customizing* wie folgt:

> Einstellungen, die der Systemverwalter bei der Systemeinführung vornehmen muß.
> Im Customizing können Sie
>
> - die SAP-Funktionalität in Ihrem Unternehmen schnell, sicher und kostengünstig einführen.

[2] Anmerkung: Der Grund für die unterschiedlichen Linienarten ist informeller Natur und dient ausschließlich der Visualisierung und dem besseren Verständnis. Es handelt sich somit um kein minimales Modell.

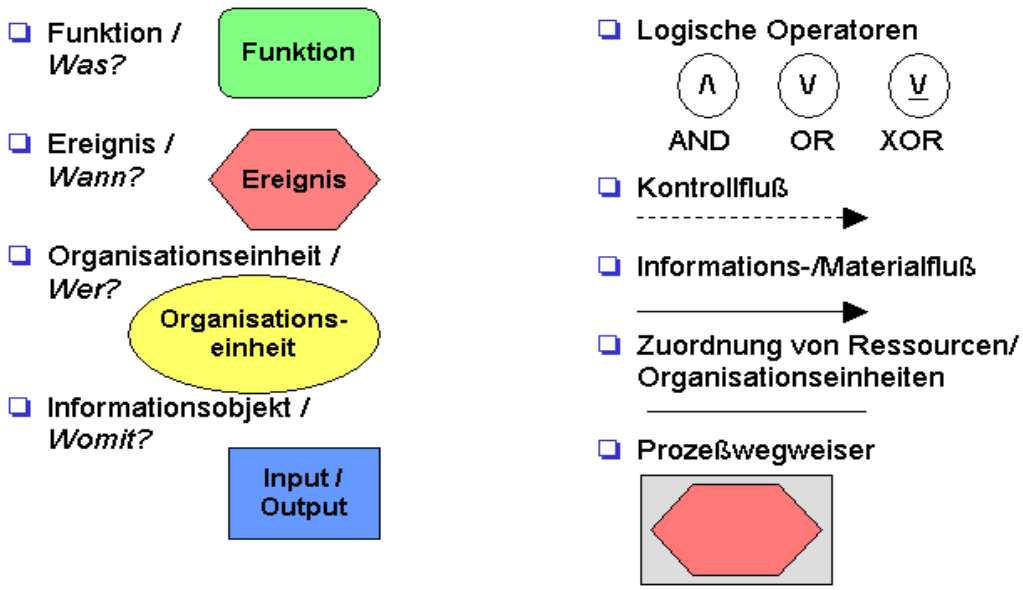

Abbildung 2.5: Elemente der ereignisgesteuerten Prozeßketten

- die unternehmensneutral ausgelieferte Funktionalität den spezifischen betriebswirtschaftlichen Anforderungen Ihres Unternehmens anpassen.
- die Phasen der Einführung und der Anpassung in einem einfachen Werkzeug für Projektsteuerung dokumentieren und verwalten.

Das Customizing ist vor Inbetriebnahme des Systems obligatorisch.

Aus dem ersten Satz dieser Definition geht bereits hervor, daß es sich beim *Customizing*, um Einstellungen handelt, die bei jeder Systemeinführung gemacht werden müssen. Innerhalb dieser Arbeit sollen jedoch nicht nur *Muß*-Einstellungen behandelt, sondern auch Einstellungen diskutiert werden, die nach einer Systemeinführung folgen können oder losgelöst von einer Systemeinführung sind. Des weiteren sind nicht nur die betriebswirtschaftlichen Aspekte für diese Arbeit von Belang, sondern insbesondere die Techniken für Systemeinstellungen und Erweiterungen.

Innerhalb des R/3 System befindet sich hinter der Transaktion *SPRO* der Einstieg in das *Customizing-Menü*, in welchem diese obligatorischen Einstellungen gemacht werden müssen.

Das *Lexikon Informatik und Datenverarbeitung* [Sch97] definiert *Customizing* folgendermaßen:

> Anpassung von *Standardsoftware* an individuelle Kundenwünsche. In Betracht kommen drei Möglichkeiten, und zwar Parametrisierung, Konfigurierung oder Individualprogrammierung.
> ...

Diese Definition ist wesentlich abstrakter, als die der SAP AG und abstrahiert von betriebswirtschaftlichen Spezifika. Die drei Möglichkeiten Parametrisierung, Konfigurierung und Individualprogrammierung werden im Laufe dieser Arbeit wiederholt verwendet.

Da innerhalb dieser Arbeit die einzelnen Techniken jedoch am Beispiel des R/3 Systems der SAP AG gezeigt werden, wird der Begriff der *Adaption* als ein abstrakterer Begriff für das Anpassen, Erweitern, etc. von Standardsoftware gewählt, um Mißverständnisse zu vermeiden. Der Begriff *Customizing*, der vom englischen Wort *Customer* (deutsch: Kunde) abstammt, ist des weiteren zu sehr auf Kundeneinstellungen bezogen, was der Abstraktion von den zu besprechenden Techniken und Methoden nicht genügt.

Unter dem Begriff *Adaption*, der vom lateinischem Verb *adaptare* (deutsch: anpassen) abgeleitet ist, sollen alle Arten der Anpassung, Konfiguration, Parametrisierung, Erweiterung, etc. verstanden werden. Ist im Folgenden also die Rede von *Adaption* bzw. *adaptieren*, so ist dies nicht auf eine bestimmte Standardsoftware bzw. auf den Jargon eines Standardsoftwareherstellers bezogen, sondern verallgemeinert zu verstehen.

2.3 Vorgehen bei der Einführung von betrieblichen Standardsoftwaresystemen

Projekte mit der Zielsetzung, eine Standardsoftware (zum Beispiel das R/3 System) einzuführen, ziehen sich über mehrere Monate beziehungsweise Jahre hin. Der Grund für die langwierige Einführung eines Standardsoftwaresystems liegt zum einen in der Komplexität des Systems und zum anderen in der *Art und Weise*, wie die Einführung geplant und ausgeführt wird begründet. Die Dauer der Einführung wird insbesondere durch die Anzahl der verwendeten Module beeinflußt. Wird nur ein Modul, zum Beispiel das R/3 Modul FI (Finanzwesen), eingeführt, so gestaltet sich das Projekt einfacher, als wenn mehrere Module (z.B. FI, CO[3], HR[4] und MM[5]) in einem Projekt integriert werden, da zwischen den Modulen Abhängigkeiten bestehen.

Die *Art und Weise* der Einführung, also welche Methoden, Modelle und Strategien verwendet werden, hängt zu meist von den begleitenden Beratern ab. Je nach Unternehmensberatung wurden verschiedene eigene Methoden entwickelt, um eine Einführung zu strukturieren und für den Kunden besser verständlich zu machen.

Die ASAP-Methode ist von der SAP AG entwickelt worden, um die Einführung zu beschleunigen und dem Kunden mehr Einsicht in das Einführungsprojekt zu ermöglichen. In ASAP stecken Erfahrungen der SAP AG und ihrer Partnern aus vielen Projekten und Praxisbeispielen. Die erfolgreichsten Methoden, die zu einer schnelleren und strukturierteren Einführung geführt haben, wurden in ASAP zusammengefaßt [SAP00a]. ASAP wurde zunächst in den USA entwickelt und bei R/3 Einführungen eingesetzt und ist mittlerweile die *weltweite Standardeinführungsmethode* [BE99] für R/3. Die SAP AG nutzt die gewollte Doppeldeutigkeit von ASAP mit *AcceleratedSAP* und *„As Soon As Possible“*, um ihrer Methode zur Einführung einer Standardsoftware zusätzlichen Ausdruck zu verleihen.

Ein wesentliches Merkmal von ASAP ist die Sicht auf die Einführung der Standardsoftware. Der Fokus liegt nicht auf dem R/3 Referenzmodell und seinen Prozessen, sondern auf dem Unternehmen. Das Unternehmen entwirft zunächst seine Versionen der zu unterstützenden

[3] CO - Controlling
[4] HR - Human Resources (Personalwirtschaft)
[5] MM - Material Management (Materialwesen)

Geschäftsprozesse und Organisationsstrukturen [BE99]; anschließend wird aus den sogenannten *„Best Business Practices“* das Sollkonzept entwickelt.

Neben der Einführung unterstützt ASAP auch bei einem *R/3 Upgrade*. Im Folgenden wird nur auf die Möglichkeiten von ASAP, die Einführung zu unterstützen, eingegangen. Für Informationen zu dem Nutzen von ASAP für ein *R/3 Upgrade* wird auf die in diesem Kapitel referenzierte Literatur verwiesen.

Der Nutzen von ASAP für den Kunden wird von der SAP AG unter anderem durch die Beantwortung der folgenden Fragen beschrieben [SAP00a]:

- Wieviel wird die R/3-Einführung kosten?
- Wie lange wird die R/3-Einführung dauern?
- Wie kann ich eine hohe Qualität sicherstellen?
- Welche Werkzeuge helfen mir bei der R/3-Einführung?
- Welche Ressourcen benötige ich?

Die Frage der Kosten wird unter anderem durch das *Quicksizing* (siehe Kapitel 2.3.2) bezüglich der Hardwarekosten geklärt. Die Softwarekosten ergeben sich aus den Lizenzgebühren, die je nach Modulen und Arbeitsplätzen zu Beginn des Projektes ermittelt werden. Die Personalkosten (Zahlungen an die Unternehmensberatung, etc.) ergeben sich wiederum unter anderem aus der Dauer des Einführungsprojektes und der Anzahl der eingesetzten Berater.

Qualitätssicherung zieht sich durch alle Phasen des Projektes und wird auch nach Beendigung der Einführung durch stetige Erweiterungen (*Hot-Packages, Legal Changes Patches*) unterstützt. Die Frage der zur Verfügung stehenden Hilfsmittel und Werkzeuge wird in Kapitel 2.3.2 beantwortet. Die Einführungsdauer und benötigten Ressourcen sollen bereits zu Beginn der Einführung mit ASAP berechnet werden. In der Literatur [BE99, Bra99] finden sich zwar Zeitpläne für die einzelnen Phasen, die insgesamt eine Einführungdauer von ca. neun Monaten angeben, jedoch zeigt die Erfahrung der SAP AG [SAP00b], daß kleine Einführungsprojekte mit zwei Monaten auskommen können, größere Projekte jedoch zweistellige Monatszahlen benötigen. Im Folgenden werden die Phasen der Einführung mit ASAP und die zugehörigen Werkzeuge beschrieben.

2.3.1 Die Phasen der strukturierten Einführung mit ASAP

Die einzelnen Phasen einer Einführung sind in der *ASAP Roadmap* (siehe Abbildung 2.6) zusammengefaßt. Diese „Landkarte“ enthält die einzelnen Stationen, die während des Einführungsprojektes durchlaufen werden. Im Folgenden werden die fünf Phasen und die *Abschlußphase* grob skizziert. Es wird nicht auf den Inhalt der einzelnen Schulungen zu den entsprechenden Phasen eingegangen. Für den Inhalt dieser Schulungen sei auf den *AcceleratedSAP Implementation Assistant* [SAP99b] verwiesen.

Abbildung 2.6: Die ASAP Roadmap [SAP99b]

Phase 1: Projektvorbereitung

Die Projektvorbereitungsphase dient der groben Planung und Vorbereitung des Einführungsprojektes. Die Projektmitglieder müssen festgelegt werden und die ersten Aufgaben werden den einzelnen Personen zugeteilt. In der Literatur [Bra99] finden sich bereits detaillierte Aufgaben und Rollen, wie zum Beispiel Projektleiter, Systemadministrator, Datenbankadministrator oder Berechtigungsadministrator. Erste Schulungen für das Projektteam in Bezug auf das Standardsoftwaresystem und auch allgemeine Managementschulungen sind ebenso in der ersten Phase vorgesehen.

Entscheidend in der ersten Phase ist die Festlegung der Einführungsstrategie. Die Ziele des Unternehmens, die mit der Einführung von einer Standardsoftware erreicht werden sollen, müssen festgelegt sein. Sollen zum Beispiel die Transportkosten gesenkt werden, so stellt dies andere Anforderungen an die kommenden Phasen, als wenn ein verbesserter Kundendienst angestrebt wird.

Ein interessanter Punkt ist die frühzeitige Planung für die Schulungen der Anwender und die Einrichtung eines *Help-Desks*, um Anwenderprobleme zu beantworten. Ebenso soll das Projekt kein *Geheimnis* im Unternehmen sein, sondern alle Mitarbeiter sollen im Rahmen eines *Kickoff-Meetings* über das Projekt informiert werden.

Die einzelnen Schritte einer Phase sind in sogenannte Arbeitspakete unterteilt, die sich wiederum in einzelne Aktivitäten aufteilen. Ein Arbeitspaket soll als Ergebnis immer ein Dokument haben und ist derart verteilt, daß nur bestimmte Rollen zur Erstellung eines Ar-

beitspakets benötigt werden. Abschließend erfolgt, wie auch in allen folgenden Phasen, eine Qualitätsprüfung, bei der alle Daten gesammelt und auf ihre Richtigkeit überprüft werden sollen.

Phase 2: Business Blueprint

In Phase zwei wird das Sollkonzept in Form des *Business Blueprints* entwickelt. Im *Business Blueprint* (siehe auch Kapitel 2.1.2) sind unter anderem die Geschäftsprozesse des Unternehmens festgehalten. Hierzu muß zuerst ein Verständnis der Berater für die Unternehmensabläufe entwickelt werden, um anschließend mit dem Unternehmen den *Business Blueprint* auszuarbeiten.

Ebenso sind die Organisationsstrukturen des Unternehmens in den *Business Blueprint* aufzunehmen. Durch diesen *Business Blueprint* wird die Funktionalität des Standardsoftwaresystems eingegrenzt, so daß die branchenspezifischen Gegebenheiten ausgewählt werden.

Mit diesem *Business Blueprint* wird das erste System aufgesetzt. Dieses System dient im Verlauf des weiteren Prozesses als Entwicklungssystem und als Spielmandant für Schulungen und Testzwecke.

Phase 3: Realisierung

Zunächst werden in der Realisierungsphase die Einstellungen des Entwicklungssystems getestet. Ziel ist ein sogenanntes *Baseline-System*[6] zu erhalten, welches 80% des Tagesgeschäftes eines Unternehmens abbilden soll. Die eigentliche Qualitätssicherung wird auf einem zweiten Standardsoftwaresystem, dem Qualitätssicherungssystem, durchgeführt, wobei ein erster Transport der Einstellungen aus dem Entwicklungssystem in das Qualitätssicherungssystem vollzogen wird.

Zur Realisierungsphase gehört ebenso das Planen und Implementieren von eigenen ABAP-Entwicklungen, Datenübernahmeprogrammen, Schnittstellenprogrammen, Berechtigungskonzepten, etc. Die Realisierungsphase ist somit die Hauptphase für die Einstellungen direkt im System.

Im nächsten Schritt werden die restlichen 20% in das Qualitätssicherungssystem implementiert. Hierbei handelt es sich zumeist um unternehmensspezifische Sonderfälle, die gegebenenfalls durch *User-* und *Customer-Exits* (siehe auch Kapitel 5.2) in das System eingebaut werden, sowie um Geschäftsprozesse, denen eine nur geringe Bedeutung zukommt. Am Schluß der Realisierungsphase wird aus den Systemdaten des Qualitätssicherungssystems das Produktivsystem aufgesetzt.

[6] Der Begriff Baseline-System wird in der Literatur[BE99, Bra99, SAP00a] sowohl für das Entwicklungssystem als auch für das Qualitätssicherungssystem genutzt. Einheitlich ist jedoch der Implemtierungsgrad von 80%.

Phase 4: Produktionsvorbereitung

Vor dem tatsächlichen Produktivbetrieb des Standardsoftwaresystems müssen die Einstellungen am System erneut überprüft werden; dies betrifft insbesondere die wichtigsten Geschäftsprozesse. Performanz-Tests müssen durchgeführt werden, um zu verifizieren, ob das *Quicksizing* eine Konfiguration empfohlen hat, die für den täglichen Betrieb ausreichend ist.

Schulungen für die Anwender müssen ebenso wie die Entwicklung einer Strategie für die Datenübernahme durchgeführt werden. Die Einrichtung des *Help-Desks* muß erfolgen, um nach dem Produktivstart möglichst schnell auf Anwenderprobleme reagieren zu können.

Der letzte Schritt vor dem Produktivstart ist die Übernahme der Daten aus dem Altsystem.

Phase 5: Go Live & Support

Der *Help-Desk* wird zu Beginn der fünften Phase stärker belastet sein, als zu einem späteren Zeitpunkt. Eine langfristige Planung für eine Anwenderunterstützung muß festlegen, wie die Anwender im Anschluß betreut werden sollen. Des weiteren gibt das Überwachen der Systemleistung zu Beginn des Produktivbetriebes Aufschluß über mögliche Änderungen an der Systemlandschaft, um Performanzprobleme zu bewältigen, die nicht vorhersehbar waren beziehungsweise nicht erkannt wurden.

Der *Support* aus dem Titel der fünften Phase bezieht sich nicht nur auf die Anwender, sondern auch auf das Standardsoftwaresystem selbst. Mit dem *Online Service System (OSS)* können Lösungen ab- bzw. angefragt und aktuelle *Hot-Packages* beziehungsweise *Legal Changes Patches* bezogen werden. Die Einrichtung von *EarlyWatch* erlaubt das ferne Überwachen der Installation während des Produktivbetriebes. *EarlyWatch* dient der Konfiguration von Systemparametern zur Leistungsoptimierung und Erhöhen der Verfügbarkeit.

Im Rahmen eines *Review Workshops* sollen die Erfahrungen des Projektes gesammelt werden, um späteren Projekten (Upgrade, Verteilung) Nutzen zu bringen. Das Aufdecken von möglichen Risikofaktoren, die sich in der ersten Zeit des Produktivbetriebes herauskristallisiert haben, ermöglicht ein frühes Reagieren.

Stetige Optimierung

Stetige Optimierung betrifft fortlaufende Schulungen für neue Anwender und Anwender, deren Arbeitsumfeld sich ändert. Auch die Definition von fortlaufenden Systemvorgängen[7], um Fehler und Risikofaktoren aufzudecken, gehört zur Phase der stetigen Optimierung.

Das *Einspielen* von Systemupgrades (Hot-Packages, Legal Changes Patches) muß auch mit der Einweisung in neue Funktionalitäten verbunden sein beziehungsweise müssen Änderungen den entsprechenden Anwendern mitgeteilt werden.

[7] Unter fortlaufenden Systemvorgängen werden alle Arten des Überwachen (*Monitoring*) des Systems verstanden

2.3.2 Werkzeuge zur Unterstützung des Einführungsprozesses

Die SAP AG liefert inzwischen zu jedem R/3 Softwarepaket eine ASAP CD-ROM mit aus. Auf dieser CD sind verschiedene Werkzeuge (*Tools*) enthalten, die die Einführung mit ASAP unterstützen.

Implementation Assistant

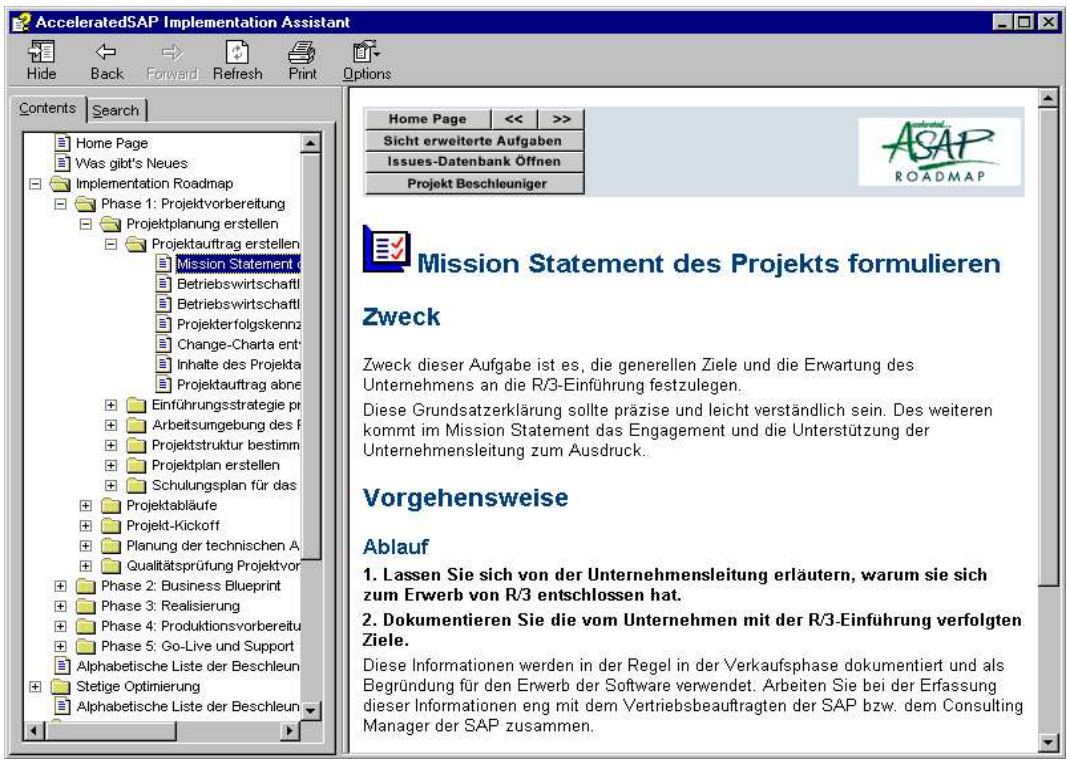

Abbildung 2.7: Der ASAP „Implementation Assistant“ [SAP99b]

Der *Implementation Assistant* (vgl. Abbildung 2.7)ist die übergeordnete Anwendung der ASAP Werkzeuge. Enthalten sind unter anderem die ASAP *Roadmap*, mit einer detaillierten Beschreibung der einzelnen Arbeitspakete und Abläufe der einzelnen Phasen, das ASAP Glossar und die verschiedenen Verweise auf Microsoft Word-Dokumente, Excel-Tabellen, Power-Point-Folien und Projekt-Vorlagen. In den Beschreibungen der einzelnen Aufgaben sind Verknüpfungen zu Vorlagen für Projektdokumente und zu verwenden ASAP Werkzeuge enthalten.

Question and Answer database (Q&Adb)

Die Fragen der Q&Adb (vgl. Abbildung 2.8) sollen von den Projektmitgliedern sukzessive beantwortet werden und sind die Grundlage für den *Business Blueprint*. Die Q&Adb generiert aus den Antworten die *Business Process Master List* in Form einer Microsoft Excel-Tabelle,

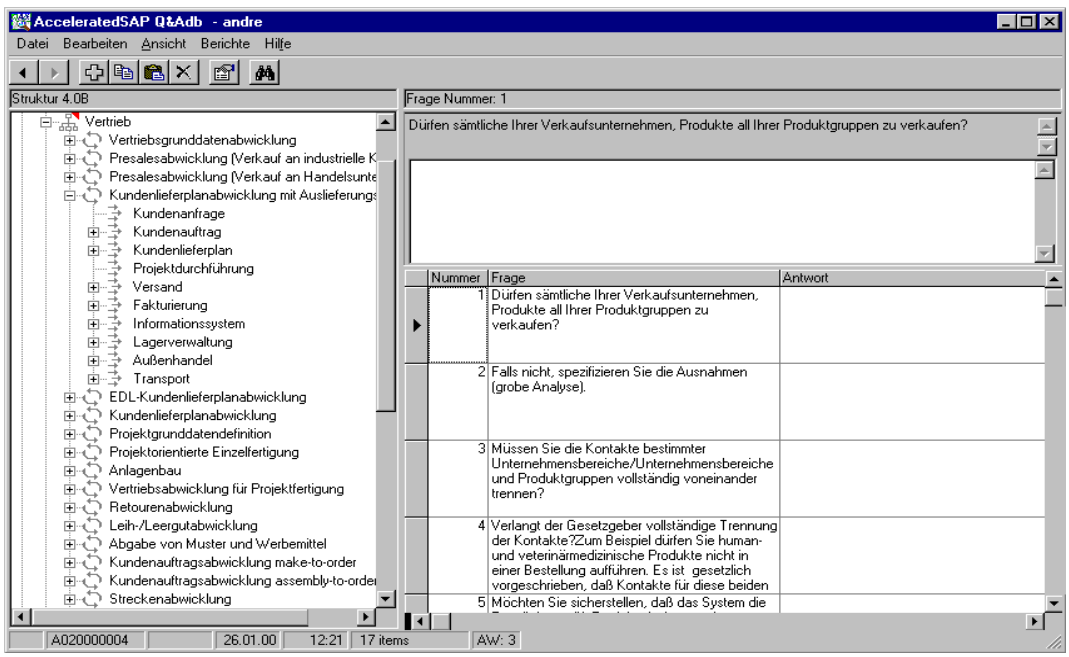

Abbildung 2.8: Die ASAP „Question and Answer database“ [SAP99c]

in der die Geschäftsprozesse enthalten sind. Innerhalb der Q&Adb können zu den einzelnen Prozessen *EPKs* angezeigt werden, wobei nicht der SAPGUI, sondern ein Programm eines Drittanbieters genutzt wird, um das Diagramm anzuzeigen. Änderungen können an diesen Diagrammen vorgenommen und gespeichert werden, welches mit dem SAPGUI nicht möglich wäre, da der *Business Navigator* diese Option nicht bietet.

Concept Check Tool

Das *Concept Check Tool* (vgl. Abbildung 2.9) soll die ersten beiden Phasen von ASAP (siehe Kapitel 2.3.1) unterstützen. Wie in Abbildung 2.9 zu sehen ist, werden dem Sachbearbeiter Fragen gestellt, die oftmals betriebswirtschaftlichen Inhalt haben und bereits vor der Einführung das System eingrenzen können und erste Einstellungen (beispielsweise Mandanten) erfassen. Eine Verknüpfung zum *Quicksizing* in das *SAPNet* ermöglicht bereits frühzeitig ein erstes Abschätzen der benötigten Hardware-Ressourcen, die anhand der entsprechenden Anwenderzahlen pro Modul ermittelt werden.

Die einzelnen Fragen des *Concept Check Tools* werden dynamisch gestellt, und stehen in Abhängigkeit von den bisher gegebenen Antworten. Im weiteren Vorgehen werden somit die relevanten Fragen herausgefiltert.

Alle Methoden und ASAP Werkzeuge haben das gemeinsame Ziel, die Einführung der Standardsoftware zu beschleunigen und zu standardisieren. Hilfreich kann die Nutzung von ASAP insbesondere bei der Zusammenarbeit mehrerer Beratungsgesellschaften sein, da alle Projektmitarbeiter auf einer gemeinsamen Basis aufbauen können und die Vorgehensweise allen bekannt ist.

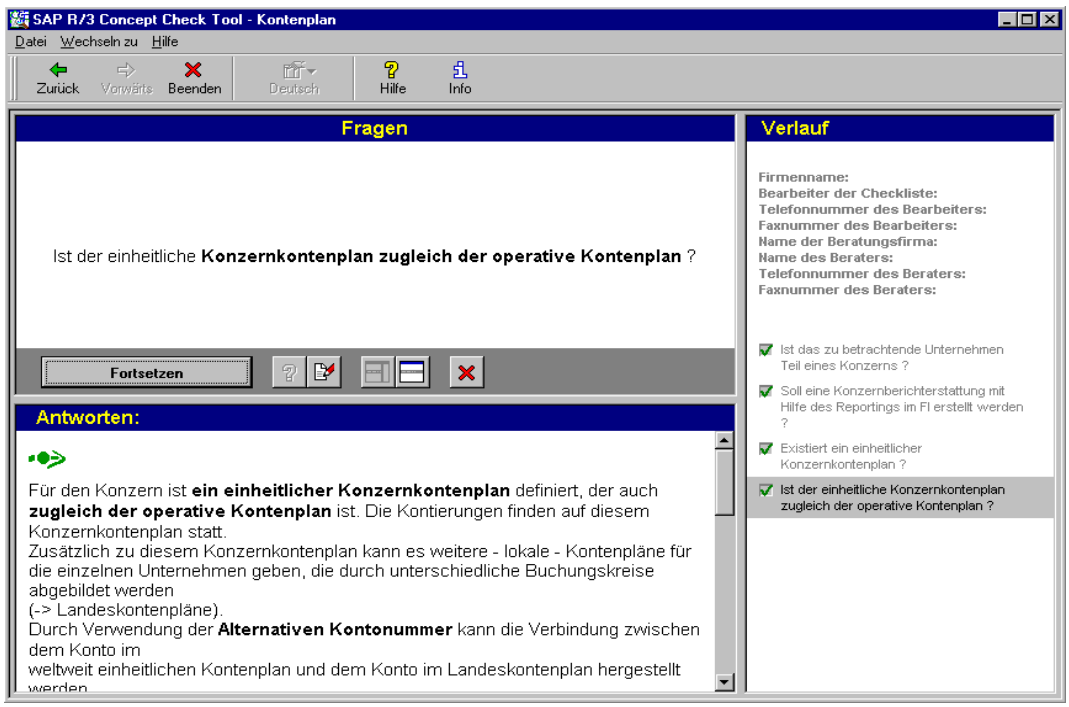

Abbildung 2.9: Das ASAP „Concept Check Tool“ [SAP99a]

Vorteile können sich ergeben, wenn das Unternehmen vor Beginn der Einführung bereits mit der Q&Adb und dem *Concept Check Tool* eine IST-Aufstellung macht und den Beratern eine Grundlage gibt, um möglichst schnell in die Unternehmensstrategie und Geschäftsabläufe einzusteigen.

Kapitel 3

Architekturelle Klassifikation von Adaptionskonzepten

Für die Gliederung der weiteren Arbeit wird in diesem Kapitel die Grundlage gelegt. Die verschiedenen Adaptionskonzepte werden zunächst anhand der Softwarearchitektur von einer Standardsoftware klassifiziert, um anschließend zu einer von der Softwarearchitektur losgelösten Klassifikation für Adaptionskonzepte zu gelangen. Hierbei werden zunächst die Adaptionskonzepte charakterisiert und anschließend eine dreistufige Klassifikation entwickelt.

Einleitend wird exemplarisch die Softwarearchitektur einer betrieblichen Standardsoftware beschrieben (Kapitel 3.1). Im Anschluß werden die Ansätze zur Adaption anhand der Softwarearchitektur in Kapitel 3.2 aufgezeigt, um abschließend in Kapitel 3.3 die Adaptionskonzepte zu charakterisieren und zu klassifizieren.

3.1 Architektur betrieblicher Standardsoftwaresysteme

Mehrschichtige Softwarearchitekturen sind hierarchisch aufgebaut [SG96]. Jede Schicht fungiert gegenüber der nächsthöheren Schicht als Dienstanbieter (Server) und gegenüber der darunterliegenden Schicht als Dienstnehmer (Client).

Betriebliche Standardsoftwaresysteme wie R/3 bestehen aus drei solcher Schichten. Die unterste Schicht, die Datenbank, dient der persistenten Datenhaltung. Die mittlere Schicht, die Applikationsebene, stellt der nächsthöheren Schicht Anwendungen zur Verfügung, um Daten zu speichern, und bereitet diese Änderungen für die Datenbank auf. Die Präsentationsebene ist die äußerste Schicht und nutzt die von der Applikationsebene zur Verfügung gestellten Anwendungen, um Daten zu präsentieren.

Diese dreistufige Client/Server-Architektur ermöglicht die beliebige Verteilung der einzelnen Schichten auf verschiedene Peripheriegeräte. Ein zentrales System, auf dem alle Ebenen zusammengefaßt sind, ist ebenso möglich, wie die Verteilung der Ebenen auf verschiedene Rechner. Wie die einzelnen Schichten verteilt werden können, ist in Abbildung 3.1 dargestellt.

Beispielsweise können in einem R/3 System theoretisch bis zu 98 Applikationsserver betrieben

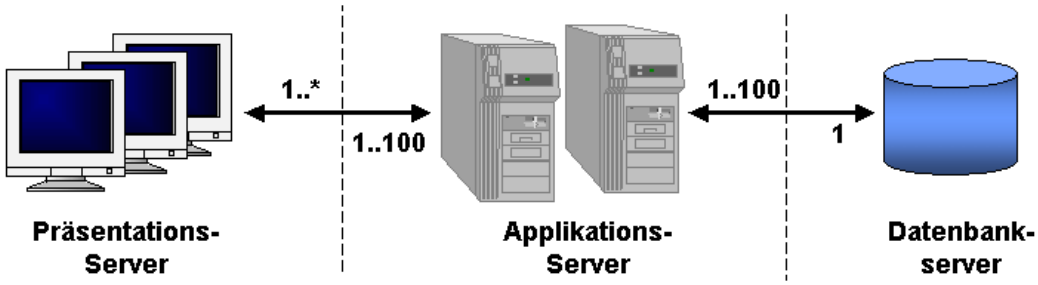

Abbildung 3.1: Client/Server-Architektur des R/3 Systems

werden. Die Zahl 98 erklärt sich durch die Tatsache, daß jeder Applikationsserver durch eine zweistellige Nummer eindeutig identifiziert werden kann und die Nummern 98 und 99 durch den EPS[1] bzw. SAP-Router reserviert sind.

Die Anzahl der Präsentationsserver ist unbegrenzt, jedoch muß die Zahl der Benutzer, insbesondere die Zahl der *gleichzeitig arbeitenden* Benutzer *(concurrent users)*, vor der Installation bekannt sein, da dies bei der Verteilung der Applikationsserver und der Größe des Datenbankservers ein entscheidenden Faktor ist.

Im Gegensatz zu Präsentations- und Applikationsservern ist in einem R/3 System immer genau ein Datenbankserver zu finden. Die Restriktion eines Datenbankservers in einem R/3 System kann bei einer Installation zum Flaschenhals werden, so daß gerade die Größe eines Datenbankservers und auch die angeschlossene Infrastruktur für den Betrieb eines R/3 Systems wichtige Faktoren im Hinblick auf die Performanz sind.

Die Anzahl der Applikationsserver kann zwar an die spezifischen Bedürfnisse angepaßt werden, jedoch kann ein einzelner Work-Prozeß immer nur auf einem Applikationsserver ausgeführt werden, so daß es bei größeren Prozessen (z.B. Jahresabschluß) auch hier zu einem Engpaß kommen kann.

3.1.1 Persistente Datenhaltung - Datenbankserver

Um Daten persistent zu speichern, nutzt ein R/3 System die Dienste eines relationalen Datenbanksystems[2] als Datenbankserver. In dieser relationalen Datenbank werden jedoch nicht nur Anwendungsdaten in Tabellen abgelegt, sondern zum Beispiel auch Anwendungsprogramme und Systemsteuerungsinformationen.

Der Zugriff auf die Datenbank erfolgt über SQL[3], so daß verschiedene Datenbanken unterschiedlicher Hersteller angesprochen werden können. Zu den von der SAP AG zertifizierten Datenbanksystemen zählen (zum Zeitpunkt der Ausarbeitung dieser Arbeit):

- Oracle

[1] EPS - External Project Software

[2] In diesem Kapitel kann der Begriff *Datenbanksystem* jederzeit durch *Datenbankmanagementsystem* ersetzt werden. Die beiden Begriffe werden nicht weiter unterschieden.

[3] SQL - Structured Query Language

- Informix
- DB2 von IBM
- SQL-Server von Microsoft
- SAP DB (ursprünglich Adabas D)

Da jedoch die einzelnen Datenbanksysteme eigene SQL-Dialekte haben, die teilweise mit zusätzlicher Funktionalität ausgestattet sind, wurden in ABAP zwei sogenannte *SQL-Levels* integriert [BE99]:

Native SQL erlaubt die Nutzung der vollen Funktionalität der unterliegenden Datenbank. Soll ein R/3 System von einer Datenbank auf eine andere Datenbank[4] portiert werden, so sind die *Native SQL* Anweisungen auf Kompatibilität zu prüfen.

Open SQL garantiert die Kompatibilität zu allen R/3 unterstützten Datenbanken. Um dies zu gewährleisten, wird eine Anweisung, die in *Open SQL* geschrieben wurde, von der R/3 eigenen Datenbankschnittstelle in *Native SQL* übersetzt (vgl. Abbildung 3.2).

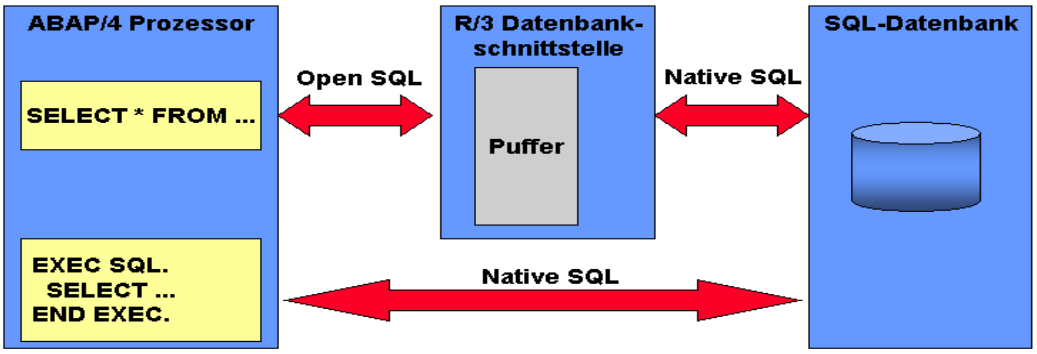

Abbildung 3.2: Open vs. Native SQL

Daß die Nutzung von *Native SQL* Anweisungen in ABAP ermöglicht wird, ist ausschließlich auf die gegebenenfalls erhöhte Performanz und Abwärtskompatibilität zu alten Versionen zurückzuführen.

Greift ein R/3 System auf die Datenbank zu, so nutzt das System den Benutzer *SAPR3*. Dieser Benutzer ist Eigentümer aller Datenbanktabellen, die vom R/3 System angelegt werden. Wie die Benutzerverwaltung in einem R/3 System abgebildet wird, findet sich in Kapitel 4.4.

3.1.2 Applikationsebene - Applikationsserver

Der Applikationsserver wird von der SAP auch als Basissystem bezeichnet, da der Applikationsserver die ausführende Instanz ist, um Änderungen in die Datenbank zu schreiben und

[4] Die Portierung bezieht sich auf Datenbankmanagementsysteme unterschiedliche Hersteller, die untereinander nicht vollständig kompatibel sind.

sämtliche betriebswirtschaftliche Funktionalität des R/3 System bereitstellt. Die auf jedem Applikationsserver installierte ABAP VM[5] interpretiert die R/3-Anwendungsprogramme, die in der R/3 eigenen Sprache ABAP erstellt wurden.

Der zentrale Prozeß eines Applikationsservers ist der sogenannte SAP-Dispatcher *(dispatch= verteilen)*. Dieser Dispatcher existiert genau einmal auf jedem Applikationsserver und verwaltet die Systemspeicherressourcen, organisiert die Kommunikation zwischen Prozessen und erzeugt eine variable Anzahl von Work-Prozessen. Für die einzelnen Work-Prozesse existieren die folgenden Ausprägungen:

Dialog-Work-Prozesse bearbeiten die verschiedenen Benutzerdialoge. Hierbei kann ein Work-Prozeß für verschiedene Benutzerdialoge verantwortlich sein. Die Verteilung der Benutzerdialoge auf die verschiedenen Work-Prozesse übernimmt der Dispatcher.

Hintergrund-Work-Prozesse (Batch-Work-Prozesse) benutzen zwar teilweise die gleichen Programme wie *Dialog-Work-Prozesse*, jedoch ist die Bearbeitung für das System nicht so aufwendig, da die explizite Dialogverarbeitung (z.B. Aufbau von Bildschirmbildern) entfällt. *Hintergrund-Work-Prozesse* werden durch das CCMS[6] gesteuert.

Sperrverwaltungs-Work-Prozeß (Enqueue-Work-Prozeß) hält zum Sperren von Objekten eine Sperrtabelle im Hauptspeicher des Applikationsservers. Das R/3 System nutzt nicht die Sperrmechanismen der Datenbank, da es zum Beispiel nicht möglich wäre, betriebswirtschaftliche Objekte über alle Schritte einer Wertschöpfungskette zu reservieren. Der *Enqueue-Dienst* kann wahlweise auf dem Applikationsserver oder dem Datenbankserver installiert werden und kann bei großen Installationen auch mehrfach in einem R/3 System eingesetzt werden.

Verbuchungs-Work-Prozesse ermöglichen das asynchrone Verbuchen[7] von Daten. Im R/3 System ist sowohl asynchrones als auch synchrones Verbuchen möglich. Beim synchronen Verbuchen führt der Dialog-Work-Prozeß die Änderungen in der Datenbank durch und der Benutzer muß auf das Ende der Transaktion warten. Beim asynchronen Verbuchen hingegen übergibt der *Dialog-Work-Prozeß* den Auftrag an einen *Verbuchungs-Work-Prozeß*, der die Änderungen in die Datenbank schreibt. Zusätzlich wird beim asynchronen Verbuchen zwischen V1 und V2 Buchungen unterschieden. V1 Buchungen, die primären Verbuchungskomponenten, werden mit höherer Priorität bearbeitet als V2 Buchungen, die sekundären Verbuchungskomponenten[8].

Spool-Work-Prozesse übergeben gepufferte Daten an Ausgabegeräte (z.B. Fax-Geräte, Drukker)

Ein weiterer eigenständiger Prozeß neben dem Dispatcher ist der *Message-Dienst*. Im Gegensatz zu allen anderen Prozessen und Work-Prozessen, existiert in einem R/3 System genau ein *Message-Server-Prozeß*. Alle Applikationsserver melden ihre Dienste am *Message-Dienst* an. Meldet sich ein Benutzer an einem R/3 System an, so leitet der *Message-Dienst* diese Anfrage an einen Work-Prozeß weiter. Selbige Weiterleitung erfolgt auch bei Sperranfragen, die

[5] ABAP VM - ABAP Virtual Machine (siehe [BE99])
[6] CCMS - Computing Center Management System
[7] Das Verbuchen meint das Schreiben der Änderungen einer Transaktion in die Datenbank.
[8] Für eine detaillierte Beschreibung sei auf [BE99] oder [CDI98] verwiesen.

der *Message-Dienst* an einen *Enqueue-Work-Prozeß* weiterleitet. Zusätzlich ist der *Message-Dienst* auch für die Lastverteilung auf den Applikationsservern verantwortlich und delegiert die Anfragen unter Berücksichtigung der momentanen Auslastung der verschiedenen Applikationsserver.

Für diese Arbeit ist von besonderer Relevanz der *Gateway-Service*, der die Kommunikation zwischen dem R/3 System und anderen Systemen steuert. Der *Gateway-Service* kann entweder vom SAP-Dispatcher gestartet werden oder als eigenständiger Systemprozeß laufen.

Soll ein R/3 System aus einem externen System angesprochen werden, so empfängt zunächst der *Gateway-Service* diese Anfrage und leitet sie per TCP/IP an das R/3 System weiter. Bei den externen Systemen kann es sich um andere R/3 Systeme, R/2 Systeme oder Fremdsysteme handeln. Das Gateway besteht aus den folgenden drei Prozessen:

Der Gateway-Leser mit dem Programm *gwrd* bzw. *gwrd.exe* ist der Hauptprozeß des *Gateway-Dienstes*. Der *Gateway-Leser* verwaltet die *Gateway-Work-Prozesse* und nimmt zunächst alle CPI-C[9] Aufträge an. Bei Verbindungen über TCP/IP kann der Auftrag direkt vom *Gateway-Leser* bearbeitet werden. Kommt der Auftrag über LU6.2[10] oder DCAM[11] an, so leitet er diesen Auftrag an einen *Gateway-Work-Prozeß* weiter.

Der Gateway-Work-Prozeß mit dem Programm *gwwp* bzw. *gwwp.exe* bearbeitet Verbindungen, die entweder über LU6.2 oder DCAM aufgebaut werden sollen.

Der Gateway-Monitor mit dem Programm *gwmon* bzw. *gwmon.exe* analysiert und administriert das Gateway. Der Monitor kann sowohl über eine Konsole als auch über den SAP-GUI (Transaktion *smgw*) aufgerufen werden.

3.1.3 Präsentationsebene - Präsentationsserver

Für die Kommunikation zwischen dem Benutzer und dem R/3 System wurde der SAPGUI[12] entwickelt. Der SAPGUI als Präsentationsserver erhält vom Applikationsserver die Daten, die vom Benutzer selektiert wurden. Zusätzlich werden auch Daten zum Bildschirmaufbau mitgesendet, so daß auch Informationen zur Aufbereitung der selektierten Daten übertragen werden.

Die Daten zur grafischen Darstellung, die zwischen dem Präsentations- und Applikationsserver übertragen werden, sind plattformunabhängig. Erst der Präsentationsserver nutzt die umgebungspezifischen Möglichkeiten, um das tatsächliche Bildschirmbild zu erzeugen. Der SAP-GUI steht zum Beispiel für *MOTIF*, *OS/2 Presentation Manager*, *Macintosh* und *Windows-Plattformen* zur Verfügung.

Neben dem SAPGUI existieren weitere Möglichkeiten, mit dem R/3 System zu agieren. Über elektronische Formulare und MAPI[13] ist es möglich, mit Hilfe eines Mail-Clients Zugriff auf Business Workflows zu bekommen. Durch den ITS[14] (siehe Kapitel 5.5) ist der Zugang zum R/3 System via Webbrowsers möglich.

[9] CPI-C (Common Programming Interface for Communication) siehe Kapitel 6.2.1
[10] LU6.2 ist Teil der SNA und wurde von IBM entwickelt.
[11] DCAM (Data Communication Access Method) wird unter BS2000 von Siemens verwendet.
[12] SAPGUI - SAP Graphical User Interface
[13] MAPI - Messaging Application Programming Interface
[14] ITS - Internet Transaction Server

3.2 Ansätze zur Adaption in der Systemarchitektur

Als Grundlage für Ansätze zur Adaption in der Systemarchitektur wird eine dreistufige Architektur mit den Ebenen Präsentation, Applikation und Datenbank angenommen (vgl. Kapitel 3.1). Es bieten sich somit drei Ansatzpunkte, die Standardsoftware zu adaptieren:

1. Adaption auf dem Präsentationsserver
2. Adaption auf dem Applikationsserver
3. Adaption auf dem Datenbankserver

Adaption mittels des Präsentationsservers unterteilt sich in zwei Kategorien: Zum einen kann die Funktionalität des Präsentationsservers genutzt werden, um die Standardsoftware zu adaptieren, zum anderen können Schnittstellen des Präsentationsservers eine Adaption ermöglichen. Wird die Adaption mittels der Funktionalität des Präsentationsservers durchgeführt, so werden die Dienste des Applikationsserver genutzt, um zum Beispiel Änderungen in Datenbanktabellen durchzuführen oder neue Benutzer in dem Standardsoftwaresystem anzulegen. Stellt der Präsentationsserver zusätzliche Schnittstellen zur Verfügung, so können beispielsweise Tabellenkalkulationen oder Textverarbeitungssysteme angesprochen werden. Je nach Ausprägung dieser Schnittstellen ist auch die Anbindung eines externen Programmes möglich.

Applikationsserver stellen zumeist keine eigene Funktionalität zur Verfügung, um Adaptionen zu ermöglichen. Ein Applikationsserver besteht im Allgemeinen aus einer variablen Anzahl von Prozessen, die als Dienstanbieter fungieren. Adaption auf der Ebene des Applikationsservers wird somit durch Schnittstellen ermöglicht, die durch seine Prozesse bereitgestellt werden. Diese Form der Adaption wird beispielsweise verwendet, um zwei Standardsoftwaresysteme zu verbinden oder ein externes System an eine Standardsoftware anzubinden.

Zugriffe auf Ebene des Datenbankservers sind über Schnittstellen oder durch vom Datenbankhersteller ausgelieferte Programme möglich. Beide Techniken bergen jedoch Risiken, da Standardsoftwaresysteme teilweise Funktionalitäten für Zwischenspeicherung (*Buffering*) und Sperrmechanismen aus dem Datenbanksystem auslagern. Beispielsweise verfügt das R/3 System über eigene Sperrmechanismen, die durch Adaption auf der Datenbankebene umgangen würden. Das R/3 System nutzt des weiteren sogenannte *Pool-* und *Cluster-Tabellen*, bei denen mehrere Tabellen der Standardsoftware auf eine einzige Datenbanktabelle abgebildet werden[15].

Da die Benutzerverwaltung der Standardsoftware von der Datenbank losgelöst ist (siehe Kapitel 4.4), kann die Sicherheit nur durch eine detaillierte Benutzerverwaltung auf Datenbankebene, die standardmäßig nicht von allen Standardsoftwareherstellern unterstützt wird, gewährleistet werden.

[15] Der Grund für *Pool-* und *Cluster-Tabellen* ist die Restriktion von Datenbanksystemen nur eine bestimmte Anzahl von Tabellen zu verwalten. In neueren Datenbanksystemen ist diese Restriktion weitestgehend aufgehoben. Die Anzahl der *Pool-* und *Cluster-Tabellen* reduziert sich im R/3 System somit sukzessive. Jedoch sind in *Release 3.1H* ca. 136 und in *Release 4.0B* ca. 40 *Pool-* und *Cluster-Tabellen* vorhanden.

Eine Trennung der Daten von einzelnen Mandanten wird zusätzlich erschwert, da die einzelnen Datenbanktabellen, je nach Standardsoftware, Daten mehrerer Mandanten enthalten, die zwar auf Ebene der Standardsoftware verborgen, auf Datenbankebene jedoch nur durch einen zusätzlichen Primärschlüssel gekennzeichnet sind.

Diese Problematiken erschweren die Adaption auf Ebene des Datenbanksystems, so daß diese Techniken im weiteren Verlauf dieser Arbeit nicht diskutiert werden. Somit bleiben die Möglichkeiten der Adaption auf Ebene des Präsentationsservers und des Applikationsservers, die im Folgenden nach ihrer Art charakterisiert und kategorisiert werden.

3.3 Charakterisierung und Kategorisierung der Adaptionsmechanismen

Werden die Adaptionsmechanismen anhand der Zugriffsebene in der Softwarearchitektur unterschieden, so ergeben sich drei Kategorien (vgl. Kapitel 3.2). Die einzelnen Adaptionsmechanismen können bei dieser Kategorisierung vergleichbare Ziele verfolgen, werden jedoch in unterschiedliche Kategorien eingeordnet. Die Anbindung eines Fremdsystems wäre sowohl auf Ebene der Präsentation, als auch auf der Applikationsebene möglich.

Funktionalitäten und Techniken, die mittels Adaption gesteuert beziehungsweise realisiert werden sollen, decken eine große Bandbreite ab:

- Ausgaben als Text (Bsp.: Rechnungen, Rundschreiben)
- Massen In- und Output (Bsp.: Altdatenübernahme, Massendatenverarbeitung)
- Datenaustausch und Nachrichtenaustausch bei Kopplung von Systemen
- Steuerung externer Systeme (Bsp.: Steuerung eines EDI[16]-Subsystems)
- Steuerung der Standardsoftware durch ein externes System (Bsp.: Online-Bestellsystem übergibt Daten an die Standardsoftware)
- E-Commerce (Bsp.: EDI, Online-Bestellsystem, Electronic Banking)
- Eigenentwicklungen (Bsp.: Online-Bestellsystem, Lagerverwaltung)
- Ändern von betriebswirtschaftlichen Strukturen (Bsp.: Anlegen von Buchungskreisen, Finanzstellen)
- Benutzerverwaltung (Bsp.: Anlegen verschiedener Benutzer und Rollen)
- Automatisieren von Arbeitsabläufen und Geschäftsprozessen (Bsp.: Workflow-Systeme)
- etc.

[16] EDI - Electronic Data Interchange

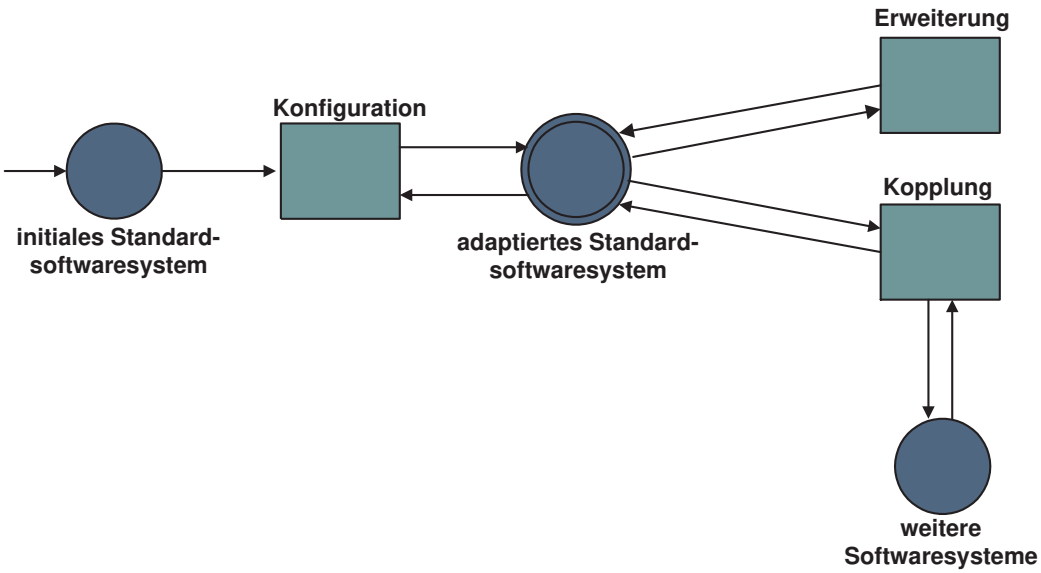

Abbildung 3.3: Zusammenhang der Kategorien zur Adaption von Standardsoftware

Diese verschiedenen Funktionalitäten und Techniken werden in drei Kategorien der Adaption zusammengefaßt:

Konfiguration bezeichnet Funktionalitäten und Techniken, die bei der Einführung einer Standardsoftware obligatorisch sind. Es handelt sich um Variationen, die von der Standardsoftware vorgesehen sind. Die Konfiguration von Standardsoftware ist obligatorisch, da

- Standardsoftwaresysteme im Rohzustand (direkt nach der Installation) nicht einsatzfähig sind,
- das Abbilden von Geschäftsprozessen ein integraler Bestandteil bei der Einführung von Standardsoftwaresystemen ist und
- Rollen und Benutzer im System abgebildet werden müssen.

Erweiterung bezeichnet Funktionalitäten und Techniken, die der Erweiterung einer Standardsoftware dienen. Erweiterungen sind für den produktiven Einsatz einer betrieblichen Standardsoftware nicht notwendig, sondern stellen eine Option dar. Das Erweitern von Standardsoftwaresystemen ermöglicht

- das Automatisieren von Geschäftsabläufen durch Workflow-Systeme, um Arbeitsabläufe zu beschleunigen bzw. zu automatisieren und Kosten zu senken,
- spezifische Anforderungen, die nicht im Standard vorgesehen sind, in die Standardsoftware abzubilden und
- eine Anbindung an das Internet, um den Zugriff auf Daten des Standardsoftwaresystems zu vereinfachen und global zu ermöglichen.

Kopplung bezeichnet Funktionalitäten und Techniken, um externe Systeme an eine Standardsoftware anzubinden oder verschiedene Standardsoftwaresysteme des gleichen Herstel-

lers zu integrieren. Die Standardsoftware stellt Schnittstellen zur Realisierung der Anbindung bzw. Kopplung zur Verfügung. Die Kopplung von Standardsoftwaresystemen mit externen (fremden und verwandten) Systemen gewinnt zunehmend an Bedeutung, da

- die Globalisierung der Märkte in großen Unternehmen zur physikalischen Trennung der Organisationseinheiten führt,
- Kunden und Lieferanten in die Geschäftsprozesse involviert werden und
- die Leistungsfähigkeit von Standardsoftware durch die Verteilung von verschiedenen Anwendungen erhöht werden kann.

In Abbildung 3.3 sind die einzelnen Adaptionskategorien in einem Petri-Netz zusammengefaßt. Aus einem *initialen Standardsoftwaresystem* wird mittels *Konfiguration* ein *adaptiertes Standardsoftwaresystem* gebildet. Das *adaptierte Standardsoftwaresystem* kann nach der *Konfiguration* als produktives System eingesetzt werden oder im Folgenden durch *Erweiterung, Kopplung* oder *erneute Konfiguration* weiter adaptiert werden.

Kapitel 4

Konfiguration

Das deutsche Wort Konfiguration hat seinen Ursprung in dem lateinischen Wort *configuratio*. Übersetzt kommt man zu einer *bestimmten Art der Gestaltung*. In diesem Kapitel werden die Arten der Gestaltung von Standardsoftwaresystemen betrachtet, die durch Parametrisierung, Annotation und nur geringen Programmieraufwand möglich sind. Durch diese Arten der Gestaltung wird ein betriebliches Standardsoftwaresystem aus einem initialen Zustand in einen konfigurierten Zustand überführt.

Innerhalb dieser Arbeit sollen unter Konfiguration folgende Adaptionstechniken von betrieblichen Standardsoftwaresystemen betrachtet werden:

- Anpassen an kundenspezifische betriebliche Anforderungen (vgl. Kap. 4.1)
- Generierung und Modifikation von Formularen (vgl. Kap. 4.2)
- Nutzung von generierten Berichten (*Reports*) zu Auswertungszwecken (vgl. Kap. 4.3)
- Benutzerrechte und -verwaltung zur Abbildung von Rollen und Benutzern (vgl. Kap. 4.4)

4.1 Anpassen an kundenspezifische Anforderungen

Ein Standardsoftwaresystem im initialen Zustand enthält keine kundenspezifischen Einstellungen. Informationen über die Struktur des Unternehmen, geografische Einsatzgebiete, Sprachen, Währungen, etc. sind nicht im Standardsoftwaresystem enthalten beziehungsweise abgebildet. Je nach Standardsoftwaresystem sind im initialen Zustand zunächst alle Module (Finanzwesen, Personalverwaltung, etc.) enthalten, auch die, deren Funktionalitäten im späteren produktiven System nicht benötigt werden beziehungsweise nicht gewünscht sind.

Das Abbilden von kundenspezifischen Anforderungen in die Standardsoftware wird im Allgemeinen mit dem Begriff *Customizing* in Verbindung gebracht. Zum Abbilden der kundenspezifischen Anforderungen in die Standardsoftware stellen die Hersteller dieser Systeme verschiedene Werkzeuge als Hilfsmittel zur Verfügung. Die Änderungen, die in der Standardsoftware durchgeführt werden, beschränken sich hierbei auf Annotationen, Parametrisierung und das Ausfüllen von Feldern.

Das Vorgehen zum Anpassen an kundenspezifische Anforderungen unterscheidet sich je nach Standardsoftware. Im Folgenden wird beispielhaft das R/3 System betrachtet.

In Kapitel 2.3 wurde bereits eine Methode zur Einführung einer Standardsoftware vorgestellt, so daß an dieser Stelle auf eine detaillierte Beschreibung des Vorgehensmodells für das *Customizing* des R/3 Systems verzichtet wird.

Die oberste betriebswirtschaftliche Schicht in einem Standardsoftwaresystem ist der Mandant[1]. In einem Standardsoftwaresystem existieren verschiedenen Mandanten, bei denen es sich um produktive Mandanten, Mandaten zu Testzwecken, Mandanten für Schulungen, etc. handeln kann. Einstellungen im Softwaresystem sind teils mandantenabhängig und teils mandantenunabhängig, so daß die Trennung des produktiven Softwaresystems von Entwicklungs- und Qualitätssicherungssystemen notwendig ist.

Die Anforderungen an Entwicklungs- und Qualitätssicherungssysteme ergeben eine Zwei- oder Drei-Systemlandschaft. Bei einer Zwei-Systemlandschaft sind das Entwicklungs- und das Qualitätssicherungssystem auf einem physischen System vereint. Bei einer Drei-Systemlandschaft werden Entwicklungs- und Qualitätssicherungssystem voneinander getrennt. Der Betrieb einer Ein-Systemlandschaft ist zwar möglich, doch ist der produktive Betrieb bei Weiterentwicklungen und Versionswechseln nicht gewährleistet beziehungsweise nicht möglich.

Im initialen Zustand des Softwaresystems ist mindestens ein Mandant vorhanden, der als Ausgangspunkt (*Referenz-Mandant*) für das weitere Vorgehen dient. Des weiteren ist in Softwaresystemen bereits im initialen Zustand eine Kopie dieses *Referenz-Mandanten* zu finden. Der *Referenz-Mandant* dient als Sicherung, falls Einstellungen in anderen Mandanten nicht mehr rückgängig gemacht werden können. Änderungen durch *Versionswechsel* und *Systemupgrades* werden beispielsweise im R/3 System ebenso im *Referenz-Mandanten* vorgenommen und anschließend in andere Mandanten transportiert.

Die einzelnen Mandaten haben getrennte Benutzerstammsätze, so daß mandantenübergreifendes Arbeiten nur durch mindestens zwei Benutzerstammsätze möglich ist. Die Daten der einzelnen Mandanten sind für andere Mandanten, obwohl einige Standardsoftwaresysteme für verschiedenen Mandanten die gleichen Datenbanktabellen nutzen, nicht sichtbar.

In der Kopie des *Referenz-Mandanten* wird die Anpassung an das Unternehmen durchgeführt. Dieser Mandant wird jedoch nicht den endgültigen produktiven Mandanten ergeben, sondern es wird wiederum eine Kopie des Mandanten erzeugt, um eine Sicherung der Einstellungen zu behalten und für gegebenenfalls weitere Kopien eine Vorlage zu haben.

Für das Anpassen an die kundenspezifischen Anforderungen wird von den Standardsoftwareherstellern ein Leitfaden mitgegeben. Im R/3 System ist dies beispielsweise der *Einführungsleitfaden*. Ein Ausschnitt dieses *Einführungsleitfadens* ist in Abbildung 4.1 zu finden.

Dieser *Einführungsleitfaden* wird auch als *Referenz-Einführungsleitfaden* bzw. *Referenz-IMG*[2] bezeichnet. Der *Referenz-Einführungsleitfaden* enthält alle notwendigen Schritte für *alle* Module die zur Einführung des Standardsoftwaresystems durchzuführen sind. Aus diesem *Referenz-Einführungsleitfaden* wird bei der Einführung ein *Unternehmens-Einführungsleitfaden* generiert, der auf die gewünschten Module und Funktionalitäten eingeschränkt ist, die im

[1] Mandant: Eine für sich handelsrechtlich, organisatorisch und datentechnisch abgeschlossene Einheit.
[2] IMG - Implementation Guide

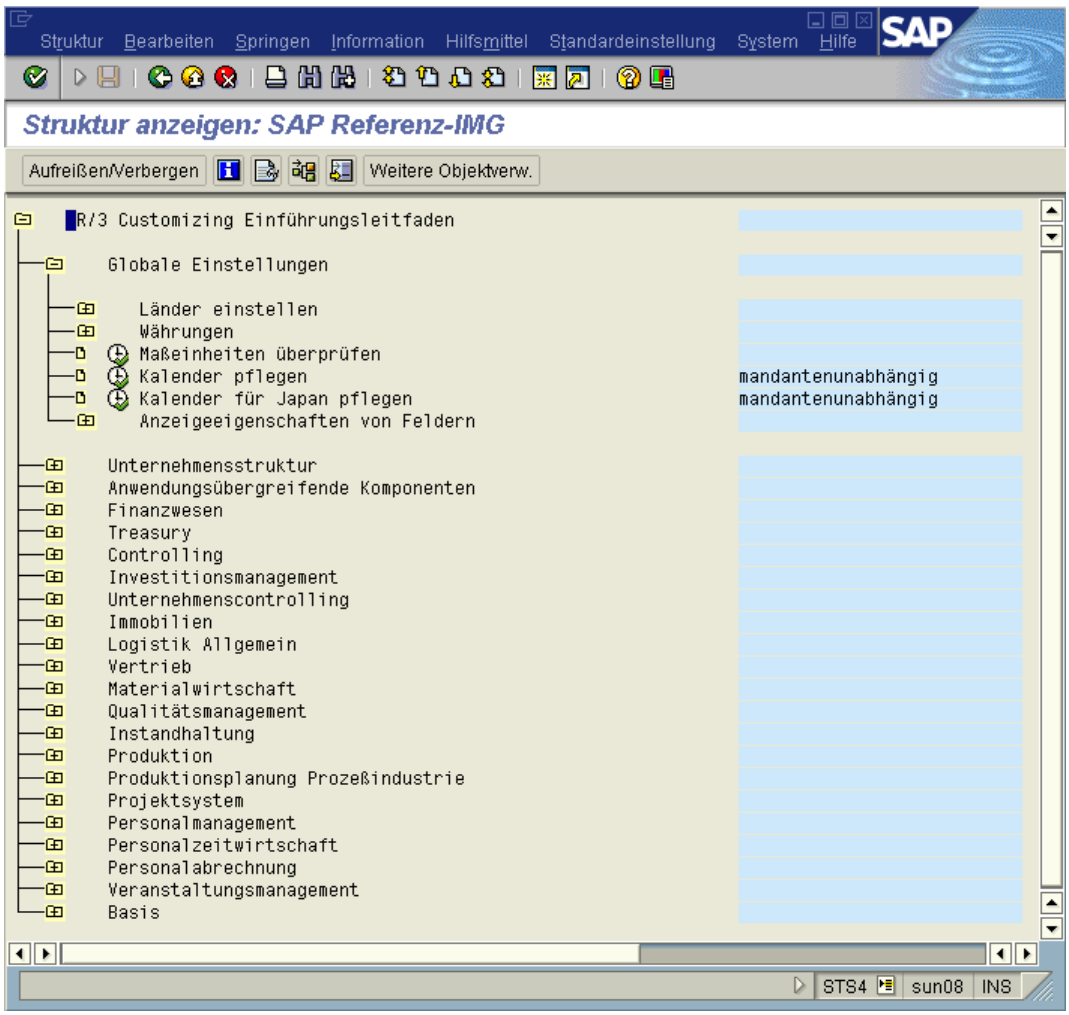

Abbildung 4.1: Der Einführungsleitfaden des R/3 Systems

späteren produktiven System genutzt werden sollen.

Ein kurzes Beispiel für eine Aktivität im *Einführungsleitfaden* ist das Anlegen eines Buchungskreises[3].

Im *Einführungsleitfaden* [SAP98c] ist das Anlegen bzw. Definieren eines *Buchungskreises* unter

> Unternehmensstruktur ⇒ Strukturpflege ⇒ Definition ⇒ Finanzwesen ⇒ Buchungskreis definieren, kopieren, löschen, prüfen

zu finden. Als Eingaben werden in diesem Schritt ein Name für den Buchungskreis, der Name der Firma, der Ort, das Land, die Währung und die Sprache gefordert. Es werden keine für

[3] Buchungskreis: Die kleinste organisatorische Einheit des externen Rechnungswesens, für die eine vollständige, in sich abgeschlossene Buchhaltung abgebildet werden kann [SAP98e].

das Finanzwesen nötigen Informationen wie Kontenrahmen, Umsatzsteuer-Id oder Geschäftsjahresvariante hinterlegt. Diese Eingaben erfolgen im Bereich *Finanzwesen* des *Einführungsleitfadens*.

Zusätzlich zu der Möglichkeit, Transaktionen im System auszuführen, bietet der *Einführungsleitfaden* die Option, zu den einzelnen Aktivitäten Notizen bzw. Dokumentationen zu hinterlegen. Dies unterstützt andere Projektmitarbeiter und hilft zu einem späteren Zeitpunkt, die Einstellungen nachzuvollziehen.

Bei der Einführung einer Standardsoftware wie R/3 wird der *Unternehmens-Einführungsleitfaden* sukzessive abgearbeitet und die entsprechenden Einstellungen vorgenommen. Erfolgen bei einer Transaktion *mandantenunabhängige* Einstellungen, so wird der Benutzer vom System gewarnt, daß sich alle folgenden Einstellungen auf *alle* Mandanten des Systems auswirken und *nicht* nur auf den aktuell bearbeiteten Mandanten.

Neben den Einstellungen für betriebswirtschaftliche Anforderungen wird im *Einführungsleitfaden* beispielsweise auch in Bereiche für die Benutzerverwaltung (siehe Kapitel 4.4) und die Konfiguration für Verteilung-ALE[4] (siehe Kapitel 6.3) verzweigt. Für eine detaillierte Beschreibung des *Einführungsleitfadens* wird auf die zugehörige Dokumentation [SAP98c] verwiesen.

4.2 Formularwesen - Generierung und Modifikation

Um in einer betriebswirtschaftlichen Standardsoftware Ausgaben als Textdokumente zu erhalten, wird eine Anwendung benötigt, die Daten zur Ausgabe selektiert und formatiert. Hierzu ist in das R/3 System das SAPscript integriert worden. Sollen zum Beispiel Rechnungs- oder Bestellformulare erstellt werden, so geschieht dies mit dem integrierten SAPscript. Die SAP AG stellt bereits vorgefertigte Formulare zur Verfügung, die jedoch an die einzelnen Gegebenheiten des Unternehmens angepaßt werden müssen.

SAPscript ist das integrierte Text-Managementsystem des R/3. SAPscript beinhaltet zum einen Methoden, die aus Textverarbeitungen wie Microsoft Word bekannt sind (Unterstreichen, Ändern der Schriftart, etc.) und zum anderen eine Programmierschnittstelle, mit der Daten zwischen dem zu druckenden Dokument und der Datenbank transferiert werden.

Die Programmiersprache des SAPscript ist verwandt mit den Sprachen STML [GW97][5] oder JSSI[Pro99] [6].

Die wesentlichen Komponenten des SAPscript sind:

- Ein Editor zum Anlegen und Bearbeiten von Zeilen eines Textes. Der Editor für Standardtexte wird über die Transaktion *SO10* aufgerufen. Für Windows-Clients steht neben dem klassischen Zeileneditor (vergleichbar mit dem ABAP Editor) ein WYSIWYG-Editor zur Verfügung.

[4] ALE - Application Link Enabling

[5] STML (Structured Tycoon Markup Language) dient der Erstellung von dynamischen HTML-Seiten mit Tycoon-2[GW98]

[6] JSSI gehört zum Java Apache Project und ermöglicht die Erstellung von HTML-Seiten mit Integration von JavaServlets.

- Stile und Formulare, die im SAPscript für die Druckaufbereitung verwendet werden. Die einzelnen Stile und Formulare sind nicht fest an einen Text gebunden, sondern werden unabhängig erstellt und verwaltet (Transaktion *SE72*).

- Der *Composer* als das zentrale Organ für die Ausgabeaufbereitung. Je nach Ausgabegerät bereitet der *Composer* die Formatierungsinformationen für den Text auf, wobei er die hinterlegten Stil- und Formularinformationen berücksichtigt.

- Die Programmierschnittstelle ermöglicht die programmgesteuerte Ausgabe von Formularen. Hierbei wird ein Report oder auch Modulpool mit einem SAPscript Formular verknüpft.

- Mehrere Datenbanktabellen, in denen zu SAPscript gehörige Texte, Stile, Formulare, etc. abgelegt werden.

Um die einzelnen Komponenten von SAPscript vorzustellen, wird im Folgenden ein SAPscript erstellt, welches eine Liste aller Debitoren des Systems ausgibt.

Der Stil definiert zunächst die Absatz- und Zeichenformate des Dokumentes. Unterschiedliche Darstellung für Überschriften, Textkörper, etc. werden also durch Stile definiert. Folgende Elemente werden in einem Stil definiert:

- In den Kopfdaten wird der Default-Absatz des Stils und eine Bedeutung hinterlegt. Zusätzlich generiert das System im Kopf Daten für den Sprachenschlüssel, Mandantennummer, Erstellungsdatum, etc.

- Die Absatzformate legen die Aufbereitung der Absätze im SAPscript fest. Ein Absatzformat wird durch die Attributgruppen Standard, Font, Tabulatoren und Gliederung definiert.

- Durch Zeichenformate wird die Gestaltung des Textes innerhalb eines Absatzes definiert. Hierbei stehen die Attributgruppen Standard und Font (siehe oben) zur Verfügung. Es ist auch möglich, ein Zeichenformat für Barcodes zu hinterlegen, so daß automatisch der richtige Barcode für die entsprechende Zeichenkette im SAPscript erscheint.

In dem von der SAP AG ausgelieferten R/3 System sind bereits Stile vorhanden. Diese sind in den Mandanten 000 und 001 zu finden. Will man einen dieser mitgelieferten Stile als Ausgangspunkt benutzen, so muß dieser zunächst in den eigenen Mandanten kopiert werden, da sonst mandantenübergreifende Änderungen vorgenommen werden.

Im nächsten Schritt wird das Formular erstellt. Ein Formular beschreibt zum einen das Seitenlayout der einzelnen Druckseiten und zum anderen werden sogenannte Textelemente in einem Formular definiert.
Ein Formular besteht aus den folgenden Punkten:

- Die Kopfdaten eines Formulars entsprechen im wesentlichen den Kopfdaten eines Stils. Auch hier werden Entwicklungsdaten, Status, Beschreibung, Seitenorientierung und die zu Beginn gültige Schriftart definiert.

- Absatz- und Zeichenformatdefinitionen sind wiederum verwandt mit den entsprechenden Attributen eines Stils. Die einzelnen Absatz- und Zeichenformatdefinitionen werden in den Seitenfenstern über den zugehörigen Namen verwendet.

- Seiten repräsentieren die späteren gedruckten Seiten. Eine Rechnung enthält zum Beispiel im Normalfall auf der ersten Seite einen Adressaten. Auf den Folgeseiten werden die einzelnen Rechnungspositionen ausgegeben und zum Schluß die AGBs. Hierzu wären somit drei Seiten notwendig. Die erste Seite enthält den Adressaten und gegebenenfalls schon einige Rechnungspositionen. Die zweite und die folgenden Seiten listen die Rechnungspositionen. Die letzte Seite enthält die AGBs.

- Seitenfenster sind die Kombination aus einer Formularseite und einem Fenster. Ein Seitenfenster definiert welches Fenster an welcher Position auf welcher Seite ausgegeben werden soll.

- Fenster sind einzelne Ausgabebereiche, die in die Seitenfenster eingebunden werden.

- Textelemente werden den Fenstern zugeordnet. Diese Textelemente werden in dem zugehörigen Druckprogramm, dies kann ein Report oder Modulpool sein, über den Namen (Call-By-Name) mit Daten gefüllt.

Ordnet man einem erstellten Fenster ein Druckprogramm zu, so werden die im Druckprogramm deklarierten Datenstrukturen verwendet (siehe Abbildung 4.2). Wird das Druckprogramm aufgerufen, so füllt es die in den Fenster referenzierten Datenstrukturen mit Werten.

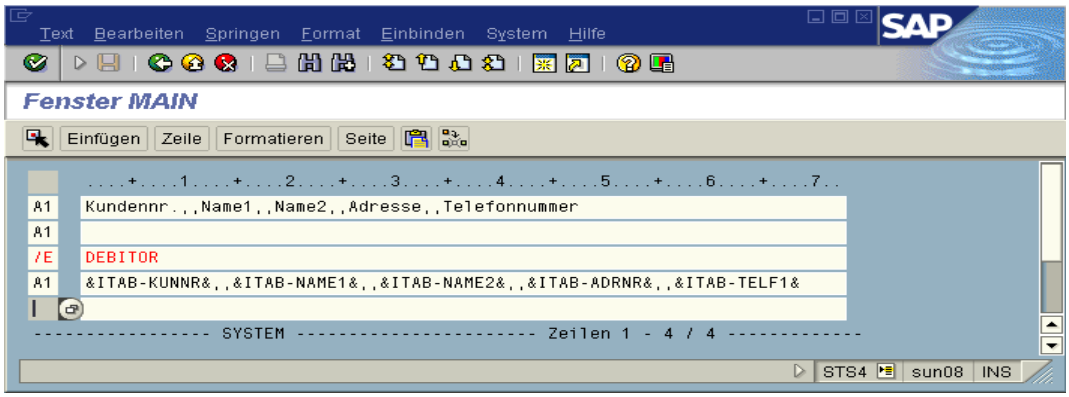

Abbildung 4.2: Seitenfenster eines SAPscript Formulars

Stile, Formate und Texte, die mit SAPscript erstellt werden, liegen in dem von der SAP AG definiertem ITF[7] vor. Das ITF besteht aus einem Formatfeld und einem Textfeld. Im Formatfeld werden sowohl Absatzformate als auch Steuerkommandos übergeben. In Abbildung 4.2 definiert */E* ein Textelement mit dem Namen *Debitor*. Die Textfelder eines SAPscript Textes liegen unter anderem auch in den Formaten RTF oder Winword-DOC vor, wobei das zugehörige Formatfeld gesetzt wird.

[7] ITF - Interchange Text Format

Zur Vereinfachung wird in diesem Beispiel nur die Erstellung eines Reports als zugehöriges Druckprogramm beschrieben, wobei auch ein Modulpool mit mehreren Screens bzw. DynPros möglich wäre (siehe Abbildung 4.2).

Die Programmierschnittstelle benutzt sogenannte SAPscript-Funktionsbausteine, die von der SAP AG definiert wurden. In Abbildung 4.2 öffnet der Funktionsbaustein *OPEN_FORM* durch den Parameter *FORM* ein Formular mit dem Namen *YAW_FORMU1*. In dem Druckprogramm bilden diese Funktionsbausteine den Kommunikationskanal zum Formular. Da es sich bei einem Druckprogramm um ein ABAP/4 Programm handelt, kann man die volle Funktionalität von ABAP/4 nutzen. Somit ist es möglich auf Tabellen zuzugreifen und Anfragen zu formulieren, soweit die Berechtigungsstruktur dies zuläßt.

Bei der Erstellung von SAPscripts werden verschiedene Rollen benötigt, die mit unterschiedlichen Berechtigungen im R/3 System verbunden sind. Zum einen der ABAP Entwickler, der das Druckprogramm schreibt und zum anderen der SAPscript Formular Programmierer, der das Druckprogramm als gegeben nimmt und auf die Datenstrukturen des Druckprogramms referenziert, um die Ausgabe über die Formulare zu verteilen beziehungsweise zu „visualisieren“.

Da die SAP AG selbst kein System zur Portooptimierung im R/3 System integriert hat, wurde eine Schnittstelle im SAPscript eingebaut. Dieses RDI[8] ermöglicht den Anschluß externer Textmanagementsysteme. Über diese Schnittstelle werden nur Formulardaten übertragen und keine Layoutinformationen wie Schrifttyp oder Seitengröße. Die Aufbereitung muß also außerhalb des R/3 Systems erfolgen.

Ein weiterer Nachteil der Verwendung von RDI zum Drucken besteht in der Auftragsverwaltung. Es wird zwar eine Fehlermeldung im R/3 System angezeigt, wenn die Daten nicht korrekt übertragen worden sind, doch Fehler im externen Druckprogramm, die zum Beispiel bei der Aufbereitung der Daten entstehen, werden nicht im R/3 System bekannt. Hierzu müßten dann Workflow-Systeme (siehe Kapitel 5.1) oder Mailsysteme die Fehlermeldung an R/3 zurückgeben.

Die Ausgabe des *Output-Stream* kann entweder als Spool oder IDoc (siehe Abbildung 4.4) erfolgen. Im Ausgabemodus Spool besteht der Datenstrom aus Header-, Daten-, Sortier- und Kontrollsätzen. Jeder Datensatz kennzeichnet durch sein erstes Byte, um welche Art von Daten es sich handelt.

SAPscript eignet sich nur für die Datenselektion, die Formatierung von Dokumenten und das Exportieren selektierter Daten als Spooldatei oder IDoc[9]. Änderungen von Daten oder eine bidirektionale Kommunikation mit externen Systemen ist mit SAPscript nicht zu erreichen.

4.3 Berichtswesen - Reports

Das Ziel des in Kapitel 4.2 beschriebenen Formularwesens ist es, Dokumente zu erzeugen, die auf elektronischem oder manuellem (per Post, Fax, etc.) Wege weiterverarbeitet werden. Es handelt sich bei den erzeugten Formularen beispielsweise um Rechnungen, Mahnungen, Briefe

[8] RDI - Raw Data Interface
[9] IDoc - Intermediate Document (siehe Kapitel 6.3.1)

```
REPORT ZAW_REP_FORM1 .
TABLES: KNA1.

DATA ITAB LIKE KNA1 OCCURS 10 WITH HEADER LINE.
SELECT * FROM KNA1 INTO TABLE ITAB.

CALL FUNCTION 'OPEN_FORM'
     EXPORTING
          FORM         = 'YAW_FORMU1'
          LANGUAGE     = SY-LANGU
     EXCEPTIONS
          CANCELED     = 1
          DEVICE       = 2
          FORM         = 3
          OPTIONS      = 4
          UNCLOSED     = 5
          MAIL_OPTIONS = 6
          OTHERS       = 7.

LOOP AT ITAB.
  CALL FUNCTION 'WRITE_FORM'
       EXPORTING
            ELEMENT                  = 'DEBITOR'
            FUNCTION                 = 'SET'
            TYPE                     = 'BODY'
            WINDOW                   = 'MAIN'
       EXCEPTIONS
            ELEMENT                  = 1
            FUNCTION                 = 2
            TYPE                     = 3
            UNOPENED                 = 4
            UNSTARTED                = 5
            WINDOW                   = 6
            BAD_PAGEFORMAT_FOR_PRINT = 7
            OTHERS                   = 8.
ENDLOOP.

CALL FUNCTION 'CLOSE_FORM'
     EXCEPTIONS
          UNOPENED                 = 1
          BAD_PAGEFORMAT_FOR_PRINT = 2
          OTHERS                   = 3.
```

Abbildung 4.3: Quellcode des Reports - Druckprogramm zu SAPscript

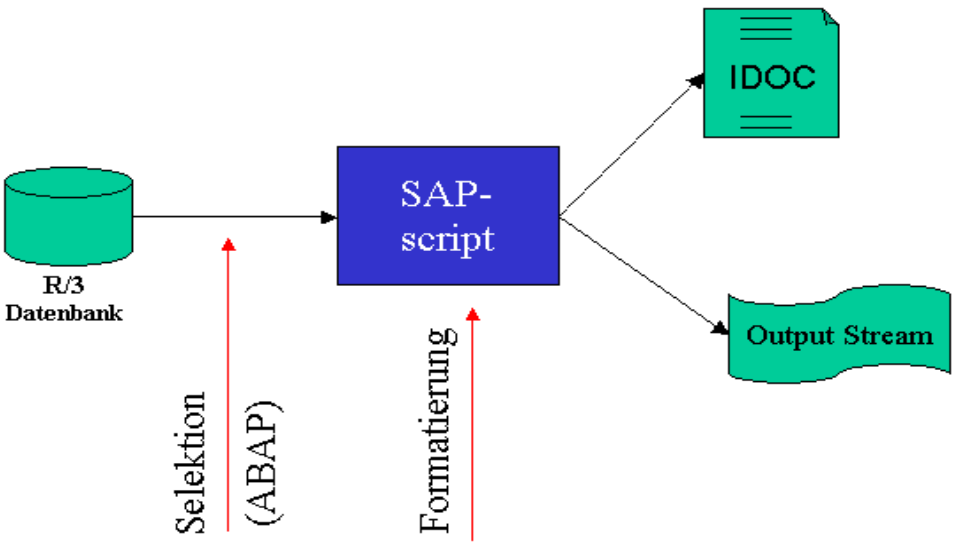

Abbildung 4.4: Ausgabe von SAPscript

und Faxe. Die Ausgabe (*Output*) des Berichtswesen ist hingegen eine Auswertung, die aus den Daten des Systems generiert wird. Bei diesen Ausgaben kann es sich beispielsweise um eine GuV[10], Lagerbestandslisten, Personaleinsatzplanungen oder um eine einfache Geburtstagsliste handeln. Im Gegensatz zu Formularen sind in Berichten mathematische Operationen, wie Summenbildung oder prozentuale Verteilung, von größerem Belang. Im Formularwesen ist das Einbinden von Summen, etc. nur durch Programmierung des zugehörigen Druckprogramms möglich. Das Berichtswesen berücksichtigt diese Anforderungen hingegen und stellt mit den entsprechenden Werkzeugen die notwendigen Funktionalitäten bereit.

Das Berichtswesen von Standardsoftwaresystemen deckt durch vordefinierte Berichte bereits einen Teil der Anforderungen ab. Auswertungen des Finanzwesens wie GuV, Journale, Summen- und Saldenlisten, offene Postenlisten oder Preiskalkulationslisten und Bestandslisten der Materialwirtschaft sind in den Standardsoftwaresystemen bereits enthalten. Besteht in einem Unternehmen der Bedarf, weitergehende oder spezielle Auswertungen vorzunehmen, so müssen diese durch Werkzeuge der Standardsoftware eigenhändig erstellt werden.

Das R/3 System stellt beispielsweise zur Generierung von Berichten drei Werkzeuge zur Verfügung:

Der Report-Writer erstellt Berichte aus den Daten verschiedener Anwendungskomponenten des Standardsoftwaresystems. Der *Report-Writer* selektiert die nötigen Daten und bereitet sie in der gewünschten Form auf. Das Errechnen von Kennzahlen, Prozentsätzen, Abweichungen, Summen etc. wird vom *Report-Writer* unterstützt. Eine grafische Aufbereitung der Berichte ist über die im R/3 System integrierte Grafikschnittstelle möglich. Die Weitergabe des Berichts an eine Tabellenkalkulation wird ermöglicht, um den Bericht gegebenenfalls weiter zu verarbeiten.

[10] GuV - Gewinn- und Verlustrechnung

Der Bericht unterteilt sich im *Report-Writer* in einen Kopf, mehrere Zeilen, mehrere Spalten, eine allgemeine Selektion und Texte. Ein WYSIWYG-Editor für das Anlegen von Berichten mit dem *Report-Writer* ist nicht vorhanden.

Der Report-Painter ist eine eingeschränkte Variante des *Report-Writers*. Die mögliche Komplexität der zu generierenden Berichte des *Report-Painter* ist geringer als die des *Report-Writers*. Das Anlegen von Berichten erfolgt über einen WYSIWYG-Editor. Das Ausgabeformat des *Report-Painters* und des *Report-Writers* ist identisch.

Die ABAP/4-Queries dienen dem Erstellen von Listen. Wird eine *ABAP/4-Query* angelegt, so generiert das R/3 System im Hintergrund automatisch einen *ABAP Report*. Das Anlegen von *ABAP/4-Queries* geschieht menügesteuert, wobei keine Kenntnisse in *ABAP-Listenprogrammierung* notwendig sind. *ABAP/4-Queries* unterteilen sich in *Grundlisten*, *Statistiken* und *Ranglisten*, die miteinander kombinierbar sind. *Grundlisten* sortieren Daten nach verschiedenen Kriterien und Ermöglichen eine Summenbildung. *Statistiken* bieten die Möglichkeit eine Verdichtung der Daten vorzunehmen und somit mehrere Felder mit Hilfe von Regeln zusammenzufassen. *Ranglisten* ergeben nach Werten sortierte *Statistiken*.

Als kurzes Beispiel wird die Generierung einer *ABAP/4-Query* zur Erzeugung einer Liste mit allen Debitoren, sortiert nach der Debitorennummer, schrittweise beschrieben.

Vor der Generierung einer *ABAP/4-Query* müssen zunächst *Benutzergruppen* und *Sachgebiete* eingerichtet werden. Die Benutzergruppen sind von der Benutzerverwaltung (siehe Kapitel 4.4) des R/3 System losgelöst. Mehrere Benutzer, die in einer *Benutzergruppe* zusammengefaßt sind, haben die Möglichkeit, gemeinsam auf die für ihre Gruppe zugeordneten *Queries* zuzugreifen. *Sachgebiete* ermöglichen eine *Vorauswahl* der Daten. Werden aus mehreren Tabellen Daten benötigt, so wird der Benutzer möglicherweise aus den vielen Spalten nicht die für ihn relevanten herausfinden. Gegebenenfalls sollen auch bestimmte Spalten für eine Benutzergruppe nicht sichtbar sein und müssen ausgeblendet werden.

Bei dem Anlegen eines Sachgebietes werden die Tabellen, die später in den *Queries* verwendet werden sollen, spezifiziert. Ein *Join* über mehrere Datenbanktabellen, wobei auch *Left-*,*Right-* und *Inner-Joins* unterstützt werden, ermöglicht den gleichzeitigen Zugriff auf mehrere Tabellen. Nach dem Anlegen des *Sachgebietes* werden eine oder mehrere *Sachgruppen* angelegt, die mehrere Spalten der Tabellen in logische Blöcke zusammenfassen. Soll eine Liste aller Debitoren erstellt werden, so wird im *Sachgebiet* die Tabelle *KNA1* als Quelle angegeben und die gewünschten Felder (z.B. Debitorennummer, Name1, Adresse) werden in einem *Sachgebiet* hinterlegt.

Beim Anlegen der *Query* wird das zuvor angelegte *Sachgebiet* als Herkunft für die Daten angegeben. Nach Eingabe eines Titels, Anzahl der Zeilen, maximaler Spaltenbreite, etc. werden die *Sachgruppen* und anschließend die Felder für die Liste selektiert. Bei der anschließenden Pflege der *Grundliste* werden die einzelnen Felder auf der Liste plaziert und Sortierkriterien hinterlegt. Die Pflege für *Statistiken* und *Ranglisten* erfolgt, wenn benötigt, im Anschluß.

4.4 Benutzerrechte und Benutzerverwaltung

Standardsoftwaresysteme müssen mehrere Tausend Benutzer verwalten und gewährleisten, daß kein Benutzer Zugriff auf Daten erhält, für die seine Zugangskennung nicht autorisiert ist. Da ein Standardsoftwaresystem des weiteren innerhalb eines Unternehmens eine wesentliche Rolle einnimmt und die Daten je nach eingesetzten Komponenten (Finanzwesen, Controlling, Personalverwaltung, etc.), zu den sensitivsten Daten eines Unternehmens zählen können, ist eine detailliertes Sicherheitskonzept nötig.

Beispielhaft werden am R/3 System die Konzepte zur Benutzerverwaltung und zur Vergabe von Benutzerrechten aufgezeigt. Im Umfeld des R/3 Systems müssen Benutzer auf drei Ebenen unterschieden werden:

- Benutzer auf Betriebssystemebene
- Benutzer auf Datenbankebene
- Benutzer im R/3 System

Auf Betriebssystemebene existieren die Benutzer `ora<sid>`[11] und `<sid>adm`[12] für ein R/3 System. In R/3 Release 4.0 ist der Benutzer `ora<sid>` Mitglied der Gruppen `oper` und `dba`, der Benutzer `<sid>adm` hingegen nur Mitglied von `oper`. Mitglieder der Gruppe `oper` können die Datenbank stoppen/starten, Sicherungen durchführen und Datenbankprüfungen anstoßen, verfügen jedoch über keine Lese- oder Schreibberechtigungen. Die Mitglieder von `dba` haben zusätzlich Lese- und Schreibberechtigungen für die Datenbank[13].

Auf der Datenbankebene ist für das R/3 System der Benutzer `sapr3` der wichtigste. Alle Anfragen des R/3 Systems an die Datenbank werden als Benutzer `sapr3` gestellt und die Transaktionen werden mit diesem Benutzer durchgeführt. Das Initialpaßwort „sap“ sollte nach der Installation geändert werden, da zum Beispiel mittels *ODBC* [Gei95] oder *SQLPlus* auf die Datenbank zugegriffen werden könnte. Des weiteren müssen die Paßwörter der beiden Datenbank-Benutzer[14] `Internal` und `System` geändert werden. Der Benutzer `Internal` ist höher privilegiert und kann geschlossene Datenbanken starten und in verschiedenen Zuständen (*mount-points*) herauf- oder herunterfahren. Der Benutzer `System` hat erst Zugriff auf die Datenbank, wenn diese geöffnet und gestartet ist. Die Initialpaßwörter der Benutzer sind abhängig von der genutzten Datenbank. Im Folgenden werden die Benutzer und Berechtigungskonzepte innerhalb des R/3 Systems beschrieben.

[11] `<sid>` bezeichnet den Systemidentifikator, der drei Zeichen lang ist und mit einem Buchstaben beginnen muß.

[12] Beschrieben wird ein unterliegendes UNIX System. Bei Microsoft Windows NT hat der Benutzer `ora<sid>` die Kennung `<sid>adm` und `<sid>adm` die Kennung `sapservice<sid>`.

[13] Innerhalb der Datenbank nimmt der Benutzer `ora<sid>` die Rolle `sys` ein, wobei `sys` einer der Datenbankadministratoren ist.

[14] Je nach Datenbankhersteller kann es sich auch um drei Benutzer handeln, die verschiedene Namen haben können.

4.4.1 Standardbenutzer

Benutzer im R/3 System sind generell mandantenabhängig. Dies bedeutet, daß ein Benutzer nur Zugriff auf den Mandanten hat, in dem er angelegt wurde. Nach der Installation eines R/3 Systems sind zunächst die Standardbenutzer `SAP*` und `DDIC` in den Mandanten 000 und 001 vorhanden.[15] Zusätzlich existiert in dem Mandanten 066, der als *EarlyWatch* bezeichnet wird, der Benutzer `EARLYWATCH`, mit dem Initialpaßwort „support“ und der Benutzer `SAP*`. Der Benutzer `EARLYWATCH` stellt das geringste Sicherheitsrisiko der drei Standardbenutzer dar. Sein Berechtigungsprofil erlaubt nur das Ausführen von Transaktionen zur Systemüberwachung.

Der Benutzer `SAP*` ist der Administrator oder auch *Superuser* des R/3 Systems. In den Mandanten 000 und 001 ist das Initialpaßwort „06071992“. Dieser Benutzer hat uneingeschränkten Zugriff auf das R/3 System und ist nach der Installation der einzige Benutzer mit dem Berechtigungsprofil `SAP_ALL` (vgl. Kapitel 4.4.3). Die SAP AG empfiehlt, nach der Installation die Initialpaßwörter zu ändern und die zugeteilten Benutzerprofile zu löschen. Wird der Benutzer `SAP*` gelöscht, so wird dieser automatisch mit dem Paßwort „pass“ neu angelegt und hat erneut das Berechtigungsprofil `sap_all`[16]. Beim Anlegen eines neuen Mandanten ist das Initialpasswort ebenso „pass“.

Der Verwalter des R/3 Repositories ist `DDIC`[17], mit dem Initialpasswort „19920706“. Als Verwalter des Repositories darf der Benutzer `DDIC` nicht gelöscht werden, da die Kennung für bestimmte Installations- und Konfigurationsaufgaben, wie zum Beispiel SPAM-Updates[18] und Transporte, genutzt werden muß.

4.4.2 Berechtigungsobjekte und Berechtigungen

Die „kleinsten Berechtigungseinheiten“ in einem R/3 System sind die *Berechtigungsfelder*. In einem Berechtigungsfeld werden Aktivitäten (z.B. Anzeigen, Ändern, Buchen) zugeordnet. Auf der nächsten Ebene schließen sich die *Berechtigungsobjekte* an, denen bis zu 10 Berechtigungsfelder zugeordnet werden können. Das Anlegen von eigenen Berechtigungsfeldern und Berechtigungsobjekten ist möglich. Um in eigenen ABAP Programmen eine Berechtigungsprüfung durchzuführen, muß dieses mit dem Befehl `CHECK-AUTHORITY` angestoßen werden, wobei das zu prüfende Berechtigungsobjekt und die Werte der Berechtigungsfelder als Parameter angegeben werden. Das Hinterlegen von Berechtigungsprüfungen für den Aufruf von Transaktionen erfolgt ebenso durch das Zuordnen eines Berechtigungsobjektes und Werten zu den Berechtigungsfeldern. Seit R/3-Release 3.0E wird bei jedem Transaktionsstart eine Berechtigungsprüfung mit dem Berechtigungsobjekt *S_TCODE* durchgeführt.

Eine Berechtigung, die den Benutzer ermächtigt, eine bestimme Aktion im R/3 System auszuführen, ist das Zuordnen von Werten zu den Berechtigungsfeldern eines Berechtigungsobjektes.

[15] Nach der Installation existieren zunächst nur die Mandanten 000, 001 und 066.

[16] Der Benutzer SAP* ist in den Programmen der SAP AG „hart verdrahtet“. Dieser Sonderfall kann nur durch das Verändern eines Profilparameters umgangen werden.

[17] DDIC ist die Abkürzung für Data Dictionary

[18] SPAM - SAP Package Manager

4.4.3 Profile und Sammelprofile

Die Komplexität des R/3 Systems ergibt eine große Anzahl von Berechtigungsobjekten. Würde man jedem einzelnen Benutzer die Berechtigungen einzeln zuordnen, so ist der Aufwand unakzeptabel hoch und die resultierende Wartbarkeit niedrig. Da in den meisten Unternehmen Benutzer mit ähnlichen beziehungsweise gleichen Tätigkeiten existieren, können mehrere Berechtigungen zu Profilen zusammengefaßt werden. Profile wiederum können zu Sammelprofilen zusammengefaßt werden, die dem Benutzer zugeordnet werden können.

Werden Änderungen in einem Profil gemacht, so wirken sich diese auf alle Benutzer aus, denen dieses Profil zugeordnet ist, dies allerdings erst bei der nächsten Anmeldung. Erweitert sich das Berechtigungskonzept mit neuen oder geänderten Berechtigungsobjekten durch das Einspielen eines *Updates* so müssen gegebenenfalls Berechtigungen nachgepflegt werden. Neuerungen, die auf diese Art und Weise entstehen können, werden von der SAP AG in dem Berechtigungsprofil „SAP_NEW“ abgelegt.

Eine weitere Möglichkeiten, um Berechtigungen zu pflegen, bietet der Profil-Generator. Im Vordergrund stehen hierbei die sogenannten Aktivitätsgruppen, die innerhalb des Unternehmens Arbeitsfelder widerspiegeln sollen. Bei der Nutzung des Profilgenerators wird zunächst aus dem *Unternehmens-IMG* (siehe Kapitel 4.1) das *Unternehmensmenü* generiert. Das Unternehmensmenü ist eine Untermenge des SAP-Standardmenüs und enthält nur die Bereichsmenüs, die für die im *Unternehmens-IMG* festgelegten Komponenten benötigt werden. Anhand der Baumstruktur des *Unternehmensmenüs* ist es mit dem Profilgenerator möglich, in dem Menübaum zu navigieren und einzelne Transaktionen freizugeben oder zu sperren. Das Hinterlegen von speziellen Attributen, zum Beispiel Berechtigung für Buchungskreis *HH01*, aber keine Berechtigung für Buchungskreis *HH02*, ist ebenso möglich.

Eine Besonderheit im R/3 Berechtigungskonzept gegenüber Berechtigungskonzepten von Betriebssystemen ist die zeitgebundene Vergabe von Berechtigungen. Einzelnen Aktivitätsgruppen können somit Zeiträume zugeordnet werden, in denen die Berechtigung aktiviert ist. Wie im R/3 System üblich, sind die Benutzer- und Berechtigungsdaten ebenso in Datenbanktabellen abgelegt. Eine Historie über die letzten Änderungen an Berechtigungen, Berechtigungsprofilen, etc. wird ebenso in einzelnen Tabellen mitgeführt.

Kapitel 5

Erweiterung

In Kapitel 4 wurden Adaptionstechniken beschrieben, die bei der Einführung eines Standardsoftwaresystem obligatorisch sind. Neben diesen Adaptionstechniken stellen die Hersteller von Standardsoftwaresystemen optionale Module und Werkzeuge zur Verfügung, um weitere Funktionalitäten in der Standardsoftware abzubilden, deren Nutzung für ein produktives System keine Voraussetzung ist. Diese Erweiterungen eines Standardsoftwaresysteme sind Gegenstand dieses Kapitels.

Zur vollständigen oder partiellen Automatisierung von Geschäftsprozessen werden in Standardsoftwaresystemen *Workflow-Systeme* integriert. In Kapitel 5.1 wird ein Workflow-System einer Standardsoftware mit seinen Komponenten und seinem Funktionsumfang vorgestellt.

Standardsoftwaresysteme bilden im Kern Standardgeschäftsprozesse ab (vgl. Kapitel 2), die nicht für ein spezielles Unternehmen entwickelt wurden, sondern für einen Markt mit einer Vielzahl von Unternehmen. Durch Konfiguration werden diese Geschäftsprozesse an ein Unternehmen weitestgehend angepaßt. Ist es nicht möglich, durch Konfiguration den Geschäftsprozeß abzuändern, so daß der Geschäftsprozeß die individuellen Anforderungen eines Unternehmens erfüllt, müssen die Standardfunktionalitäten an dieser Stelle erweitert werden. Welche Werkzeuge Standardsoftwaresysteme zur Verfügung stellen, um Änderungen durchzuführen, die im Standard zunächst nicht vorgesehen sind, wird in Kapitel 5.2 beschrieben.

Um Geschäftsprozesse durch *Validation* und *Substitution* zu ergänzen, stehen eigene Methoden und Werkzeuge zur Verfügung, die Gegenstand von Kapitel 5.3 sind. Die Erweiterung durch Personalisierung des Standardsoftwaresystems im Hinblick auf die Benutzeroberfläche des Systems folgt in Kapitel 5.4.

Die globale Vernetzung der IT-Welt durch das Internet ergibt den Bedarf, auch Standardsoftwaresysteme mit entsprechenden Erweiterungen vorzusehen. Werden Standardsoftwaresysteme an das *Internet* angebunden, können Bestellsysteme entwickelt werden, die mit den internen Warenbeständen und der Finanzbuchhaltung gekoppelt und über das Internet zugänglich sind. Das Integrieren von Kunden und Lieferanten in Geschäftsabläufe über das *Internet* erleichtert die Kommunikation zwischen den Parteien. Der Zugriff auf Daten des Standardsoftwaresystems von jedem Punkt des *Internets* erhöht die Flexibilität von Mitarbeitern und Geschäftsprozessen. Eine Möglichkeit für das Öffnen eines Standardsoftwaresystems zum *Internet* wird in Kapitel 5.5 gezeigt.

5.1 Workflow-Systeme in betrieblicher Standardsoftware

Die Optimierung von Geschäftsprozessen durch *Business Engineering* ist Gegenstand des Kapitels 2.1. Hierbei werden Geschäftsprozesse methodisch untersucht, visualisiert und optimiert. *Business Engineering* bietet keine Möglichkeit, den Ablauf eines Geschäftsprozesses durch Automatisierung zu beschleunigen. Zu diesem Zweck werden Workflow-Management-Systeme in eine Standardsoftware integriert. Der Begriff Workflow-Management-System und Workflow wird in dieser Arbeit nach [JBS97] definiert:

> Ein Workflow (WF) ist eine zum Teil automatisierte (algorithmisch) — von einem Workflow-Management-System (WFMS) gesteuert — ablaufende Gesamtheit von Aktivitäten, die sich auf Teile eines Geschäftsprozesses oder andere organisationelle Vorgänge beziehen. Ein Workflow besteht aus Abschnitten (Subworkflows), die weiter zerlegt werden können. Workflows haben einen definierten Anfang, einen organisierten Ablauf und ein definiertes Ende.

Das Ziel von Workflow-Management-Systemen in Standardsoftwaresystemen ist die möglichst weitgehende Automatisierung von Geschäftsprozessen, um einen schnellen Informationsfluß zu erhalten und somit die Zeit, die ein Geschäftsprozeß von seinem Beginn bis zu seinem Ende benötigt, zu reduzieren.

In Kapitel 5.1.1 wird die Architektur einer Workflow-Komponente vorgestellt. Anschließend werden in Kapitel 5.1.2 verschiedene Werkzeuge zur Erstellung und Modifikation von Workflows vorgestellt. Kapitel 5.1.3 beschäftigt sich mit den *Business-Objekten*, die die grundlegenden Objekte zur Definition von Workflows sind.

5.1.1 Die Architektur der Workflow-Komponente

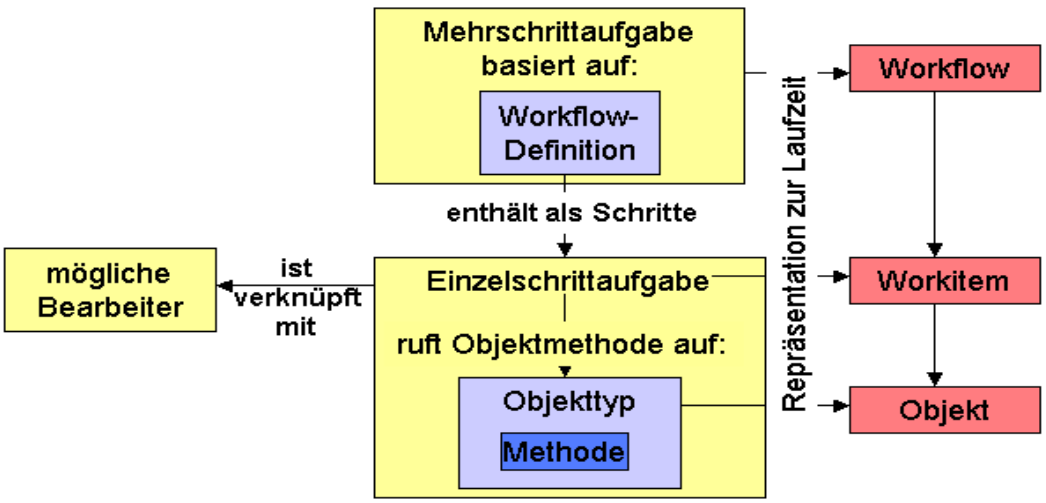

Abbildung 5.1: Architektur des SAP Business Workflows [BMS99]

In das R/3 System ist als Workflow-Management-System der *SAP Business Workflow* „als anwendungsübergreifendes Werkzeug“ [SAP98e] integriert. Innerhalb des R/3 Systems wird ein Workflow wie folgt definiert:

> Ausführung einer Mehrschrittaufgabe zur Laufzeit.
> Ein Workflow besteht aus einer Abfolge von Schritten, die entweder von Personen oder automatisiert vom System bearbeitet werden.
> Die zeitliche und logische, an die Auswertung von Bedingungen geknüpfte Reihenfolge der Schritte wird vom Workflow-Manager kontrolliert und kann durch ereignisbezogene Reaktionsmechanismen flexibel gesteuert werden.
> Technisch gesehen, wird eine Workflow-Instanz durch ein „Workflow-Item“ (Workitem vom Typ F) repräsentiert.

Die Kernbegriffe dieser Definition und deren Zusammenhang mit der Architektur des *SAP Business Workflows* (siehe Abbildung 5.1) werden im Folgenden beschrieben:

Der Workflow-Manager ist die Gesamtheit der vom SAP Business Workflow zur Laufzeit bereitgestellten Programme und Funktionen zur Steuerung und zur Ablaufkontrolle eines Workflows [SAP98e].

Eine Workflow-Instanz bildet eine von anderen Workflows unabhängige Instanz. Das parallele Ausführen mehrerer Instanzen einer *Mehrschrittaufgabe* zum selben Zeitpunkt beeinflußt die einzelnen *Workflow-Instanzen* nicht.

Die Workflow-Definition (vgl. Abbildung 5.1) ist in einer *Mehrschrittaufgabe* gekapselt. Die *Workflow-Definition*, ist die Beschreibung und technische Implementierung des Ablaufs eines Workflows und besteht aus mehreren *Einzelschrittaufgaben*, die parallel oder sequentiell angeordnet sind. Die *Workflow-Definition* wird mit Hilfe des *Workflow-Editors* angelegt und gepflegt.

Eine Aufgabe im Workflow-System beschreibt einen betriebswirtschaftlichen Vorgang. *Aufgaben* werden im *SAP Business Workflow* in *Mehrschrittaufgaben* und *Einzelschrittaufgaben* unterteilt.

Die Mehrschrittaufgabe (vgl. Abbildung 5.1) ist der betriebswirtschaftliche Rahmen für die *Workflow-Definition*.
Mehrschrittaufgaben unterteilen sich im *SAP Business Workflow* in *Workflow-Muster* (Aufgabentyp WS) und *Workflow-Aufgaben* (Aufgabentyp WF). *Workflow-Muster* sind mandantenunabhängig, planvariantenunabhängig[1] (siehe [SAP98e]) und ohne Gültigkeitszeitraum. Für *Workflow-Aufgaben* gilt das entsprechende Gegenteil. Im Initialzustand des R/3 Systems sind bereits *Workflow-Muster* in das System integriert, die entweder als Vorlage dienen oder unverändert eingesetzt werden. *Workflow-Aufgaben* werden vom Kunden definiert.

[1] „Eine Planvariante ist ein bestimmter Bereich im System, in dem Sie Informationen ablegen können. Immer dann, wenn Sie Daten im PD-System (Personalplanung und -entwicklung) anlegen, müssen Sie die Planvariante angeben, in der die Informationen abgelegt werden sollen.“ [SAP98d]

Die Einzelschrittaufgabe (vgl. Abbildung 5.1) legt fest, *was* geleistet wird und *wer* zur Ausführung berechtigt ist. Die Objektmethode (vgl. Abbildung 5.1) eines *Business-Objektes* (siehe Kapitel 5.1.3) beschreibt die Einzelschrittaufgabe (*was*). *Wer* für die Ausführung berechtigt ist, ist in der Aufbauorganisation[2] hinterlegt. Einzelschrittaufgaben, die vom System durchgeführt werden, benötigen keinen Benutzer, eine Berechtigung ist an dieser Stelle nicht erforderlich.
Einzelschrittaufgaben unterteilen sich im *SAP Business Workflow* in *Standardaufgaben* (Aufgabentyp TS) und *Kundenaufgaben* (Aufgabentyp T). *Standardaufgaben* sind das Pendant zu den *Workflow-Mustern*, *Kundenaufgaben* das Pendant zu den *Workflow-Aufgaben*.

Schritte sind die elementaren Bausteine einer Workflow-Definition. Die einzelnen Schritte können in einer *Workflow-Definition* sowohl sequentiell als auch parallel angeordnet werden[3]. Im *SAP Business Workflow* sind die folgenden Schritt-Typen definiert:

- *Aktivitäten* beschreiben zur Laufzeit Aufgaben.
- *Bedingungen* und *Mehrfachbedingungen* ermöglichen das Verzweigen unter Evaluierung einer Aussage, die einen booleschen Wert zurückgibt.
- *Ereigniserzeuger* lösen ein bestimmtes Ereignis aus. Dies kann beispielsweise die Beendigung oder der Start eines weiteren Workflows sein.
- Ein *Warteschritt* wartet zur Laufzeit auf das Eintreten eines bestimmten Ereignisses.
- Mit *Container-Operationen* werden einzelne Elemente des Workflows-Containers[4] durch Rechenoperationen und Wertzuweisungen verändert.
- Eine *Benutzerentscheidung* erfordert die Eingabe des Benutzers, welche von mehreren Aktivitäten gewählt werden soll.
- *UNTIL-Schleifen* und *WHILE-Schleifen* bieten wiederholtes Durchlaufen von mehreren Schritten zur Laufzeit an.
- *Unbestimmte Schritte* können als Platzhalter ohne Funktion eingefügt werden, um die Definition übersichtlicher zu gestalten. Einem *unbestimmten Schritt* folgt ein *unbestimmtes Ereignis*.
- *Ablaufsteuerungs-Schritte* greifen auf andere Workitems des Workflows zu.

Workitems repräsentieren zur Laufzeit eine Aufgabe oder eine Aktion. *Workitems* werden den zuständigen Benutzern in den *integrierten Eingangskorb*[5] gelegt. Workitems werden entsprechend ihres Typs unterteilt. Neben dem Typen werden *Workitems* nach ihrer Sichtbarkeit im *integrierten Eingangskorb* unterschieden. Sind Elemente im Eingangskorb nicht sichtbar, so werden diese über den *Selektionsreport* [6] angezeigt. Es existieren

[2] „Die Aufbauorganisation ist eine Zusammenfassung von Informationen, die die strukturelle und personelle Umgebung des Unternehmens umfassend." [SAP98e]

[3] Für die Regeln zur Anordnung der Schritt-Typen sei auf [BMS99] verwiesen.

[4] Der Workflow-Container enthält workflow-spezifische Systemvariablen und weitere, explizit zu definierende Elemente [SAP98e].

[5] Der integrierte Eingangskorb ist mit dem Posteingang eines E-Mail-Systems zu vergleichen. Neben E-Mails, etc. werden im Eingangskorb die Workitems des Benutzers abgelegt.

[6] „Der Selektionsreport zum Auffinden von Workitems erlaubt es, Workitems beliebiger Typen nach verschiedenen Kriterien zu selektieren. Ausgehend von der Liste mit den gefundenen Workitems kann dann in die Anzeige- und Änderungssicht des Informationssystems verzweigt werden." [SAP98d]

die folgenden verschiedenen Typen von *Workitems*, wobei in Klammern die Sichtbarkeit im *integrierten Eingangskorb* angegeben ist [SAP98d]:

- *Typ W (sichtbar)* bezeichnet ein Workitem mit Dialog. Ein Workitem dieses Typs repräsentiert eine Einzelschrittaufgabe, die einen Benutzerdialog erfordert.
- *Typ D (sichtbar)* bezeichnet eine Terminüberschreitung.
- *Typ A (sichtbar)* ist die Laufzeitrepräsentation eines Arbeitsvorrates. Ein Arbeitsvorrat ist „eine Liste von Objekten, die in einem zeitlich begrenzten Rahmen einmalig und im Gesamtzusammenhang abgearbeitet werden soll“ [SAP98d].
- *Typ F (nicht sichtbar)* ist die Laufzeitrepräsentation einer Mehrschrittaufgabe. Dieser Typ existiert genau einmal für jede Mehrschrittaufgabe und wird als *Workflow-Item* bezeichnet.
- *Typ B (nicht sichtbar)* ist die Laufzeitrepräsentation einer Einzelschrittaufgabe, die keine Benutzereingabe erfordert, und im Hintergrund (Batch) abläuft.
- *Typ E (nicht sichtbar)* repräsentiert einen Warteschritt, bei dem das Eintreten eines Ereignisses abgewartet wird.
- *Typ C (nicht sichtbar)* ist eine Sonderentwicklung im Zusammenhang mit EDI-Anwendungen (siehe [SAP98d]).

5.1.2 Werkzeuge zur Erstellung und Modifikation von Workflows

Zum Anlegen von *Workflow-Definitionen* stehen verschiedene Werkzeuge zur Verfügung:

Der Grafische Editor visualisiert die Workflow-Definition in Form der *ereignisgesteuerten Prozeßketten* (EPKs, siehe Kapitel 2.1.2). In Abbildung 5.2 ist die Workflow-Definition des Workflow-Musters für einen Arbeitsvertrag zu finden. Die einzelnen Ereignisse und Funktionen werden zusätzlich mit den entsprechenden Schritt-Typen versehen.

Mit dem Alphanumerischen Editor können die Schritte, Ereignisse und Operatoren der Workflow-Definition bearbeitet werden. Die Workflow-Definition wird in Form eines Baumes angezeigt.

Die Container-Definition legt die Export- und Importparameter der Workflow-Definition fest. Neben den Systemvariablen (z.B. Initiator, Priorität) werden im Workflow-Container die Referenzen auf die zum Workflow gehörigen Objekte angelegt.

Durch die Initialwerte werden einzelnen Objekten des Workflow-Containers Werte zugeordnet, die zu Beginn des Workflows voreingestellt sind.

Eine ausführliche Beschreibung der verschiedenen Werkzeuge und ein Beispiel für das Anlegen eines Workflows ist in der *SAP Online Hilfe* [SAP98d] enthalten. Neben diesen Werkzeugen kann der *Workflow-Wizard* zur Definition von Teilabschnitten einer *Workflow-Definition* genutzt werden. Im *Workflow-Wizard Repository* sind bereits abgegrenzte Aufgabenstellungen vorhanden, die in eine *Workflow-Definition* eingebunden werden können.

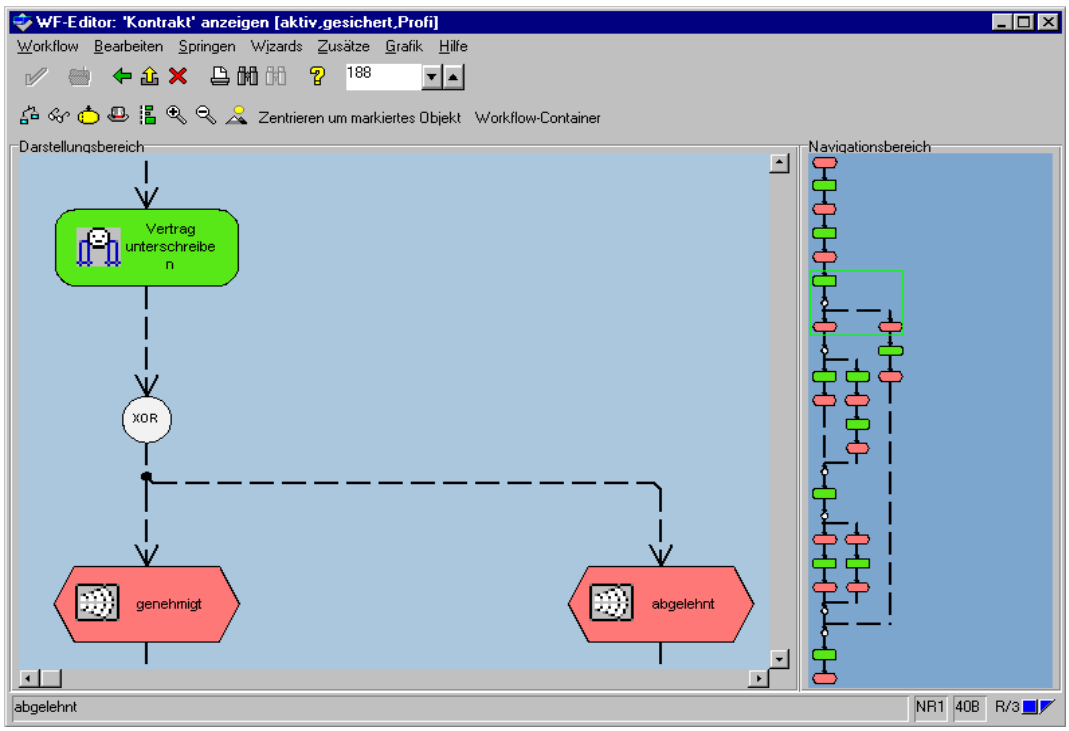

Abbildung 5.2: Grafische Visualisierung einer Workflow-Definition

Um externen Benutzern[7] einen Zugriff auf Workflows zu bieten, können Formulare genutzt werden.[8] Formulare sind Nachbildungen von Papierformularen, wobei die einzelnen Felder eines Formulars durch Werte des Workflow-Containers gefüllt werden. Diese Formulare existieren in zwei Ausprägungen. *Web-Formulare* werden durch einen Webbrowser bearbeitet (vgl. Kapitel 5.5); als Entwicklungswerkzeug dient das *SAP@WebStudio*. Zusätzlich werden Formulare durch *Visual Basic* [Mic98] erstellt, die mittels des *SAPforms Designers* bearbeitet werden; die SAP bezeichnet diese Art der Formulare als *elektronische Formulare*. Der Zugriff auf *elektronische Formulare* kann über einen *Mail-Client* erfolgen, wenn das *E-Mail-System* die Verarbeitung unterstützt, wie beispielsweise der Exchange-Server von Microsoft.

5.1.3 Aufbau und Semantik von Business-Objekten

Geschäftsobjekte (*Business-Objekte*) sind beispielsweise Rechnungen, Mitarbeiter und Debitoren. Die SAP AG definiert im R/3 Umfeld den Begriff *Business-Objekt* folgendermaßen [SAP98e]:

> Repräsentant eines zentralen betriebswirtschaftlichen Objektes der realen Welt. R/3-Business-Objekte beschreiben ganzheitliche betriebswirtschaftliche Zusammenhänge. Diese Kapselung führt zu einer Komplexitätsreduktion, da die innere

[7] Mit externen Benutzern sind in diesem Kontext Benutzer gemeint, die nicht mit dem SAPGUI agieren.
[8] Die Bearbeitung von Formularen ist auch über den SAPGUI möglich.

Strukturierung eines Business-Objekts in weitere, geschachtelte Objekte verborgen bleibt.
Business APIs (BAPIs) sind die Methoden von SAP Business-Objekten, die den Zugriff externer Systeme über Internet, DCOM oder CORBA auf R/3-Funktionalität erlauben.

In der Literatur werden Objekttyp für den Klassenbegriff und Objektinstanz für ein Objekt einer Klasse verwendet. Da es sich bei R/3 um eine betriebswirtschaftliche Software handelt, werden auch Kundenaufträge oder Servicemeldungen auf Business-Objekte abgebildet.

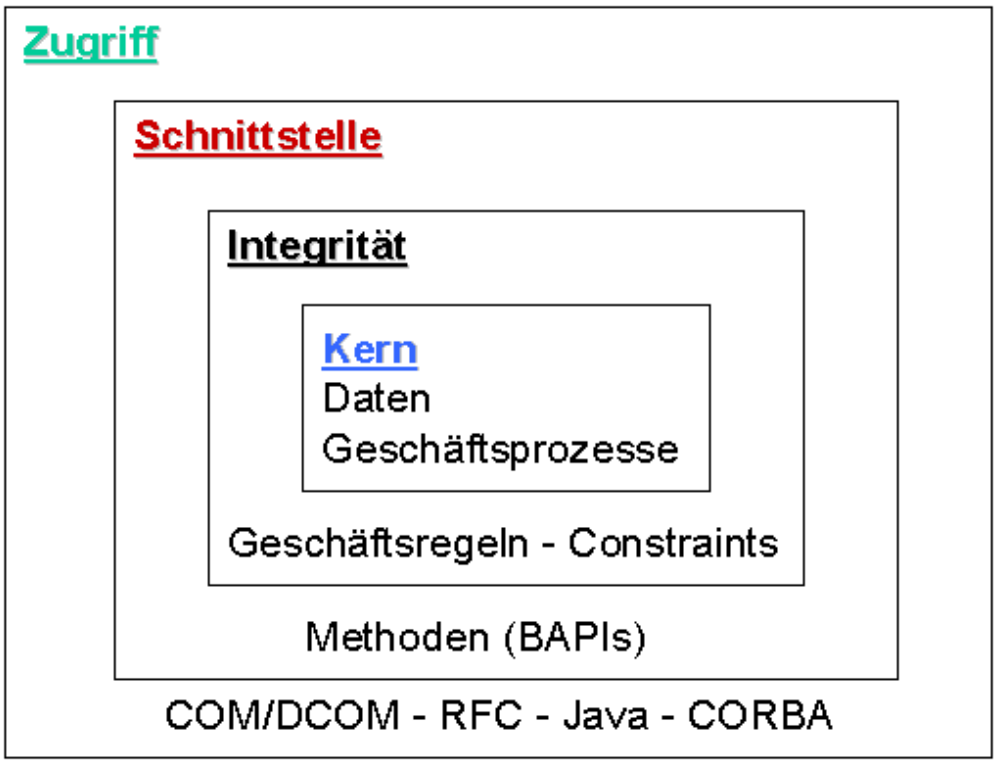

Abbildung 5.3: Aufbau eines Business-Objektes [SAP98d]

Die Struktur eines Business-Objekts zerfällt bei genauerer Betrachtung in vier Ebenen (vgl. Abbildung 5.3). Die innerste Schicht, der Kern eines Business-Objekts, enthält die Daten und Geschäftsprozesse. Die Integrationsschicht stellt sicher, daß keine Geschäftsregeln[9] und Rechte[10] bei Methodenaufrufen verletzt werden. In der Schnittstellenschicht werden die Methoden definiert, mit denen auf den Kern zugegriffen werden kann. Die Zugriffsschicht kapselt die Technologien, mit denen auf ein Business-Objekt zugegriffen werden kann.

[9] Waren können zum Beispiel nur versendet werden, wenn Sie im Lager vorhanden sind.
[10] An dieser Stelle sind Benutzerrechte bzw. Rechte von Anwendungen gemeint. Methoden dürfen nur ausgeführt werden, wenn der Benutzer bzw. die Anwendung die nötigen Berechtigungen besitzt.

Die Eigenschaften eines Business-Objekttyps sind im Folgenden dargestellt:

Objekttyp: Der Objekttyp definiert die Merkmale, die alle Instanzen gemeinsam haben. Im Objekttyp sind neben einem eindeutigen Namen auch ein Datenmodell und eine textliche Definition hinterlegt.

Schnittstellen: Jedes Business-Objekt enthält beispielsweise die Standardschnittstelle *IFSAP* mit den Methoden *ExistenceCheck* und *Display*. Gegebenenfall implementiert ein Business-Objekt weitere Schnittstellen, dessen Methoden explizit implementiert werden müssen.

Schlüsselfelder: Schlüsselfelder bestimmen die Struktur eines eindeutigen Schlüssels. Der Objekttyp *Debitor* besitzt beispielsweise das Schlüsselfeld *DebitorId*.

Methoden: Methoden sind die Funktionen bzw. Operationen, die ein Business-Objekt versteht. Teilweise zählen diese Methoden zu den BAPIs, jedoch gibt es Methoden, die noch nicht über eine API erreichbar sind.

Ereignisse: Ereignisse melden die Statusänderungen eines Business-Objekts. In der Kombination mit dem Workflow-System können durch diese Ereignisse Workflows angestoßen werden [SAP98d].

Alle Business-Objekttypen sind im *Business Object Repository (BOR)* verankert, welches das Pendant zum Parameter Repository (vgl. Buck-Emden [BE99]) darstellt. Ursprünglich (R/3 Release 3.0) wurde das BOR nur vom Business Workflow genutzt, welches sich jedoch mit der Einführung von BAPIs in R/3 Release 3.1 änderte. Der *Business-Object Browser* ist wiederum das Pendant zum *Repository Browser*. Allerdings sind die Business-Objekte nach den einzelnen Modulen und Komponenten des R/3 Systems baumartig gelistet, während im Repository die Aufteilung nach Programmen, Funktionsgruppen, Screens, etc. erfolgt.

Auch Vererbung und Polymorphismus spielen bei Business-Objekten eine wesentliche Rolle. So erbt der Objekttyp *Employee* von dem Objekttyp *Person*. *Employee* ist also ein *Subtyp* von *Person* und erbt alle Eigenschaften. Durch Polymorphismus [CW85] ist es möglich, Methoden, die von einem *Supertyp* geerbt wurden, zu überschreiben. Das Anlegen von eigenen Business-Objekten erlaubt das Abbilden von unternehmensspezifischen Anforderungen in das *BOR*. Da das Ändern von *Business-Objekten* des Standardsystems nicht erlaubt ist, muß, wenn die Methoden oder Attribute eines vorhandenen *Business-Objektes* nicht ausreichend sind, eine Kopie als *Subtyp* angelegt und die Kopie dementsprechend modifiziert werden. Wird der *Subtyp* an den *Supertyp* delegiert, so werden anschließend alle Zugriffe auf den *Supertyp* automatisch an den *Subtyp* weitergeleitet.

5.2 Erweiterung der Standardfunktionalitäten

Die Funktionalität der Standardsoftware ist durch die vorgegebenen Geschäftsprozesse eingeschränkt. Reichen die Funktionalitäten der Standardsoftware nicht aus, um spezielle beziehungsweise unternehmensspezifische Geschäftsprozesse im System abzubilden, so muß die

Standardsoftware erweitert werden. Da Herstellern von Standardsoftwaresystemen bekannt ist, daß Unternehmen mit den Standardgeschäftsprozessen nicht alle gewünschten Funktionalitäten in das Softwaresystem abbilden können, werden verschiedene Werkzeuge und Mechanismen zur Erweiterung der Geschäftsprozesse in das System integriert. Schnittstellen und leere Prozedurhüllen sind solche Erweiterungsstellen, die vom Hersteller für eine Erweiterung vorgesehen sind.

Das Einfügen von leeren Prozedurhüllen ermöglicht zwar das Erweitern des Softwaresystems an einer bestimmten Stelle, jedoch können bei einem Releasewechsel Komplikationen auftreten, wenn diese Erweiterungen weiterhin unterstützt werden sollen. Um diese Problematik ansatzweise zu lösen, werden Erweiterungen zunächst in zwei Klassen aufgeteilt:

1. *Releaseunabhängige Erweiterungen* adaptieren das Standardsoftwaresystem, ohne bei einem Releasewechsel die Kompatibilität der Erweiterung zu verlieren. Diese Art der Erweiterung wird durch leere Hüllen in dem Standardsoftwaresystem ermöglicht, die eine definierte Aufrufschnittstelle haben und nicht auf alle Objekte der Standardsoftware zugreifen können. Diese Schnittstelle muß in den folgenden Releases der Standardsoftware gleich bleiben, damit die Kompatibilität gewährleistet ist. Das Erweitern der Standardsoftware durch zusätzliche Menüfunktionen, Bildschirmdialoge, etc. muß auf der Basis dieser Schnittstellen in das Softwaresystem implementiert werden.

   Ein Zugriff der Erweiterung auf beispielsweise Tabellen muß ausgeschlossen sein, da sich zwischen den Releases die Definitionen der Tabellen oder die Tabellennamen ändern können und somit eine Kompatibilität nicht gewährleistet werden kann.

2. *Releaseabhängige Erweiterungen* adaptieren das Standardsystem und nutzen dabei Methoden und Techniken, die keine Kompatibilität zu einem nächsten Release gewährleisten können. Das Modifizieren von Objekten des Standards oder der Zugriff auf sich ändernde Objekte sind Beispiele für releaseabhängige Erweiterungen.

Im R/3 System stehen für die erste Klasse der Erweiterung, die sogenannten *Customer-Exits* zur Verfügung. Die zweite Klasse, der releaseabhängigen Erweiterungen wird durch die *User-Exits* und durch die Veränderung der Standardobjekte abgedeckt.

Die SAP AG definiert *Customer-Exits* wie folgt [SAP98e]:

> Vorgedachte Erweiterung des SAP-Standards.
>
> Potentielle Kundenanforderungen, die nicht als Standard ausgeprägt sind, werden als 'leere Modifikationshülse' im Standard vorgedacht und können beim Kunden mit kundenspezifischer Logik ausgestaltet werden.
>
> Customer-Exits werden von SAP in der Transaktion SMOD angelegt. In der Transaktion CMOD wählt der Kunde aus den SAP-Erweiterungen diejenigen aus, die er bearbeiten möchte, stellt sie zu Erweiterungsprojekten zusammen, bearbeitet die Komponenten, und aktiviert die entsprechenden Projekte.
>
> Die Aufwärtskompatibilität wird durch die Verwendung von *Customer-Exits* gewährleistet, d.h. der Aufruf eines Exits aus der Standardsoftware sowie die Gültigkeit der Aufrufschnittstelle bleiben auch in zukünftigen Release-Ständen erhalten.

Die releaseabhängigen *User-Exits* werden wie folgt definiert [SAP98e]:

> Zeitpunkt in einem SAP-Programm, zu dem ein kundeneigenes Programm aufgerufen werden kann.
>
> Im Gegensatz zu Customer-Exits kann der Entwickler über User-Exits auf Programmteile und Datenobjekte des Standards zugreifen. Beim Upgrade muß jeweils geprüft werden, ob die Erweiterung durch einen User-Exit zum Standard paßt.
>
> Es werden zwei Formen von User-Exits unterschieden:
>
> - User-Exits mit Include Member-Technik.
>   Diese sind Kundenerweiterungen, die direkt im Programm aufgerufen werden.
> - User-Exits mit Tabellentechnik.
>   Diese werden direkt über das Customizing verwendet und verwaltet.

Die beiden Definitionen grenzen *Customer-Exits* gegenüber den *User-Exits* durch die Releaseabhängigkeit ab. Innerhalb des R/3 Systems und der Dokumentation [SAP98d] ist diese Abgrenzung der Begriffe jedoch nicht gegeben. Die Begriffe werden oftmals synonym verwendet und ein *User-Exit* wird teilweise mit einem *Customer-Exit* gleichgesetzt.

Im Folgenden werden die verschiedenen Arten der *Customer-Exits* beschrieben und ein Beispiel für die Anwendung eines *Customer-Exits* gegeben. Das Erweitern des R/3 Systems mit *User-Exits* gleicht teilweise der Erweiterung mit *Customer-Exits*, da auch bei *Customer-Exits* von *Includes*[11] Gebrauch gemacht wird.

Customer-Exits lassen sich in die folgenden Kategorien einteilen:

Texterweiterungen erlauben das Ändern von Beschreibungen zu Datenelementen des *Dictionaries* und das Hinterlegen zusätzlicher Dokumentation. Die Beschreibung zu Datenelementen und die Dokumentation wird im SAPGUI angezeigt, wenn auf einem Eingabefeld die Funktionstaste `F1` gedrückt wird. Das Abbilden des Sprachgebrauchs eines Unternehmens in das R/3 System wird durch *Texterweiterungen* realisiert.

Feldexits ermöglichen das Hinterlegen von Funktionslogik in einem Feld, das an ein Datenelement gebunden ist, eines *Screens*[12]. Benutzerspezifische Prüfungen oder das Fortschreiben für Statistiken wird mit *Feldexits* in das System integriert.

Funktionsexits werden mittels des ABAP Befehls `CALL CUSTOMER-FUNCTION 'nnn'`[13] aufgerufen. Das Programm verzweigt bei einem Funktionsexit in das kundeneigene Programm, wenn dieser Exit vom Kunden aktiviert wurde. Der Funktionsexit selbst enthält den

[11] Includes sind spezielle Unterprogramme von ABAP. Includes ermöglichen verschiedenen Anwendungen, auf einen gemeinsamen Datenbestand zuzugreifen. Hierzu wird der Include in den einzelnen Programmen mit dem gleichnamigen Include-Befehl der Programmiersprache ABAP aufgerufen.

[12] Ein Screen ist ein ABAP Programm, das aus einem Bildschirmbild und der unterliegenden Ablauflogik besteht. Im deutschen Sprachgebrauch wird DynPro synonym benutzt

[13] Ein Funktionsexit wird innerhalb eines Moduls durch eine dreistellige Nummer (`'nnn'`) eindeutig identifiziert.

ABAP Befehl `INCLUDE`, der ein Unterprogramm mit einer definierten Schnittstelle aufruft.
Der Kunde kann in dem Unterprogramm seine eigene Funktionslogik hinterlegen. Theoretisch können auch Objekte genutzt werden, die nicht in der Schnittstelle definiert sind, jedoch nur Objekte der Datenbank und keine Objekte des aktuellen Programmes.

Menüerweiterungen bieten die Möglichkeit, an vorgedachten Stellen eigene Menüpunkte hinzuzufügen. Das Einbinden von kundeneigenen Menüs in die Standard-Menüs der SAP AG wird über *Menüerweiterungen* realisiert.

Dynproerweiterungen werden mit dem ABAP Befehl `CALL CUSTOMER-SUBSCREEN` zum Zeitpunkt PAI[14] ermöglicht. Das zusätzliche Einbinden von kundeneigenen Bildschirmbildern, zum Erfassen von Daten oder Anzeigen von Informationen wird mit den *Dynproerweiterungen* realisiert.

Tabellenerweiterungen ergänzen Tabellen oder Strukturen[15] mit beliebigen Feldern. Tabellenerweiterungen sind mittels *Append-Strukturen* oder *Customizing-Includes* möglich. *Append-Strukturen* sind *genau einer* Tabelle zugeordnet, wobei zu einer Tabelle mehrere *Append-Strukturen* vorhanden sein können, und werden vom Kunden angelegt. *Customizing-Includes* hingegen sind von der SAP AG vordefiniert und im Initialzustand leer. Ein *Customizing-Include* kann mehreren Tabellen zugeordnet sein.

Diese verschiedenen Erweiterungen werden über die Transaktion *smod* gepflegt. In einer Erweiterung werden verschiedene der oben aufgeführten Erweiterungstypen (Komponenten) zu einer Erweiterung zusammengefaßt. Im R/3 System sind im Initialzustand bereits Erweiterungen vorhanden, die mittels der Transaktion *smod* weiterbearbeitet werden. Ist ein *Customer-Exit* im System nicht integriert, so bietet die SAP AG die Möglichkeit, einen *Customer-Exit* zu beantragen. Die Integration dieses *Customer-Exit* erfolgt, wenn die SAP AG den *Customer-Exit* für sinnvoll erachtet, im nächsten Release.

In der Transaktion *cmod* werden eine oder mehrere Erweiterungen zu einem Projekt zusammengefaßt und zum Beispiel die Quellcodes eines Funktionsexits mit der entsprechenden Funktionslogik gefüllt.

Die Erweiterung *SAPMF02D* ermöglicht beispielsweise das Prüfen von Eingaben vor dem Speichern eines Debitors. Diese Erweiterung enthält den Funktionsexit `EXIT_SAPMF02D_001`. Der Quellcode des Funktionsexits ist in Abbildung 5.4 zu finden. Die definierte Schnittstelle ermöglicht das Prüfen der Einträge, die später in die Tabellen *KNA1,KNB1* und *KNVV* übernommen werden.

Reicht die Mächtigkeit der vorgedachten Erweiterungen nicht aus, so bietet das R/3 System einen Zugriff auf alle Objekte des R/3 Systems. Auch Änderungen an Objekten des Standards können auf diesem Wege durchgeführt werden. Diese Methode ist nur anzuwenden, wenn alle anderen Möglichkeiten der Erweiterungen berücksichtigt wurden und keine Methode das gewünschte Ergebnis erzielen kann. Das Einspielen von *Hot-Packages* oder *Legal Changes*

[14] PAI - Process After Input (vgl. Matzke [Mat98])

[15] Strukturen enthalten keine Datensätze auf der Datenbank. Sie dienen ausschließlich der Schnittstellendefinition von Programmen (siehe [SAP98d].

```
FUNCTION EXIT_SAPMF02D_001.
*"----------------------------------------------------------------------
*"*"Lokale Schnittstelle:
*"       IMPORTING
*"             VALUE(I_KNA1) LIKE  KNA1 STRUCTURE  KNA1
*"             VALUE(I_KNB1) LIKE  KNB1 STRUCTURE  KNB1
*"             VALUE(I_KNVV) LIKE  KNVV STRUCTURE  KNVV
*"       TABLES
*"              T_KNBK STRUCTURE  KNBK
*"----------------------------------------------------------------------

  INCLUDE ZXF04U01.

ENDFUNCTION.
```

Abbildung 5.4: ABAP-Quellcode eines Funktionsexits

Patches[16] kann Änderungen in das System einspielen, die den produktiven Betrieb eventuell einschränken. Einzelne R/3 Programme können geändert werden, die nach dem Einspielen durch das Ändern der Objekte des Standards nicht mehr funktionstüchtig sind.

Mittels des „*Change and Transport Organizers*“ (Transaktion `SE03`) ist es möglich die Parameter für die Systemänderbarkeit zu ersetzen. Initial ist das Ändern von R/3 Standardprogrammen nicht erlaubt. Werden die entsprechenden Parameter im „*Change and Transport Organizer*“ geändert, so können auch R/3 Standardanwendungen modifiziert werden.

5.3 Regeln zur Validierung und Substitution

Um Benutzereingaben zu überprüfen oder zu ersetzen, stehen in Standardsoftwaresystemen die Konzepte der *Validierung* und *Substitution* zur Verfügung. Zur Überprüfung bzw. Verprobung werden *Boolesche Aussagen* definiert, um die Benutzereingaben zu bewerten. *Validierungen* ermöglichen nach der Verprobung die Ausgabe von Meldungen; *Substitutionen* hingegen ermöglicht das Ersetzen von Eingaben. *Validierungen* und *Substitutionen* werden stets vor dem Schreiben auf die Datenbank durchgeführt.

5.3.1 Überprüfung von Eingaben durch Validierung

Bei der Eingabe von Daten in das R/3 System werden Daten gegen standardisierte Regeln überprüft (validiert). Um benutzerspezifische Abfragen, also *Validierungen*, einzubauen, können zusätzliche *Validierungsregeln* im *Regel-Manager* hinterlegt werden. Die Systemkomponente hinter den *Validierungen* und *Substitutionen* (siehe Kapitel 5.3.2) ist das Finanz-

[16] Legal Changes Patchess passen R/3 Systeme mit dem Modul HR (Personalwesen) an die gesetzlichen Änderungen an und somit für einen produktiven Betrieb unumgänglich.

wesen-Spezielle Ledger[17] (FI-SL). Diese Komponente dient als Werkzeug, um Standardfunktionalitäten zu erweitern, ohne das Standardsystem zu verändern. Das *Berichtswesen* bzw. *Reporting* (siehe Kapitel 4.3) zählt ebenso zu FI-SL.

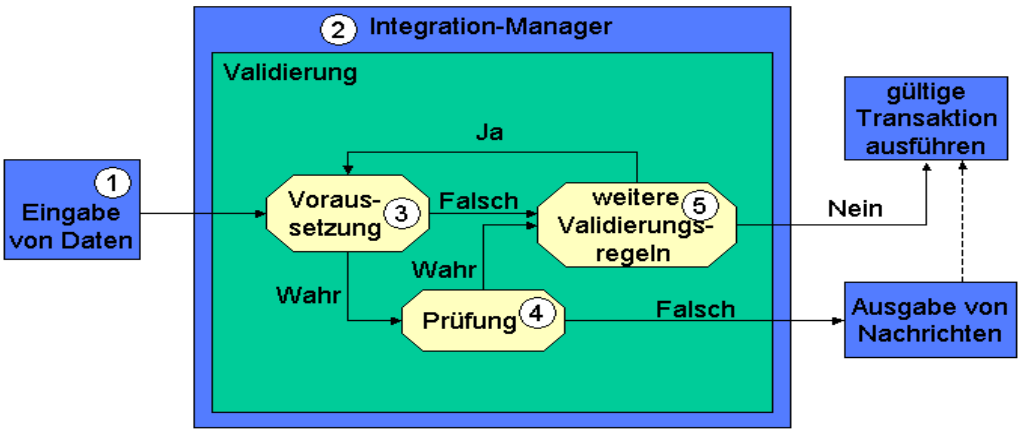

Abbildung 5.5: Ablaufdiagramm einer Validierung

Eine *Validierung* besteht aus einem oder bis zu 999 *Validierungsschritten*. Ein *Validierungsschritt* unterteilt sich wiederum in eine *Voraussetzung* und eine *Prüfung*. Die *Voraussetzung* selektiert die zu verprobenden Eingabewerte. Die *Prüfung* verprobt die vorselektierten Werte gegen die boolesche Aussage. Die einzelnen Schritte einer *Validierung* (siehe Abbildung 5.5) sind wie folgt:

1. Die eingegebenen Daten werden an das R/3 System übergeben. Handelt es sich bei den eingegebenen Daten um Werte, die verprobt werden müssen, so wird der Integrationsmanager gerufen.

2. Der Integrationsmanager, als zuständige Komponente des R/3 Systems, erhält die zugehörigen *Validierungsregeln*, vom Regel-Manager, und den eingegebenen Wert.

3. Wird die Voraussetzung bei der Verprobung erfüllt (*Wahr*), so erfolgt die Verprobung gegen die Prüfaussage. Ist die Voraussetzung nicht erfüllt (*Falsch*), ist der Wert gültig für diese Voraussetzung.

4. Bei der Prüfaussage wird der Wert gegen einen booleschen Ausdruck verprobt. Ist das Ergebnis *Wahr*, so ist der Wert für die Prüfaussage gültig. Ergibt die Prüfaussage den Wert *Falsch*, wird der Wert nicht verbucht (der Wert ist ungültig) und eine Nachricht wird ausgegeben. Die folgenden drei Nachrichtentypen stehen zur Verfügung:

   E- Nachrichten (*Error*) geben eine Fehlermeldung aus. Die Eingaben müssen, um eine Verbuchung zu ermöglichen, korrigiert werden.

[17] *Ledger* ist das englische Wort für (Haupt-)Buch. Im Bereich des Finanzwesens ist beispielsweise mit *Generell Ledger* die Hauptbuchhaltung bzw. das Hauptbuch gemeint. Wird im Folgenden das Wort *Ledger* verwendet, so ist dies in buchhalterischer Sicht gemeint.

A- Nachrichten (*Abbend*) bewirken einen Abbruch des Validierungsprozesses. Die eingegebenen Daten werden nicht verbucht und eine entsprechende Nachricht wird ausgegeben.

W- Nachrichten (*Warning*) zeigen eine Warnung, daß die Eingabe korrigiert werden soll (nicht *muß*!). Die Verbuchung kann ohne Korrektur fortgesetzt werden.

5. Liegen weitere *Validierungsregeln* für diesen Wert vor, so wird der Prozeß erneut durchlaufen. Andernfalls kann die Verarbeitung fortfahren.

Das Anlegen, Ändern und Löschen von *Validierungen* erfolgt im *Customizing* (siehe Kapitel 4.1) des R/3 Systems. Von der Wurzel des Referenz-IMGs ausgehend unter

> Finanzwesen ⇒ Spezielle Ledger ⇒ Werkzeuge ⇒ Validierung/Substitution/Regeln pflegen ⇒ Validierung pflegen

kann die Transaktion gestartet werden. Für das Editieren von Einzelschritten einer *Validierung* wird dem Entwickler als Hilfsmittel ein Formeleditor zur Seite gestellt, dem alle erlaubten Funktionen bekannt sind. Die Simulation der *Validierung* ist ebenso möglich.

Das Anlegen einer *Validierung* aktiviert diese jedoch noch nicht. *Validierungen* können entweder *global* oder *lokal* aktiviert werden. *Global* meint, das die *Validierung* für eine Gesellschaft[18] aktiviert wird; *lokal* aktiviert die *Validierung* für einen Buchungskreis[19]. Das Pflegen von *Validierungen*, wobei Pflegen sowohl das Aktivieren und Deaktivieren beinhaltet, geschieht wiederum im *Customizing*. Der zugehörige Pfad von der Wurzel des Referenz-IMGs ist

> Finanzwesen ⇒ Spezielle Ledger ⇒ Grundeinstellungen ⇒ Validierung ⇒ Lokale Validierung pflegen

für lokale *Validierungen* und

> Finanzwesen ⇒ Spezielle Ledger ⇒ Grundeinstellungen ⇒ Validierung ⇒ Globale Validierung pflegen

für globale *Validierungen*.

Der Regel-Manager bietet außerhalb der booleschen Ausdrücke eine Erweiterung, um *Validierungsregeln* mächtiger zu machen. Durch den Einsatz von *Benutzer-Exits* (vgl. auch Kapitel 5.2) ist die Möglichkeit gegeben, *Form-Routinen*[20] zu nutzen, um ABAP/4 Programme für *Validierungsregeln* einzusetzen. *Benutzer-Exits* unterteilen sich in *Validierungs-Exits* und *Substitutions-Exits* (siehe Kapitel 5.3.2).

[18] Eine Gesellschaft ist die kleinste organisatorische Einheit, für die nach der jeweiligen Handelsgesetzgebung ein Einzelabschluß zu erstellen ist. Eine Gesellschaft kann einen oder mehrere Buchungskreise umfassen[SAP98e].

[19] Ein Buchungskreis ist die kleinste organisatorische Einheit des externen Rechnungswesens, für die eine vollständige, in sich abgeschlossene Buchhaltung, abgebildet werden kann[SAP98e].

[20] Form-Routinen sind Unterprogramme in ABAP/4, die mit Prozeduren aus Pascal vergleichbar sind. Parameterübergabe via *Call-by-Reference* und *Call-by-Value* sind möglich. Die Rückgabe von Funktionswerten ist jedoch nicht möglich.

In einem *Validierungs-Exit* können beliebige ABAP/4 Funktionen genutzt werden, die auf alle Objekte des Standard zugreifen können. Folgende Regeln schränken *Validierungs-Exits* ein:

Namensraum: Die einzelnen Form-Routinen müssen bei kundeneigenen Exits mit einem `U` beginnen und von maximal vier Zahlen gefolgt deklariert werden[21]. SAPeigene Exits beginnen mit einem `'S'`. Des weiteren muß der Name des Formpools[22] in die mandantenabhängige Tabelle `T80D` eingetragen werden.

Aufruf: Ein *Validierungs-Exit* wird mit `FORM Uzzzz USING B_RESULT` aufgerufen.
Der Wert `B_RESULT` wird via *Call-by-Reference* übergeben und kann innerhalb der Form-Routine die Werte `B_TRUE` oder `B_FALSE` annehmen[23].

Datendeklaration: Das Deklarieren von Strukturen und Tabellen ist innerhalb einer Form-Routine möglich. Empfohlen[SAP98d] wird jedoch, die Deklarationen im äußeren Programm zu machen, damit der Inhalt auch von dem rufenden äußeren Programm verwendet werden kann.

Parameterdefinition: Alle Parameter und Namen der *Benutzer-Exits* müssen in der vorhandenen Form-Routine `GET_EXIT_TITLES` bekannt gemacht werden, um dem Code-Generator, der den Aufruf generiert, diese Werte zugänglich zu machen.

5.3.2 Ersetzen von Eingaben durch Substitution

Wie bei der *Validierung* von Dateneingaben, werden auch bei der *Substitution* Dateneingaben durch *Substitutionregeln* überprüft. Im Gegensatz zur *Validierung* kann eine *Substitution* nicht zum Abbruch einer Buchung führen oder Nachrichten ausgeben, sondern ermöglicht das Ersetzen von Werten. *Substitutionen* werden immer nach *Validierungen* ausgeführt, um zunächst eine Verprobung der zu substituierenden Werte zu ermöglichen.

Eine *Substitution* besteht wiederum (vgl. Kapitel 5.3.1) aus 999 *Substitutionsschritten*, die nacheinander ausgeführt werden. Ein *Substitutionsschritt* unterteilt sich in eine *Voraussetzung* und in einen oder mehrere *Substitutionswert(e)*. Die Voraussetzung prüft zunächst, ob eine *Substitution* mit den *Substitutionswerten* durchgeführt wird. Der Ablauf einer *Substitution* (siehe Abbildung 5.6) ist:

1. Die eingegebenen Daten werden an das R/3 System übergeben. Handelt es sich bei den eingegebenen Daten um Werte, die *substituiert* werden müssen, so wird der Integrationsmanager gerufen.

2. Der Integrationsmanager, als zuständige Komponente des R/3 Systems, erhält die zugehörigen *Substitutionsregeln* vom Regel-Manager und den eingegebenen Wert.

[21] Anmerkung: Die Dokumentation[SAP98d] gibt die Syntax sowohl mit drei, als auch mit vier Zahlen an.
[22] Ein Formpool faßt mehrere Form-Routinen zusammen
[23] `B_RESULT` ist eine Zeichenkette der Länge eins. `B_TRUE` hat den Wert `'T'` und `B_FALSE` den Wert `'F'`. Da ABAP/4 keine booleschen Werte (*true* und *false*) kennt, wurde diese Lösung propagiert, um Fehler bei Wertzuweisungen auf `B_RESULT` zu vermeiden (auch Groß/Kleinschreibung relevant!).

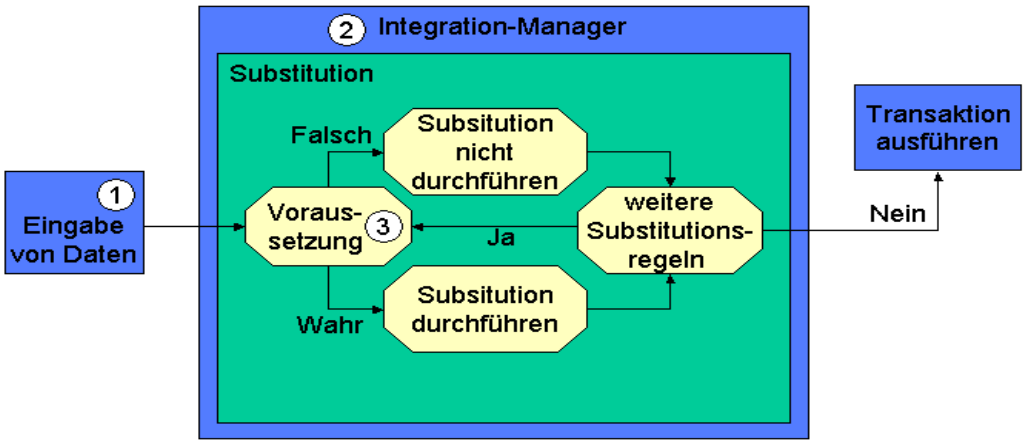

Abbildung 5.6: Ablaufdiagramm einer Substitution

3. Wird die Voraussetzung bei der Verprobung erfüllt (*Wahr*), so erfolgt die *Substitution*. Ist die Voraussetzung nicht erfüllt (*Falsch*), ist der Wert gültig für diese Voraussetzung und die *Substitution* wird nicht durchgeführt.

4. Sind weitere *Substitutionsregeln* vorhanden, so wird der Prozeß erneut durchlaufen; andernfalls kann die Verarbeitung fortfahren.

Das Anlegen, Ändern und Löschen von *Substitutionen* erfolgt, wie bei *Validierungen*, im *Customizing* (siehe Kapitel 4.1) des R/3 Systems. Von der Wurzel des Referenz-IMGs ausgehend, kann unter

> Finanzwesen ⇒ Spezielle Ledger ⇒ Werkzeuge ⇒ Validierung/Substitution/Regeln pflegen ⇒ Validierung pflegen

die Transaktion gestartet werden. Der Formeleditor und die Möglichkeit der Simulation stehen auch hier zur Verfügung.

Wie bereits bei *Validierungen*, müssen *Substitutionen* ebenfalls nach der Generierung aktiviert werden. Die entsprechenden Pfade im Referenz-IMG für die Pflege der globalen (bezogen auf Gesellschaften) und lokalen (bezogen auf Buchungskreise) *Substitutionen* sind:

> Finanzwesen ⇒ Spezielle Ledger ⇒ Grundeinstellungen ⇒ Substitution ⇒ Lokale Substitution pflegen,
> beziehungsweise
> Finanzwesen ⇒ Spezielle Ledger ⇒ Grundeinstellungen ⇒ Substitution ⇒ Globale Substitution pflegen.

Substitutions-Exits sind wie *Validierungs-Exits* eine Erweiterung. Es handelt sich wiederum um Form-Routinen, die ähnlichen Restriktionen unterliegen:

Namensraum: - siehe *Validierung* -

Aufruf: Ein *Substitutions-Exit* wird `FORM Uzzzz USING COST_CENTER.` aufgerufen.
Der Parameter `COST_CENTER` enthält den, via *Call-by-Reference* übergebenen, zu substituierenden Wert.

Deklaration: - siehe *Validierung* -

Parameterdefinition: - siehe *Validierung* -

5.4 Personalisierung der Standardsoftware

Die Personalisierung von Standardsoftwaresystemen ermöglicht es, dem Benutzer eine für ihn zugeschnittene Oberfläche zu präsentieren. Benutzer, die täglich mit einer Standardsoftware arbeiten, müssen die Oberfläche auf ihre Bedürfnisse zuschneiden können, um häufig benutzte Transaktionen und Menüpunkte schnell zu erreichen und nicht durch lange Menüpfade zur gewünschten Funktion zu gelangen. Auch die Überflutung des Benutzers durch Menüeinträge, die der Benutzer nicht benutzt bzw. nicht benutzen darf, kann durch die Personalisierung vermieden werden.

Im R/3 System kann aus dem Unternehmensmenü (vgl. Kapitel 4.4.3), welches bereits durch den Unternehmens-IMG eingeschränkt ist, ein Benutzermenü generiert werden. Benutzermenüs sind gültig für eine Aktivitätsgruppe, die dem Benutzerstammsatz (vgl. Kapitel 4.4.3) eines Benutzers zugeordnet wird.

Abbildung 5.7: Der SAP Session Manager

Seit R/3 Release 4.0 steht neben dem SAPGUI als weiteres Werkzeug der *SAP Session Manager* zur Verfügung (siehe Abbildung 5.7). Der *SAP Session Manager* dient als Navigationswerkzeug, um Benutzersitzungen (*Modi*) zu verwalten und neue Transaktionen zu starten. Zusätzlich bietet der *SAP Session Manager* einen Zugriff auf die verschiedenen Menüstrukturen (Benutzermenü, Unternehmensmenü, SAP-Menü). In die Ablage (vgl. Abbildung 5.7) kann der Benutzer eigene Funktionen einfügen, um einen schnelleren Zugriff zu erhalten. Die Modusliste[24] erlaubt das Navigieren zwischen den geöffneten *Modi*.

[24] Der SAPGUI erlaubt das Öffnen mehrerer Fenster, in denen unabhängig voneinander Transaktionen durchgeführt werden. Ein solches Fenster wird als Modus bezeichnet.

Neben der Ablage im *SAP Session Manager* existiert im SAPGUI das *dynamische Menü*, dessen Funktionalität mit der Ablage des *SAP Session Managers* vergleichbar ist. Neben dem SAP-Menü enthält das *dynamische Menü* eine Favoritenliste, die von jedem Benutzer autonom verwaltet wird. Die Favoritenliste des *dynamischen Menüs* entspricht der Favoritenliste (*Bookmark-Liste*) eines Webbrowsers. Einträge können entweder aus dem SAP-Menü übernommen oder explizit mit dem entsprechenden Transaktionscode hinzugefügt werden.

Das Ausblenden von bestimmten Menüs für einzelne oder alle Benutzer ist über die Pflege der Tabelle *SSM_USR* für einzelne Benutzer und der Tabelle *SSM_Cust* für alle Benutzer möglich.

5.5 Anbindung von Standardsoftwaresystemen an das Internet

Für eine Anbindung des R/3 Systems an das Internet wurde der *Internet Transaction Server* (kurz ITS) entwickelt. Theoretisch ist auch die Nutzung des SAPGUIs über das Internet möglich. Der Java-SAPGUI vermeidet zusätzlich die Installation eines GUIs auf dem Client. Allerdings wurde die Weiterentwicklung des Java-SAPGUIs laut der SAP AG[25] eingestellt.

Bei der Nutzung des ITS bleibt die Anwendungslogik im R/3 System verborgen. Es müssen also ABAP Programme geschrieben bzw. bestehende Programme genutzt werden. Für den umgekehrten Ansatz, also die Anwendungslogik außerhalb des R/3 System aufzubauen, stehen Methoden, die in Kapitels 6 beschrieben werden, zur Verfügung.

Ziel des ITS ist nicht nur die Anbindung von Mitarbeitern, die keinen SAPGUI zur Verfügung haben und zum Beispiel an Workflows beteiligt werden sollen, sondern auch, eine Verbindung zu Kunden und Lieferanten herzustellen. Gerade die Integration von Kunden durch Online-Bestellsysteme, um Bestandsabfragen, Aufträge, etc. durchzuführen, hat in den letzten Jahren an Bedeutung gewonnen.

Im Folgenden wird zunächst die Architektur des ITS beschrieben und die HTML-Erweiterung *HTML-Business* kurz skizziert. Anschließend folgen die verschiedenen Möglichkeiten, über den ITS mit einem R/3 System zu kommunizieren, und eine Beschreibung der verschiedenen *Internet Application Components (IACs)*.

5.5.1 Architektur eines Servers zur Anbindung an das Internet

Die Architektur des ITS setzt sich aus zwei Komponenten zusammen (siehe Abbildung 5.8).

Das *WGate (Web Gateway)* ist die Schnittstelle zum HTTP-Server und bildet das Bindeglied zwischen dem HTTP-Server und dem AGate. Das WGate nimmt die Anfragen des HTTP-Servers entgegen und wandelt diese Anfrage in ein für das AGate kompatibles Format um. In der entgegengesetzten Richtung nimmt das WGate die bereits fertigen HTML-Seiten des AGates entgegen und leitet diese an den HTTP-Server weiter.

[25] Auf Anfrage nach dem Java-SAPGUI antwortete die SAP AG im Juni 1999, daß die Weiterentwicklung des Java-SAPGUIs eingestellt worden sei und als Internet Anbindung der ITS genutzt werden soll. Im Juli 1999 folgte jedoch eine weitere Beta-Version des Java-SAPGUIs, der über den FTP-Server der SAP AG zu beziehen ist.

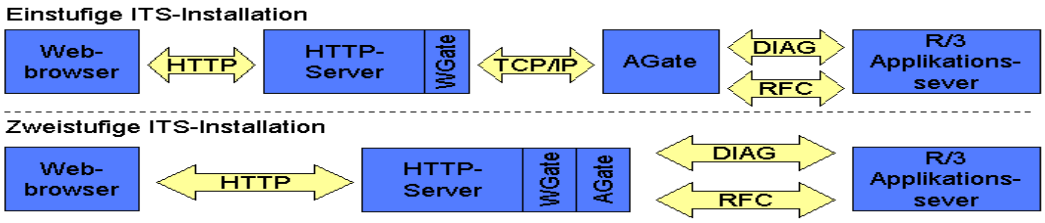

Abbildung 5.8: Architektur des Internet Transaction Servers

Das WGate kann auf HTTP-Servern verschiedener Hersteller installiert werden, die über ein *Common Gateway Interface (CGI)* verfügen. Für *Netscape Server* und den *Microsoft Internet Information Server* stehen Module zur Verfügung, die sich an die NSAPI[26] bzw. ISAPI[27] binden und performanter als eine Anbindung über CGI sind.

Das *AGate (Application Gateway)* kommuniziert mit dem R/3 System und dem WGate. Das Protokoll für die Verbindung zwischen AGate und WGate ist TCP/IP. Die Verbindung zwischen dem AGate und dem R/3 System wird entweder über RFC (siehe Kapitel 6.2.2) oder DIAG[28] abgewickelt. Im Gegensatz zum WGate ist das AGate nur für Windows NT von Microsoft verfügbar und zum Zeitpunkt dieser Arbeit noch nicht auf andere Betriebssysteme portiert worden.

Wie bei einem Applikationsserver kontrolliert ein Dispatcher die Prozesse und Threads des AGates. Diese Threads sind nicht an eine Benutzer-Sitzung gebunden, sondern können verschiedene Anfragen bearbeiten. Die Zuordnung der Benutzer zu einer Sitzung übernimmt der *Session Manager* des AGates, der ebenso für die Anmeldung am R/3 System zuständig ist. Die Zuordnung eines Webbrowsers zu der entsprechenden Sitzung auf dem Server erfolgt über *Cookies* [KM97].

Neben dem R/3 internen Teil einer Transaktion über einen Webbrowser werden auf dem AGate weitere Bausteine benötigt. Alle im Folgenden beschriebenen Bausteine können mittels des *SAP@Web Studio* erstellt werden, welches von der SAP AG zusammen mit dem ITS ausgeliefert wird. Die Summe der folgenden Bausteine wird als *ITS-Service* bezeichnet.

HTML-Business Interpreter und HTML-Templates

Auf dem AGate sind zur Erstellung der HTML-Seiten, die an das WGate weitergeleitet werden, sogenannte HTML-Templates gespeichert. HTML-Templates bestehen aus einer Kombination von standardisiertem HTML (siehe [Wor99]) und der Erweiterung HTML-Business der SAP AG. Die HTML-Business Ausdrücke werden vom HTML-Business Interpreter des AGate interpretiert und durch die vom R/3 System erhaltenen Daten ersetzt. Des weiteren können auf dem AGate sprachabhängige Texte vorhanden sein, die ebenso durch HTML-Business Ausdrücke gekennzeichnet sind und erst zur Laufzeit eingefügt werden. Eine kurze

[26] NSAPI - Netscape Server API
[27] ISAPI - Internet Server API
[28] DIAG - Dynamisches Informations- und Aktions-Gateway. DIAG wird ebenso für die Kommunikation zwischen SAPGUI und Applikationsserver verwendet.

Beschreibung der HTML-Business Erweiterung erfolgt in Kapitel 5.5.2.

Service-Beschreibung

Zu jeder Web-Transaktion (siehe Kapitel 5.5.3 gehört eine Service-Beschreibung (spezielle Service-Beschreibung), die im Dateisystem des AGate-Servers abgelegt ist. Zusätzlich zu den Service-Beschreibungen der einzelnen Web-Transaktionen existiert eine globale Service-Beschreibung (*global.srvc*), die Standardparameter enthält. Fehlt in einer speziellen Service-Beschreibung ein Parameter, so wird dieser aus der globalen Service-Beschreibung übernommen. Wenn Parameter in der speziellen Service-Beschreibung enthalten und ebenso in der globalen Service-Beschreibung verfügbar sind, so überschreibt die spezielle Service-Beschreibung die Parameter der globalen Service-Beschreibung.

Folgende Parameter sind zum Beispiel in Service-Beschreibungen zu finden:

~appserver oder ~messageserver enthält den Namen des zu rufenden R/3 Systems. Entweder wird ein Applikationsserver direkt angesprochen oder der *Message-Server* leitet die Anfrage weiter.

~transaction enthält die zu startende R/3 Transaktion.

~login (optional) enthält den Benutzernamen für die Anmeldung am R/3 System.

~password (optional) enthält das zum Benutzernamen gehörige Paßwort für die Anmeldung. Das Paßwort kann mittels DES[29] verschlüsselt werden.

~client (optional) enthält den Mandanten für die Anmeldung.

~language (optional) enthält die Anmeldesprache. Der ITS nutzt diesen Parameter zusätzlich, um die sprachabhängigen Informationen zu erstellen.

~theme (optional) ermittelt die gewünschte Gestaltungsart.

~userTimeout (optional) bestimmt in Minuten den Zeitpunkt, wann eine inaktive Sitzung terminiert wird. Das manuelle Terminieren einer Sitzung ist über *~Command=Logoff*[30] möglich.

~ErrorMarker (optional) kann einen HTML Ausdruck enthalten, der beim Auffangen eines Fehlers mittels *assert* (siehe Kapitel 5.5.2), ausgegeben wird.

Die globale Service-Beschreibung eines Internet Transaction Servers ist in Abbildung 5.9 zu finden. Für eine Beschreibung der nicht erwähnten Parameter wird auf den Programmierleitfaden zu IACs [SAP99e] verwiesen.

[29] DES - Data Encryption Standard[PB96]

[30] Der Parameter *~Command* gehört nicht zu den Service-Beschreibungen, sondern wird über die URL an das WGate geleitet. Z.B.: http://www.myhost.com/scripts/wgate/?~Command=Logoff

```
~routestring
~messageserver
~logingroup
~systemname     NR1
~appserver      sap.sts.tu-harburg.de
~systemnumber   00
~client         002
~login          its
~password       des26(a34sdf6788sdi4d)
~language
~timeout        60
~usertimeout    60
~theme          99
~runtimemode    pm
~cookies        1
~urlarchive     /scripts/sapawl.dll
~urlimage       /sap/its/graphics
~urlmime        /sap/its/mimes
~exiturl        http://its.sts.tu-harburg.de
~hostunsecure   its.sts.tu-harburg.de
~portunsecure   80
~hostsecure     its.sts.tu-harburg.de
~portsecure     443
~xgateways      sapxgwfc
```

Abbildung 5.9: Globale Service-Beschreibung eines Internet Transaction Servers

Sprachabhängige Ressourcen

Die vom R/3 System an das AGate übertragenen Daten werden automatisch in der Anmeldesprache dargestellt. Um in den HTML-Templates je nach Anmeldesprache unterschiedliche Texte erzeugen zu können, werden auf dem AGate sprachabhängige Dateien abgelegt (*language resource files*). Diese Dateien enthalten zu einem *language resource key* einen entsprechenden Text, der vom HTML-Business Interpreter zur Laufzeit in das HTML-Template eingefügt wird.

Themes

Die Bausteine Service-Beschreibungen und Sprachabhängige Ressourcen werden auf Dateien abgebildet. Ein *Theme* ist die Untergliederung eines *ITS-Services*. Hierbei haben alle *Themes* die selbe Funktionalität und unterscheiden sich nur in der grafischen Aufbereitung. Somit ergibt sich die Möglichkeit, die selbe Web-Transaktion für verschiedene Benutzer unterschiedlich in einem Webbrowser aufzubereiten. *Themes* werden allgemein durch zwei Zeichen identifiziert, wobei der *Theme* 99 in der globalen Service-Beschreibung standardmäßig eingetragen ist.

Die mögliche Trennung von WGate und AGate auf verschiedene Server hat zwei wesentliche Vorteile:

- Durch die Trennung können mehrere WGates mit einem AGate kommunizieren. Somit können mehrere HTTP-Server die Anfragen bearbeiten.
- Das Hinzufügen einer *Firewall* zwischen AGate und WGate erhöht die Sicherheit zwischen Internet und R/3 System. Nur dem AGate ist die Adresse des R/3 Systems und gegebenenfalls ein Benutzer mit zugehörigem Paßwort bekannt, so daß ein Eindringen auf dem WGate ein geringeres Risiko darstellt, als das Eindringen auf dem AGate.

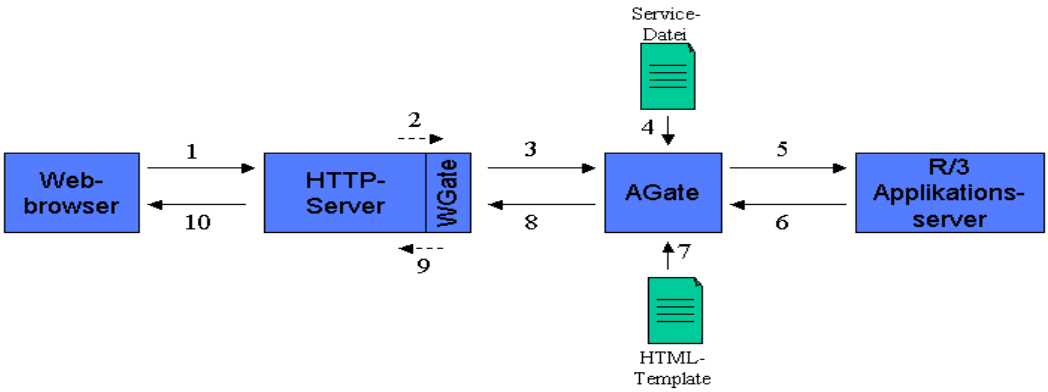

Abbildung 5.10: Ablaufdiagramm einer ITS-Sitzung

Wie eine Anfrage eines Webbrowsers an ein R/3 System über einen ITS aussieht, ist in Abbildung 5.10 am Beispiel einer *Web-Transaktion* zu finden. Hierbei werden die folgenden Phasen durchlaufen:

1. Ein Benutzer stellt mit Hilfe eines Webbrowsers eine Anfrage an eine bestimmte URL.
2. Der HTTP-Server erkennt anhand der URL, daß die Anfrage an das WGate weitergeleitet werden muß.
3. Das WGate bereitet die Anfrage auf und leitet sie an das AGate weiter.
4. Das AGate lädt zunächst die *Service-Datei* und entnimmt die nötigen Informationen.
5. Das AGate stellt die Anfrage an das R/3 System.
6. Das R/3 System antwortet an das AGate.
7. Das AGate liest das nötige HTML-Template und generiert mit der Antwort des R/3 Systems die HTML-Seite.
8. Das AGate sendet die HTML-Seite an das WGate.
9. Das WGate leitet die HTML-Seite an den HTTP-Server weiter.
10. Der HTTP-Server liefert die HTML-Seite an den rufenden Client aus.

5.5.2 Eine Erweiterung der HyperText Markup Language (HTML)

Die HyperText Markup Language (HTML)[Wor99] ermöglicht das Publizieren von Texten im *World Wide Web*. HTML nutzt zur Formatierung verschiedene Kommandos, um zum Beispiel Texte hervorzuheben oder Aufzählungen zu ermöglichen. Das Erstellen von HTML-Seiten, die kontextabhängig sind und dynamisch zur Laufzeit generiert werden, ist mit HTML nicht möglich. HTTP-Server stellen zur Generierung von solchen dynamischen HTML-Seiten verschiedene Möglichkeiten zur Verfügung. Fast jeder HTTP-Server verfügt über ein *Comman Gateway Interface (CGI)* [Gun96], um im Hintergrund zum Beispiel *Perl-Skripte* [SSP99] auszuführen und die Eingaben von Benutzern zu verarbeiten. Weitere Möglichkeiten sind die Nutzung von *JavaScript* [Fla97] oder Visual-Basic-Script (VB-Script) [Lom97], die direkt in die HTML-Seite eingebunden und vom Webbrowser interpretiert werden können.

Die SAP AG nutzt zwar die CGI-Technologie, um das WGate anzubinden, jedoch generiert das WGate keine HTML-Seiten sondern leitet diese nur an den HTTP-Server weiter beziehungsweise nimmt die Anfragen des HTTP-Servers an und übergibt diese an das AGate. Die Generierung der HTML-Seiten, die mit den Informationen des R/3 System versehen werden, läuft vollständig auf dem AGate ab. Hierbei nutzt das AGate die HTML-Business Erweiterung, um HTML-Seiten dynamisch zu generieren. Die HTML-Business Erweiterung [SAP99d] ist vergleichbar mit der STML [GW97] von Tycoon-2 [Wah98] oder den *Java Server Pages (JSPs)* [Sun99].

Um HTML-Business Ausdrücke in HTML einzubetten, muß entweder der Befehl `<server>` oder ein *Backquote* (‘) genutzt werden. Die *Backquotes* werden benötigt, da HTML das Verschachteln von Ausdrücken nicht erlaubt.

Die wichtigsten HTML-Business Erweiterungen sind:

- Das Referenzieren von einfachen R/3 Screen Felder, erfolgt über den Namen des zu referenzierenden Feldes. HTML-Business Ausdrücke werden, wenn auf dem R/3 Screen ein analoges Feld existiert, durch den Wert des Feldes ersetzt.
- Das Referenzieren von Tabellen und Step-Loops (siehe [Mat98]) erfolgt mithilfe des `array[j]` Kommandos.
- HTML-Business Ausdrücke sind analog zu JavaScript oder C. Gültige Ausdrücke sind zum Beispiel `1++`, `(1+1)*2` und `x==2 && y==4`.
- HTML-Business Funktionen generieren URLs und HTML Texte dynamisch. Es stehen die folgenden Funktionen zur Verfügung:

  assert ermöglicht das Auffangen von Fehlermeldung aus dem R/3 System. Mittels *assert* können die Fehlermeldungen des R/3 Systems durch individuelle Fehlermeldungen aus der Service-Beschreibung ersetzt werden.

  write gibt Feldinhalte aus, wobei alle Leerzeichen eliminiert werden.

  writeEnc gibt Feldinhalte aus, wobei der Feldinhalt gemäß URL Anforderungen kodiert wird. Sonderzeichen und Leerzeichen werden mit ihrem hexadezimalen Wert ausgegeben.

  wgateURL erzeugt dynamische URLs, die kompatibel zum HTTP-Server sind.

mimeURL erzeugt Verknüpfungen zu statischen Dateien auf dem HTTP-Server. Die ältere Version *imageURL* ist im ITS 2.2 noch verfügbar.

archiveURL greift auf das *iXOS Archiv* zu. Da der ITS ursprünglich von der Firma iXOS entwickelt wurde, ist dies eine Anbindung an das iXOS eigene Archiv-System.

- *If-Then-Else* Ausdrücke können mit der folgenden Syntax genutzt werden:
  `if` (expression) statement { [`elsif` | `elseif`] (expression) statement }
  [`else` statement] `end`
- Repeat-Schleifen können mit der folgenden Syntax genutzt werden:
  `repeat` expression times statement `end`
- For-Schleifen können mit der folgenden Syntax genutzt werden:
  `for` (expression ; expression ; expression) statement `end`

5.5.3 Internet Anwendungen und Komponenten

Internert Application Components (IACs) sind R/3 Transaktionen, Funktionsbausteine und R/3 Reports, die mit Hilfe des ITS über einen Webbrowser genutzt werden können. Zu einem vollständigen IAC gehören des weiteren die korrespondierenden HTML-Templates und die Service-Beschreibungen.

Die Entwicklung von IACs teilt sich in den R/3 internen und externen Teil auf. Im R/3 System muß die Anwendungslogik vorhanden sein, um die Anfrage des HTTP-Servers zu verarbeiten. Außerhalb des R/3 Systems müssen die in Kapitel 5.5.1 beschriebenen Service-Beschreibungen angelegt und die HTML-Templates erstellt werden, aus denen das AGate die HTML-Seiten für den HTTP-Server erzeugt.

Die *webfähigen* Transaktionen beziehungsweise Anwendungen unterteilen sich in drei Klassen, die im Folgenden vorgestellt werden.

WebRFC - Web Remote Function Call

Die Grundlage für WebRFCs ist der *Remote Function Call (RFC)*, der in Kapitel 6.2.2 ausführlich behandelt wird. Bei der Programmierung von WebRFC wird das *WebRFC Gateway Interface* genutzt, um auf Daten des R/3 Systems zuzugreifen.

Zu jedem WebRFC gehört ein webfähiger Funktionsbaustein im R/3 System, der vom ITS gerufen wird. Allen webfähigen Funktionsbausteinen ist die folgende Schnittstelle gemein:

QUERY_STRING ist eine Tabelle, in der alle URL-Parameter abgelegt sind, die vom Webbrowser an den Funktionsbaustein übergeben werden.

HTML enthält die vom Funktionsbaustein erzeugte HTML-Seite, wobei diese R/3 intern in einer Tabelle abgelegt ist.

MIME enthält Daten in binärer Form, die wiederum R/3 intern in einer Tabelle abgelegt sind. Die beiden Tabellen *HTML* und *MIME* unterscheiden sich im R/3 System durch ihre

Struktur und die Datentypen der einzelnen Felder. Zur Zeit kann entweder eine *HTML* oder *MIME* Tabelle übergeben werden. Beides gleichzeitig ist nicht möglich.

CONTENT_TYPE spezifiziert den Inhalt der Tabellen *HTML* und *MIME*. Bei *HTML* ist der *CONTENT_TYPE* `text/html`. Der *CONTENT_TYPE* für die Tabelle *MIME* kann alle gültigen MIME-Typen annehmen (z.B.:`image/gif`).

CONTENT_LENGTH gibt die Länge der Daten in den zurückgegebenen Tabellen an. Dieser Wert muß für *MIME* Tabellen gesetzt werden.

RETURN_CODE erlaubt das Abbauen der Verbindung nach dem WebRFC. Der Wert 0 beendet die RFC-Verbindung nach dem Funktionsaufruf. Bei dem Wert 1 wird die RFC-Verbindung offengehalten. Der Wert 2 schließt die RFC-Verbindung und terminiert die ITS-Sitzung.

Die Tatsache, daß ABAP keine *Streams* unterstützt, erklärt, warum eigene Tabellen für *HTML* und *MIME* existieren. Eine Zeile eines HTML-Dokuments entspricht in ABAP einer Zeile der Tabelle *HTML*. Die Tabelle *QUERY_STRING* enthält in ihren Zeilen *Name-Wert* Paare, die iterativ in dem ABAP Programm abgearbeitet werden können.

Bei der Entwicklung von WebRFCs wird die HTML-Seite, wie bereits an den Parametern der Schnittstelle zu erkennen, im R/3 System generiert. Eine Beispielanwendung, die alle Debitoren mit einem gewählten Anfangsbuchstaben ausgibt, ist in Abbildung 5.11 dargestellt.

Um die Anwendung aufzurufen, wird der ITS-Service *webrfc* genutzt, der als Parameter den Funktionsbaustein *y_www1* aufruft.

```
http://its.sts.tu-harburg.de/scripts/wgate/webrfc/!?_function=y_www1&name=M
```

Die Entwicklung von WebRFCs erfolgt im allgemeinen anwendungsbezogen und ist vom Layout und bestimmten Texten unabhängig. Um eine WebRFC Anwendung auf ein Unternehmen anzupassen, um zum Beispiel das Firmenlogo einzubetten oder den Firmennamen im Text zu verwenden, stehen *Templates* zur Verfügung. Die Tabelle *WWWDATA* enthält HTML-Texte oder MIME-Objekte, die im Quellcode des WebRFC referenziert und mittels des Funktionsbausteins *WWW_HTML_MERGER* eingefügt werden können. Es stehen weitere Funktionsbausteine und Methoden zu Verfügung, die ausführlich in der Online-Dokumentation [SAP99f] beschrieben werden, um die WebRFC Anwendungen generischer zu gestalten.

Web-Reporting

Für den schnellen Zugriff auf SAP Standard Reports (siehe auch Kapitel 4.3) bietet sich das *Web-Reporting* an. *Web-Reports* sind ein Spezialfall von den beschriebenen *WebRFCs* und nutzen ebenso Funktionsbausteine des R/3 Systems, die über RFC angesprochen werden.

Der Vorteil von *Web-Reporting* gegenüber *WebRFCs* und *Web-Transaktionen* ist die schnelle Realisierung. Der Entwickler braucht keine Kenntnisse in ABAP oder HTML, mögliche Variationen im Layout sind jedoch nicht vorhanden. Ziel des *Web-Reporting* ist der schnelle Zugang zu Informationen des R/3 Systems für Kunden oder Mitarbeiter, die keinen SAPGUI

```
FUNCTION Y_WWW1.
*"----------------------------------------------------------------------
*"*"Lokale Schnittstelle:
*"       TABLES
*"              QUERY_STRING STRUCTURE  W3QUERY
*"              HTML STRUCTURE  W3HTML
*"              MIME STRUCTURE  W3MIME
*"       CHANGING
*"             VALUE(RETURN_CODE) LIKE  W3PARAM-RET_CODE
*"             VALUE(CONTENT_TYPE) LIKE  W3PARAM-CONT_TYPE
*"             VALUE(CONTENT_LENGTH) LIKE  W3PARAM-CONT_LEN
*"----------------------------------------------------------------------
  DATA: ITAB LIKE KNA1 OCCURS 10 WITH HEADER LINE.
  DATA: NAME(30) TYPE C.
  REFRESH HTML.
  REFRESH MIME.

  CONTENT_TYPE = 'text/html'.
  LOOP AT QUERY_STRING.
    IF QUERY_STRING-NAME = 'NAME'.
      CONCATENATE QUERY_STRING-VALUE '%' INTO NAME.
    ENDIF.
  ENDLOOP.
 CONCATENATE NAME '%' INTO NAME.

  SELECT KUNNR NAME1 INTO CORRESPONDING FIELDS OF TABLE ITAB
          FROM KNA1 WHERE NAME1 LIKE NAME.
  HTML = '<html>'. APPEND HTML.
  HTML = '<body>'. APPEND HTML.
  HTML = '<h1>Ausgabe der Debitoren</h1>'. APPEND HTML.
  HTML = '<hr>'. APPEND HTML.
  HTML = '<table border=1>'. APPEND HTML.
  HTML = '<tr>'. APPEND HTML.
  HTML = '<th>Kundennummer</th>'. APPEND HTML.
  HTML = '<th>Name</th>'. APPEND HTML.
  LOOP AT ITAB.
    HTML = '<tr>'. APPEND HTML.
    CONCATENATE '<td>' ITAB-KUNNR '</td>' INTO HTML. APPEND HTML.
    CONCATENATE '<td>' ITAB-NAME1 '</td>' INTO HTML. APPEND HTML.
    HTML = '</tr>'. APPEND HTML.
  ENDLOOP.
  HTML = '</table>'. APPEND HTML.
  HTML = '</body>'. APPEND HTML.
  HTML = '</html>'. APPEND HTML.
ENDFUNCTION.
```

Abbildung 5.11: ABAP-Quellcode eines WebRFCs

```
REPORT YREPORT1 .

DATA: ITAB LIKE KNA1 OCCURS 10 WITH HEADER LINE.

SELECT KUNNR NAME1 FROM KNA1 INTO TABLE ITAB.

LOOP AT ITAB.
WRITE: /, AT 05 ITAB-KUNNR, AT 30 ITAB-NAME1.

ENDLOOP.
```

Abbildung 5.12: ABAP-Quellcode eines Web-Reports

besitzen, aber Informationen in aktueller Form benötigen. Per Definition sind alle R/3 Reports webfähig, soweit sie nur Tabellen und Texte ohne Grafiken enthalten. Die Generierung eines *Web-Reports* erfolgt für den Benutzer über einen Mausklick im Browser auf einen Link, der dem ITS mitteilt, welcher Report angezeigt werden soll.

```
http://its.sts.tu-harburg.de/scripts/wgate/
  webrfc/!?_function=WWW_GET_REPORT&_report=yreport1
```

Der zu diesem Aufruf gehörende ABAP-Report, der alle Kundennummern und Namen von Debitoren ausgibt, ist in Abbildung 5.12 zu finden.

Auf R/3 Seite nimmt der Funktionsbaustein `WWW_DISPATCH_REQUEST` die Anfrage entgegen, extrahiert und interpretiert die Parameter *_function* und *_report*. Aus dem Parameter *_function* erkennt der Funktionsbaustein, daß die Anfrage für `WWW_GET_REPORT` bestimmt ist und ruft diesen. Der Funktionsbaustein `WWW_GET_REPORT` ruft den gewünschten Report. Die Ausgabe des Reports wird dann in umgekehrter Reihenfolge an den ITS zurückgegeben.

Die *Templates*, die von der SAP als Standard mit ausgeliefert werden, sind:

WEBREPORTING_REPORT wird von den Funktionsbausteinen `WWW_GET_NODE_LIST`, `WWW_GET_SELSCREEN` und `WWW_GET_REPORT` aufgerufen.

WEBREPORTING_SELSCREEN wird von dem Funktionsbaustein `WWW_GET_SELSCREEN` aufgerufen. Um interaktive Reports zu verarbeiten (die Eingaben von Benutzern erwarten), erzeugt dieser Report eine HTML-Seite mit den zu füllenden Feldern, um anschließend das Ergebnis mit den *Template* `WEBREPORTING_REPORT` zu erzeugen.

WEBREPORTING_TREELIST wird von dem Funktionsbaustein `WWW_GET_TREE_LIST` aufgerufen. Reportbäume erlauben die Auswahl aus mehreren zu generierenden Reports. `WEBREPORTING_TREELIST` erzeugt eine HTML-Seite, in der alle Reports als Baum dargestellt werden, die für diese Darstellung freigegeben sind.

WEBREPORTING_TREENODE wird von dem Funktionsbaustein `WWW_GET_TREE_NODE` aufgerufen und erzeugt wie `WEBREPORTING_TREELIST` einen Baum aller Reports, jedoch wird als Parameter der Knoten des Baumes festgelegt, so daß nur Teilbäume angezeigt werden.

Web-Transaktionen

Die vorgestellten Methoden, um R/3 Daten im Web aufzubereiten, erlauben zwar auch eine Interaktion mit einem Benutzer, jedoch ist zum Beispiel die Erstellung eines Online-Bestellsystems mit Hilfe von einfachen Reports nicht möglich. Eine Transaktion, die über mehrere Bildschirmbilder läuft, vom Suchen eines Artikels bis hin zu einer Bestellung, ist mit *WebRFC* und *Web-Reporting* nicht zu realisieren.

Hierzu dienen *Web-Transaktionen*, die eine Benutzersitzung über mehrere Bildschirmbilder erlauben und mit vollständigen ABAP Transaktionen zu vergleichen sind. Die Entwicklung von einer *Web-Transaktion* erfordert sowohl die Entwicklung eines ABAP-Programms, als auch die Erstellung der HTML-Templates (siehe Kapitel 5.5.1) und der Service-Beschreibungen (siehe Kapitel 5.5.1). Im Gegensatz zum *WebRFC* und dem *Web-Reporting* können bei *Web-Transaktionen* die von der SAP zur Verfügung gestellten HTML-Templates und Service-Beschreibungen nur begrenzt genutzt werden.

Zur Generierung der HTML-Templates stellt das *SAP@Web Studio* einen Generator zur Verfügung, der sich mit dem R/3 System verbindet und aus den einzelnen Bildschirmbildern der R/3 Transaktion einzelne HTML-Dateien aufbaut. Allerdings müssen einige Daten nachträglich per Hand gepflegt werden. Dies betrifft zum Beispiel Auswahl- und Ankreuzfelder. Obwohl in ABAP als Eingabefelder der Länge 1 erzeugt, müssen die Felder per Hand mit dem Wert "X" belegt werden[31]. Des weiteren können einige *Screen-Elemente* aus ABAP nicht nach HTML übertragen werden. Dies betrifft insbesondere Menüleisten und Funktionstastenbelegungen. Diese Bausteine werden vom *SAP@Web Studio* vollständig ignoriert. Zur Erstellung der Service-Beschreibung bietet wiederum das *SAP@Web Studio* einen Generator. Durch Eingabe der Parameter kann die Service-Beschreibung auch mit verschlüsseltem Paßwort erstellt werden.

Durch das Konzept der IACs öffnet sich das R/3 System für den Bereich des *E-Commerce*. In naher Zukunft sind viele Anwendungen auf Basis des *Internet Transaction Servers* zu erwarten. Derzeit ist es der einzige Ansatz der SAP AG, das R/3 System für einen Webbrowser zu öffnen, bei dem alle Prozesse für die Transaktion auf dem Server ablaufen. Standardmäßig werden nach der Installation bereits einige Anwendungen zur Verfügung gestellt, die auf die firmenspezifischen Gegebenheiten angepaßt werden können.

[31] Die beschriebene Situation betrifft das *SAP@Web Studio* bis Version 4.5B.154. In zukünftigen Versionen wird diese Einschränkung sicherlich entfallen

Kapitel 6

Kopplung

Betriebliche Standardsoftwaresysteme sind in ihrem Funktionsumfang auf die Erfüllung betriebswirtschaftlicher Aufgaben beschränkt, so daß in einem Unternehmen andere Softwaresysteme für nicht-betriebswirtschaftliche Aufgaben eingesetzt werden müssen. Eine Kopplung beider Systeme ist sinnvoll, wenn Daten des einen Systems im anderen System weiterverarbeitet werden oder beide Systeme auf einen gemeinsamen Datenbestand zugreifen müssen.

Ein Szenario für eine solche Konfiguration ist der Einsatz einer speziellen Lagerverwaltungssoftware. Diese Software steuert unter anderem elektronisch die Staplerterminals und optimiert die Verteilung der Güter, so daß die meistgefragten Güter am schnellsten erreichbar sind. Betriebliche Standardsoftwaresysteme behandeln zum einen alle Güter gleich und unterscheiden nicht nach der Anzahl der Aufträge. Zum anderen besteht keine Möglichkeit, mit der Standardsoftware eine elektronische Steuerung der Staplerterminals vorzunehmen.

In diesem Szenario werden Warenanforderungen in das Standardsoftwaresystem eingegeben. Das Standardsoftwaresystem kennt die Soll-Bestände und kommissioniert den Auftrag. Die Anfrage wird an die Spezialsoftware weitergeleitet, die das Warenpaket zusammenstellt und gegebenenfalls eine Fehlermeldung über falsche Warenbestände, auf Grund einer Differenz zwischen Ist- und Soll-Bestand, zurück an das Standardsoftwaresystem gibt.

Gründe für die Kopplung mehrerer Softwaresysteme sind:

- Ein bestehendes externes System soll über Schnittstellen an die betriebliche Standardsoftware gekoppelt werden.
- Die Funktionalitäten des betrieblichen Standardsoftwaresystems decken nicht die Anforderungen eines Unternehmens ab. Ein neues externes System muß die zusätzlichen Anforderungen auffangen und die beiden Systeme müssen über Schnittstellen gekoppelt werden.
- Mehrere betriebliche Standardsoftwaresysteme eines Herstellers müssen über Schnittstellen gekoppelt werden. Gründe für eine Verteilung sind beispielsweise die geografische Verteilung der Systeme durch die Organisationsstruktur oder die nicht ausreichende Performanz eines einzelnen Systems, um die Anforderungen abzudecken.
- Die Übernahme des Datenbestandes eines alten Systems muß vollzogen werden.

Standardsoftwaresysteme, wie das R/3 System, stellen Schnittstellen zur Lösung dieser Problematiken und somit zur Kopplung der Systeme bereit.

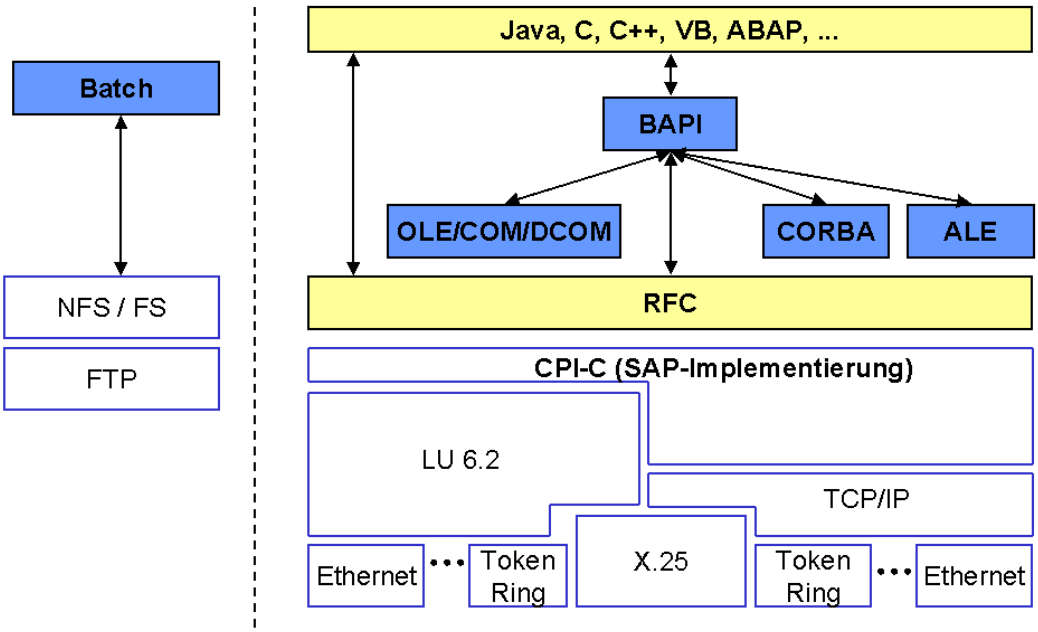

Abbildung 6.1: Programmierschnittstellen des R/3 Systems

In Abbildung 6.1 sind die verschiedenen Schnittstellen des R/3 Systems veranschaulicht. Mit Ausnahme des *Batch*-Verfahrens, das mit einer Datei-Schnittstelle arbeitet, setzen alle Kopplungstechniken auf dem Kommunikationsprotokoll *CPI-C* (siehe Kapitel 6.2.1) auf. Die einzelnen Schnittstellen werden in diesem Kapitel nacheinander vorgestellt und gegeneinander abgegrenzt.

6.1 Massendatenverarbeitung

Bei der Massendatenverarbeitung gilt es, große Datenmengen in eine Standardsoftware zu importieren oder zu exportieren. Die Verarbeitung der Daten erfolgt hierbei in oder aus einer Datei heraus. Insbesondere bei der Übernahme von Daten aus Altsystemen wird diese Art der Verarbeitung gewählt. Eine Kopplung zweier Systeme über eine Dateischnittstelle ist von Nutzen, wenn es keine Möglichkeit gibt, die Systeme über Programmierschnittstellen miteinander kommunizieren zu lassen.

Im R/3 System ist dies Verfahren unter dem Namen Batch-Input bekannt. Es ist die einfachste und schnellste Variante Daten in ein R/3 System zu übernehmen. Fast alle Programme bieten einen Dateiexport als Text- oder Binärdatei an, so daß der Export aus dem Altsystem möglich ist.

Für den Import bieten sich in R/3 Release 4.0B drei Methoden an, wobei nur zwei vorgestellt werden, da es sich bei der dritten um eine veraltete Methode (`CALL DIALOG`) handelt, die nur

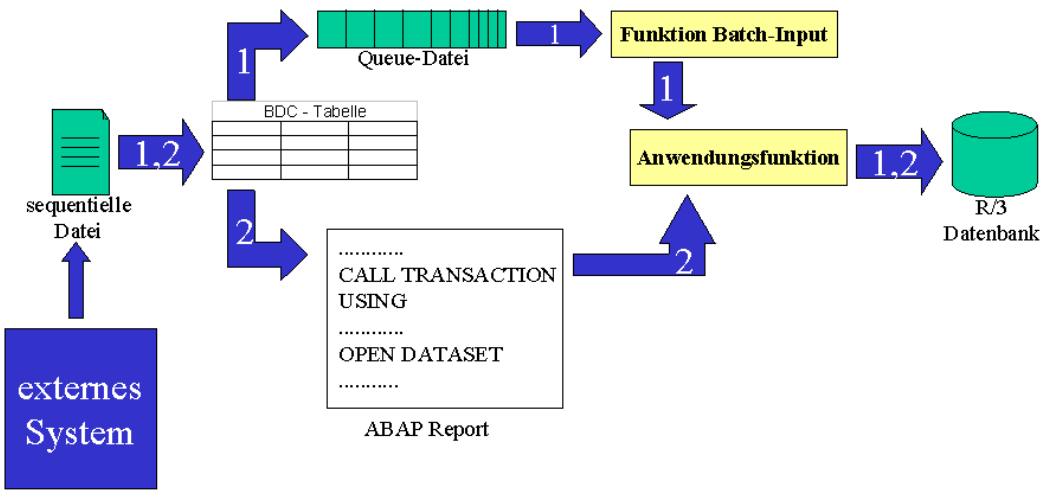

Abbildung 6.2: Batch-Input Verfahren

selten Anwendung findet.

Der Vorteil des *Batch-Input* liegt in der Konsistenzerhaltung der Datenbank. Da das Batch-Input Verfahren dieselben Dialogtransaktionen verwendet wie jeder R/3 Anwender, können keine Inkonsistenzen im System entstehen. Wenn allerdings im Customizing Fehler vorliegen, so wirken sich diese möglicherweise auch auf den Batch-Input aus. In Abbildung 6.2 sind beide Methoden des Batch-Input skizziert[1].

Bei dem Batch-Input Verfahren mit Mappe (Pfad 1 in Abbildung 6.2) muß zuvor eine Batch-Input-Mappe erstellt werden, die Stapel von Transaktionen enthält. Dieser Stapel mit Transaktionen wird, wenn die Mappe ausgeführt wird, *online* im System abgespielt. Dies bedeutet, daß jede Transaktion, die in der Mappe gespeichert ist, einzeln durchlaufen wird. Erst wenn ein Zyklus durchlaufen ist, wird der nächste Datensatz aus der Queue-Datei (vgl. Abbildung 6.2) geholt und der Stapel mit Transaktionen erneut abgearbeitet. Die wichtigsten Merkmale des Batch-Input Verfahrens mit Mappe sind:

- asynchrone Verarbeitung
- Übernahme von Daten mehrerer Transaktionen
- Protokollierung in dem Batch-Input-Verarbeitungsprotokoll
- keine parallele Verarbeitung von Mappen
- Verbuchung auf der Datenbank erfolgt sofort

Der Vorteil bei der Verarbeitung mit `CALL TRANSACTION USING`[2] (Pfad 2 in Abbildung 6.2)

[1] BDC - Batch Data Communication

[2] `CALL TRANSACTION USING` ist der entsprechende Funktionsaufruf in ABAP für dieses Batch-Input-Verfahren.

liegt in der Geschwindigkeit der Verarbeitung. Jedoch werden keine Protokolleinträge gepflegt und die mögliche Stapelverarbeitung entfällt ebenso.

Der ABAP Report verarbeitet bei dieser Methode die Daten während seines Programmablaufs. Hierzu ruft der ABAP Report die entsprechenden Transaktionen direkt auf und es werden keine Bildschirmbilder durchlaufen. Merkmale des zweiten Batch-Input Verfahrens sind:

- synchrone Verarbeitung (das Programm wartet bis die Ausführung beendet ist)
- *eine* Transaktion zur Datenübernahme
- synchrone und asynchrone Verbuchung auf der Datenbank möglich (Programmparameter)
- kein Batch-Input Verarbeitungsprotokoll

Beiden Methoden gemein ist, daß zunächst die Struktur der sequentiellen Datei umgewandelt werden muß, damit die Datei richtig interpretiert werden kann. Die Information über diese Struktur ist in der BDC-Tabelle hinterlegt (vgl. Abb. 6.2).

Soll eine ständige Kommunikation von R/3 mit Fremdsystemen realisiert werden, so ist zwar die Programmierung über Schnittstellen (siehe die folgenden Unterkapitel) komplizierter, doch gerade bei zeitkritischen Anwendungen ist Batch-Input nicht die richtige Alternative. Selbst ein *Scheduling* bzw. eine *Jobsteuerung* ist im R/3 System standardmäßig nicht integriert, sondern muß durch Programme wie *BatchMan* [Bau99] von Drittherstellern realisiert werden.

6.2 Programmierschnittstellen einer Standardsoftware

Eine direkte Kommunikation zwischen verschiedenen Systemen wird durch Massendatenverarbeitung, wie sie in Kapitel 6.1 beschrieben wurde, nicht ermöglicht. Soll eine direkte Kommunikation zwischen Softwaresystemen etabliert werden, so sind Programmierschnittstellen zu nutzen. Programme, die diese Schnittstellen nutzen, greifen auf standardisierte Bibliotheken einer Programmiersprache zu, um eine Verbindung aufzubauen.

Bei dem R/3 System existieren zwei Programmierschnittstellen, die zur Kopplung von Systemen genutzt werden können. In Kapitel 6.2.1 und Kapitel 6.2.2 werden die Programmierschnittstellen *Common Programming Interface for Communication (CPI-C)* und *Remote Function Call (RFC)* beschrieben.

6.2.1 CPI-C - Common Programming Interface for Communication

Die Programmierung mit CPI-C ist nur notwendig, wenn R/2 Systeme vor Release 5.0D angesprochen werden sollen oder bestehende Programme, die CPI-C nutzen, geändert werden. Ansonsten empfiehlt sich, eine der Schnittstellen, die oberhalb von CPI-C angesiedelt sind (siehe Abbildung 6.1), zu nutzen, da diese ausgereifte Funktionsbausteine und Mechanismen zur Fehlerbehandlung bieten.

CPI-C [IBM96] wurde 1987 von IBM im Rahmen der SAA-Norm[3] als standardisierte Kommunikationsschnittstelle für die Programm-Programm-Kommunikation definiert. Die Implementierung von CPI-C in R/3 ist an den X/Open Standard von CPI-C angelehnt und teilt sich in die Funktionsgruppen *CPI-C Starter Set* und *Advanced Function Calls* auf.

Ursprünglich wurde CPI-C für die IBM Mainframes entwickelt und agierte über LU6.2. Die Umsetzung von CPI-C auf TCP/IP erlaubte nur einen Teil des Interfaces zu übernehmen, da nicht alle Funktionalitäten von LU6.2 in TCP/IP zu finden sind.

CPI-C arbeitet im Halbduplexbetrieb und ist bereits im R/2 System implementiert. Es eignet sich somit unter anderem zur Kommunikation zwischen R/3 und R/2 Systemen. Des weiteren kann die Kommunikation zwischen zwei R/3 Systemen oder einem R/3 und einem externen System erfolgen.

Im R/3 System finden sich die folgenden Funktionen, wobei in Klammern die zugehörigen ABAP/4-Befehle zu finden sind:

- CMINIT (COMMUNICATION INIT) legt die Kommunikationsparameter fest, indem es die SIDEINFO-Datei[4] mit den Kommunikationsparametern auswertet.
- CMALLOC (COMMUNICATION ALLOC) veranlaßt das zu diesem Zeitpunkt aktive Programm, die Verbindung aufzubauen.
- CMACCP (COMMUNICATION ACCEPT) veranlaßt das zu diesem Zeitpunkt passive Programm, die Verbindung zu akzeptieren.
- CMSEND (COMMUNICATION SEND) sendet Dateien.
- CMRCV (COMMUNICATION RECEIVE) empfängt Dateien.
- CMDEAL (COMMUNICATION DEALLOCATE) beendet die Verbindung zwischen beiden Kommmunikationspartnern.

6.2.2 RFC - Remote Function Call

Der *Remote Function Call (RFC)* ist ein von der SAP AG definierter Standard[5], der die Kommunikation gegenüber CPI-C (vgl. Kapitel 6.2.1) vereinfachen soll. RFC setzt auf CPI-C auf und ermöglicht die Kommunikation zwischen folgenden Systemen:

- zwei unabhängigen SAP Systemen
- Client-Server Kommunikation, mit einem externen System als Client und einem R/3 System als Server
- Client-Server Kommunikation, mit einem R/3 System als Client und einem externen System als Server

[3] SAA - System-Application-Architecture

[4] Die SIDEINFO-Datei entspricht im R/3 System der Tabelle TXCOM, die über die Transaktion SM54 gepflegt wird.

[5] RFC ist in dieser Arbeit nicht mit dem weltweiten Standard RPC [Sun88] des Unix-Systems zu verwechseln, sondern ist Sprachgebrauch der SAP AG und dem R/3 Umfeld zu verstehen.

Die RFC-Schnittstelle übernimmt die Kommunikationssteuerung, die Parameterübergabe und die Fehlerbehandlung bei der Übertragung.

Mit Hilfe der RFC-Schnittstelle ist es möglich, Funktionsaufrufe auf entfernten Rechnern auszuführen, vergleichbar mit dem RPC[6] des UNIX-Systems [CDI98]. Im R/3 System sind die Prozeduren, die aufgerufen werden, sogenannte Funktionsbausteine, die in ABAP programmiert sind.

Das RFC-Schnittstellensystem untergliedert sich in die Aufrufschnittstelle für ABAP-Programme und für Nicht-SAP-Programme, die im Folgenden getrennt voneinander beschrieben werden.

Remote Function Call mit SAP-Programmen

Um Remote Function Calls aus einem R/3 System heraus durchführen zu können, muß zunächst im R/3 System eine *RFC-Destination* (Transaktion *SM59*) angelegt werden. Die wichtigsten Destinationen sind im Folgenden kurz beschrieben:

R/2 Verbindungen richten mit dem Host-Namen und ggf. Benutzerinformationen ein entferntes R/2 System als RFC Destination ein. Die Benutzerinformationen sind möglicherweise dem SAP-Gateway [CDI98] bekannt, so daß diese entfallen können.

R/3 Verbindungen benötigen als Parameter den Host-Namen und den Systemnamen zum Anlegen der Destination. Wenn der aktive User in beiden R/3 Systemen bekannt ist, so brauchen keine Benutzerinformationen eingegeben zu werden.

Logische Destinationen benutzen bekannte Destinationen (zum Beispiel eingerichtete R/3 Destinationen) und fügen eigene Informationen hinzu. Die zusätzlichen Informationen können zum Beispiel die Benutzerdaten oder der Mandant sein.

TCP/IP Verbindungen ermöglichen zwei Aktivierungsarten: Entweder das Starten oder das Registrieren von Programmen. Das Windowshilfe-System und Winword werden zum Beispiel über diese Verbindungsart aus dem R/3 System heraus gestartet. Programme können entweder von einem expliziten Host, dem Frontendserver oder einem Applikationsserver aus gestartet werden.

Die Namen der einzelnen Destinationen werden beim Aufruf von Funktionsbausteinen als Parameter mit übergeben. Ohne diese Destinationen lehnt R/3 die Kommunikation ab.

Beim RFC in ABAP-Programmen werden drei Arten von Aufrufen unterschieden, die durch ihre Eigenarten, sowohl für die Datenintegrität als auch für die Atomarität entscheidend sind.

synchroner RFC: Synchroner RFC verlangt, daß das entfernte R/3 System verfügbar ist. Ansonsten kehrt das Programm mit einer Fehlermeldung zurück.

Mit Hilfe des Zusatzes `DESTINATION` wird dem R/3 System bei einem `CALL FUNCTION` mitgeteilt, daß es sich um einen RFC Aufruf handelt. Ohne Zusätze handelt es sich

[6] RPC - Remote Procedure Call

automatisch um einen synchronen RFC. Ein kurzes Beispiel, welches aus der Destination `SUN14_1` alle Debitoren holt, ist in Abbildung 6.3 zu finden.

asynchroner RFC: Zum Zeitpunkt des Aufrufs eines RFC muß das entfernte R/3 System bei einem asynchronen RFC verfügbar sein. Im Gegensatz zum synchronen RFC kann jedoch mit der Verarbeitung fortgefahren werden, ohne das Ergebnis des RFC abwarten zu müssen.

Das Suffix `STARTING NEW TASK taskname` startet einen asynchronen RFC. *Import-*, *Export-* und *Exception-Parameter* sind zulässig. Um ein korrektes Ergebnis bei der Weiterverarbeitung zu erhalten, muß mittels des Kommandos `WAIT` auf die Rückgabe gewartet werden.

transaktionaler RFC: Beim transaktionalen RFC wird der gerufene Funktionsbaustein im R/3 System genau einmal ausgeführt. Des weiteren muß zum Zeitpunkt des RFC-Aufrufs das entfernte System nicht unbedingt verfügbar sein. Das R/3 System wiederholt den Aufruf in definierten Zeitabständen (parametrisierbar) bis zu einer maximalen Anzahl von Versuchen (parametrisierbar).

Der Aufruf eines Funktionsbausteins als transaktionaler RFC geschieht mittels des Suffixes `IN BACKGROUND TASK`. Beim transaktionalen RFC dürfen weder *Export-* noch *Exception-Parameter* an den Funktionsbaustein übergeben werden. *Import-Parameter* führen zwar zu keinem Fehler beim Übersetzen oder zur Laufzeit, jedoch liefern sie nicht das gewünschte Ergebnis. Der transaktionale RFC stellt die Basis für ALE (vgl. Kapitel 6.3) dar.

Remote Function Call mit Nicht-SAP-Programmen

Um auch aus Nicht-SAP-Programmen über *Remote Function Call* mit R/3 Systemen kommunizieren zu können, stellt das R/3 System Bibliotheken für unterschiedliche Programmiersprachen bereit. Bis zu R/3 Release 3.0 wurde nur eine C-Bibliothek zur Verfügung gestellt. Für R/3 Release 4.0 stehen auch C++- und Java-Bibliotheken zur Verfügung. Die einzelnen Bibliotheken sind bezüglich ihrer Mächtigkeit äquivalent, so daß in dieser Arbeit als Beispiel die Java-Bibliothek (die Pakete *com.sap.rfc* und *com.sap.rfc.exception*) behandelt wird.

Wie in Abbildung 6.4 zu sehen ist, wird der RFC aus Java nicht direkt an einen Applikationsserver des R/3 Systems weitergeleitet, sondern mittels CORBA über einen zusätzlichen Server weiterverarbeitet. Beim *Java Client* handelt es sich um eine Java Applikation oder ein Applet, das mittels der Packages *com.sap.rfc* einen RFC startet. Das anschließend erzeugte Proxy Objekt übersetzt die Informationen des RFC in eine portable CORBA-Präsentation [Obj99]. Mit Hilfe der CORBA Technologie wird die Verbindung zum ORB[7] des Server hergestellt. Auf der Serverseite wird der Java RFC in C++ umgesetzt und an einen Applikationsserver des R/3 Systems weitergeleitet.

Für einen *JRFC (Java RFC)* sind die folgende Schritte notwendig:

1. Zur Initialisierung des Java RFC wird ein *Factory Manager* Objekt erzeugt. Dieses Objekt ist der zentrale Kommunikationspunkt zwischen Client und Server. Zusätzlich

[7] ORB - Object Request Broker

```
REPORT ZAW43 .

DATA: ITAB LIKE BRFCKNA1 OCCURS 2 WITH HEADER LINE.

CALL FUNCTION 'RFC_CUSTOMER_GET'

 DESTINATION 'SUN14_1'
    EXPORTING
         KUNNR             = '*'
         NAME1             = '*'
     TABLES
          CUSTOMER_T        = ITAB
     EXCEPTIONS
          NOTHING_SPECIFIED = 1
          NO_RECORD_FOUND   = 2
          OTHERS            = 3.

CASE SY-SUBRC.
  WHEN 1.
    WRITE 'Nothing specified!'.
  WHEN 2.
    WRITE 'No record found!'.
  WHEN 3.
    WRITE 'Unknown exception!'.
  WHEN OTHERS.
    WRITE: AT 05 'Name', AT 40 'Ort'.
    WRITE: /, '------------------------------------------------------'.
    LOOP AT ITAB.
      WRITE: /, AT 05 ITAB-NAME1, AT 40 ITAB-ORT01.
    ENDLOOP.
ENDCASE.
```

Abbildung 6.3: Quellcode eines ABAP Reports - RFC zwischen zwei R/3 Systemen

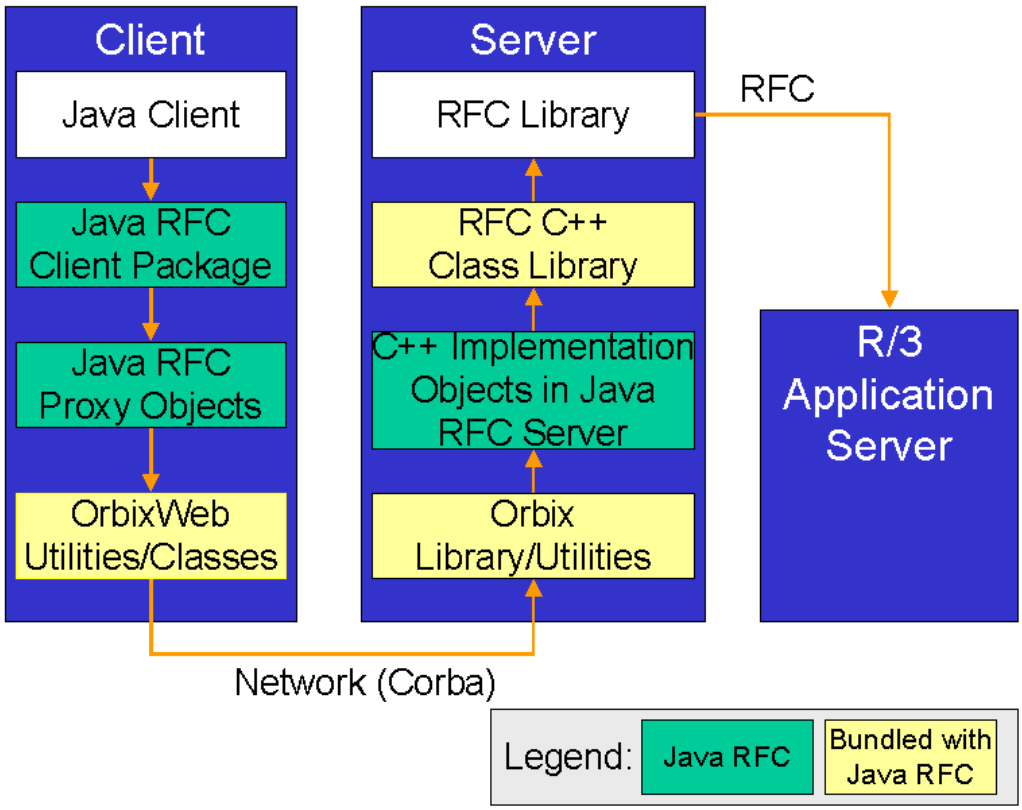

Abbildung 6.4: RFC über Java

benötigt der *Factory Manager* ein *MiddelwareInfo* Objekt, welches den RFC Server und den Typ des ORBs enthält. SAP liefert zum Zeitpunkt dieser Arbeit einen ORB der Firma Orbix aus, ermöglicht jedoch durch geeignete Schnittstellen den Einsatz anderer ORBs[8].

2. Zum Öffnen der RFC Verbindung werden zunächst zwei Objekte benötigt. Das erste ist ein *UserInfo* Objekt, welches die Anmeldeinformationen enthält, das zweite enthält die R/3 Systeminformationen, wie Hostname und System-Id. Mit Hilfe dieser beiden Objekte wird ein *Connection* Objekt der Klasse *IRfcConnection* erzeugt. Nach erfolgreichem Aufbau der Verbindung können über die Methode *getSystemInfo* die R/3 Systeminformationen abgerufen werden, die unter anderem den Versionsstand enthalten, da sich die RFC Schnittstellen zwischen den einzelnen R/3 Versionen unterscheiden.

3. Zum Erzeugen und Initialisieren des *Function Module* Objektes wird das Interface *IRfcModule* benötigt. Dieses Interface kapselt die RFC Parameter und wandelt diese in lineare Nachrichten (*Marshalling*) um.

[8]Der momentan verfügbare ORB, der von der SAP AG zur Verfügung gestellt wird, erfordert das Betriebssystem Windows NT4.0, so daß hier noch Weiterentwicklungen zu erwarten sind, die auch für andere Betriebssysteme Alternativen bieten.

4. Je nach aufgerufener RFC Methode sind bestimmte Parameter zu setzen, die an das R/3 System übergeben werden. Hierbei kann es sich sowohl um einfache Parameter handeln, die als String übergeben werden, als auch um Strukturen beziehungsweise ganze Tabellen. Je nach Parameter stehen hier eigene Methoden und Klassen zur Verfügung.

5. Zum Ausführen wird dem Objekt, das das Interface *IRFCModule* implementiert, die Nachricht *callReceive* geschickt.

6. Zur Verarbeitung der Rückgabe stehen, wie bei den an das R/3 System übergebenen Parametern, dieselben Methoden und Klassen zur Verfügung, um entweder einfache Parameter, Strukturen oder Tabellen zu verarbeiten (*Unmarshalling*).

Im Anhang A ist der Quellcode eines einfachen Java RFC zu finden, der über die Methode *RFC_CUSTOMER_GET* alle Kunden ausgibt. Zum Starten der Applikation muß der Interpreter mittels

```
java -Xbootclasspath:"C:\SAP\JRFC\Client\classes;
d:\jdk1.2\jre\lib\rt.jar" MyClient
```

gestartet werden, um den ORB der Firma Orbix zu aktivieren und zusätzlich die *Java Runtime Environment* zu setzen.

Die Kopplung von externen Systemen mittels JRFC ermöglicht den direkten Aufruf von Funktionsbausteinen im R/3 System. Diese Funktionsbausteine decken zwar grundlegende Funktionalitäten ab; um aber spezielle Anbindungen von externen Systemen zu ermöglichen, müssen einzelne Funktionsbausteine in ABAP selbst entwickelt werden.

Bei der Suche, welche RFC Bausteine bereits zur Verfügung stehen, hilft der *SAP Assistant*, der im Paket *SAP-Automation* enthalten ist, sich direkt mit dem R/3 System verbindet und Auskunft über die vorhandenen Funktionsbausteine und deren Parametrisierung gibt.

6.3 Applikations-Kopplung

Um betriebswirtschaftliche Anwendungen und zugehörige Daten zu dezentralisieren, wurden Methoden und Techniken zur losen Kopplung verschiedener Anwendungen entwickelt. Ziel ist es, einen konsistenten Datenbestand zu gewährleisten, ohne auf einen zentralen Datenbestand zugreifen zu müssen.

Im Gegensatz zum Batch-Input (siehe Kapitel 6.1), bei dem eine Menge von Daten ins R/3 System importiert, CPI-C, bei dem Datenströme ausgetauscht, und RFC, bei dem Funktionsaufrufe in entfernten Systemen durchgeführt werden, steht bei ALE mit IDocs der Austausch von Nachrichten zwischen lose gekoppelten Systemen im Vordergrund. Die SAP AG beschreibt die Einsatzmöglichkeiten wie folgt[SAP98d]:

> Application Link Enabling (ALE) ermöglicht die Verteilung betriebswirtschaftlicher Prozesse und Funktionen auf mehrere, lose gekoppelte R/3-Systeme. Die Anbindung von R/2- und Fremdsystemen wird unterstützt.

Hierzu bietet R/3 die Möglichkeit, über ALE einen konsistenten Datenbestand in verschiedenen R/3 Systemen zu erhalten. Hierbei ist allerdings darauf zu achten, daß auf den einzelnen R/3 Systemen verschiedene betriebswirtschaftliche Anwendungen durchgeführt werden. Würde auf zwei Systemen das Modul FI im Einsatz sein und möglicherweise gar die gleichen Buchungskreise und Mandanten bearbeitet werden, so ist die Konsistenz der Daten nicht mehr gewährleistet. Das von der SAP AG entwickelte Szenario könnte eine Trennung der Buchhaltung von der Lagerhaltung beinhalten. Rechnungen werden über ALE an die Buchhaltung weitergeleitet und Aufträge, die die Buchhaltung passiert haben, werden an die Lagerhaltung übergeben.

Bereits in der Einführung von R/3 ist ALE verankert, da bereits zu Beginn eines Einführungsprojektes die Verteilung der Anwendungen eine entscheidende Rolle spielt (siehe hierzu Kapitel 4.1).

In Abbildung 6.5 ist die Kommunikation zwischen zwei R/3 System zu sehen. Die eigentliche Kommunikation zwischen den Systemen wird über einen Applikationsserver und dessen Gateway-Prozeß abgewickelt (siehe Kapitel 3.1).

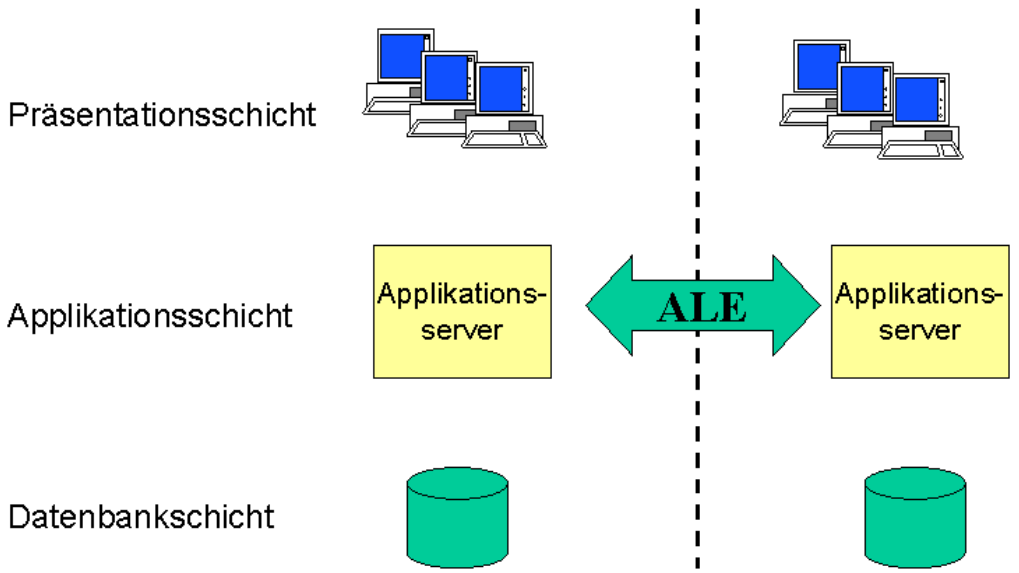

Abbildung 6.5: Kommunikation von zwei R/3 Systemen über ALE

Der Austausch von Nachrichten zur Verteilung von Anwendungsfunktionen erfolgt *asynchron* über die ALE Ausgangs- und Eingangsverarbeitung. Für diesen Austausch wird das sogenannte IDoc[9] (übersetzt: Zwischendokument) verwendet. Auch die Integration des Workflow-Systems (vgl. Kapitel 5.1) im Zusammenhang mit ALE kann den Nachrichtenaustausch überwachen und steuern.

Diese asynchrone Verbindung wird nur zum Austausch von Daten verwendet; es ist immer ein

[9]IDoc - Intermediate Document

schreibender Anteil vorhanden. Soll ausschließlich im entfernten System gelesen werden, so geschieht dies über einen synchronen Funktionsaufruf, der bereits im Kapitel 6.2.2 diskutiert wurde.

Im Folgenden wird der Aufbau von IDoc beschrieben und anschließend auf die ALE Ausgangs- und Eingangsverarbeitung eingegangen. Im Anschluß folgt ein kurzes Beispiel.

6.3.1 Aufbau von IDocs

Wenn in der Literatur von IDocs die Rede ist, so werden zwei Begriffe oft synonym verwendet. In dieser Arbeit soll zwischen dem IDoc-Typ und einem IDoc unterschieden werden. Ein IDoc leitet sich generell von einem IDoc-Typ ab, wobei der IDoc-Typ das Aussehen des IDocs beschreibt. Im R/3 System sind standardmäßig für die wichtigsten Geschäftsvorfälle bereits IDoc-Typen vorhanden. Sollen diese erweitert werden, so ist dies über die IDoc-Basisverwaltung möglich. Zur Erstellung von IDocs sei auf die SAP Online Hilfe [SAP98d] verwiesen.

Ein IDoc-Typ besteht aus einem Kontrollsatz, mehreren Daten- und Statussätzen. In allen drei Teilen ist eine eindeutige IDoc-ID vorhanden.

Der Kontrollsatz enthält Sender, Empfänger, Inhalt, Struktur, Status,etc. des IDocs. Der Status des IDocs ist zum Beispiel für die Versendung an R/3 Systeme unterschiedlicher Versionen wichtig, da je nach R/3 Version unterschiedliche IDoc-Typen existieren und gegebenenfalls auf ältere IDoc-Typen zurückgegriffen werden muß, um die Verarbeitung ohne Fehler durchzuführen.

Die Statussätze enthalten Information über die bisherigen Verarbeitungsschritte eines IDocs. Zum Beispiel `"Zu verarbeiten 18:01:03"` oder `"Erfolgreich verarbeitet 18:02:01"`.

Die Datensätze bestehen aus einem Verwaltungsteil und Segmenten. Der Verwaltungsteil gibt zum Beispiel Auskunft, welches zugehörige Segment das Elternsegment ist. Dies deutet bereits an, daß die Segmente in einer hierarchischen Beziehung zu einander stehen. Da die einzelnen Segmente in den unterschiedlichsten IDoc-Typen vorkommen können, ist diese Information in den Datensätzen gespeichert. Die Auskunft über einen Debitor kommt sowohl in einem Debitoren IDoc vor, als auch in einem Rechnungs IDoc.

Welcher Datensatztyp (bzw. welche Version) gesendet wird, entscheidet die ALE Schnittstelle selbstständig, da ihr die Version des Empfängers über die Angabe der *Destination* bekannt ist (vgl. Kapitel 6.2.2).

Die Segmente besitzen Attribute, Dokumentation und Segmentfelder, während die Segmentfelder wiederum Attribute, Dokumentation, Feldwerte und einen Datenteil besitzen. Die maximale Länge der Datenteile eines Segmentes ist eine 1000 Zeichen lange Feldleiste.

Die Attribute geben an, ob es sich um den Basistyp, ein muß oder optionales Segment handelt. Des weiteren werden bei Segmentfeldern die Daten des R/3 internen Repositories (vgl. Buck-Emden [BE99]) genutzt, um die interne Feldlänge und das dazugehörige Datenelement zu definieren. Die externe Feldlänge ist ebenso gespeichert. Die Feldwerte sind nur bei dem ersten Segmentfeld eines Segments relevant und geben die mögliche Operationen bei der

Verarbeitung der Segmentfelder an. Zulässige Operationen sind zum Beispiel Ändern, Löschen und Hinzufügen.

In Abbildung 6.6 ist ein Teil des IDoc-Typs *DEBMAS03* (Master IDoc für Debitoren zu R/3 Release 4.0) zu sehen. In der Abbildung wird der Aufbau eines IDoc-Typs als Baumstruktur deutlich.

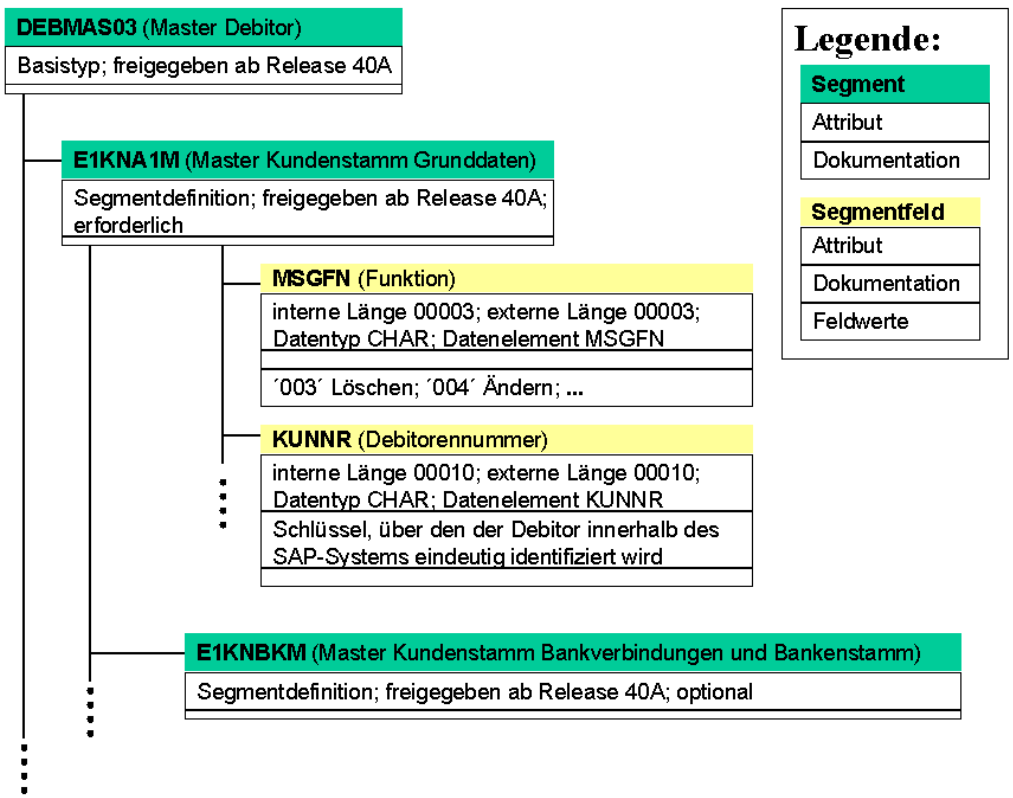

Abbildung 6.6: Aufbau eines IDoc-Typ (DEBMAS03)

6.3.2 ALE Ausgangsverarbeitung

In der Ausgangsverarbeitung baut die Anwendung durch einen Funktionsbaustein ein sogenanntes Master-IDoc auf und leitet dies an die ALE-Schicht weiter. Folgende Schritte werden durchlaufen, um das Kommunikations-IDoc, das anschließend übertragen wird, zu erstellen:

Die Empfängerermittlung liest aus dem IDoc, welches analog zu einem Brief einen Absender und Empfänger hat, den angegebenen Empfänger. Gegebenenfalls hat bereits die Anwendung alle nötigen Informationen in das IDoc eingetragen. Anderenfalls ermittelt die ALE-Schicht anhand des Verteilungsmodelles, das im Customizing gepflegt wird, die Empfängeradresse. Auch mehrere Empfänger können durch das Verteilungsmodell entstehen, so daß die Empfängerermittlung für jeden Empfänger ein eigenes IDoc erstellt.

In der Datenselektion werden anhand des Verteilungsmodelles die einzelnen IDocs mit Daten gefüllt.

Durch die Segmentfilterung können einzelne Segmente explizit aus einem IDoc entfernt werden. Durch die Einstellung von Filtern im Customizing können je nach Sender und Empfänger Filter hinterlegt werden.

Die Feldumsetzung entnimmt den im Customizing eingestellten Regeln die nötigen Operationen, um eine fehlerfreie Kommunikation zwischen den unterschiedlichen Systemen zu ermöglichen. Welche Felder umgestellt werden müssen, kann die ALE-Schicht entweder anhand des R/3 Releases ermitteln oder es müssen für Fremdsysteme explizite Eingaben gemacht werden.

Die Versionsumwandlung kann innerhalb eines IDocs zusätzlich Segmentfelder an Segmente anhängen oder ganze Segmente hinzufügen. Wie auch bei der Feldumsetzung wird hier die fehlerfreie Kommunikation zwischen unterschiedlichen Systemen gewährleistet. Im Anschluß an die Versionsumwandlung werden die IDocs in der Datenbank abgelegt.

Die Versendesteuerung unterteilt sich in die zeitliche Steuerung und die Mengensteuerung. Die zeitliche Steuerung terminiert, wann das IDoc übertragen werden soll. Die Mengensteuerung bündelt gegebenfalls mehrere IDocs zu einem größeren Auftrag und schickt diesen bei Erreichen eines im Customizing zu pflegenden Schwellwertes ab.

Bei einem Fehler in der Ausgangsverarbeitung wird auf das Workflow-System zugegriffen und der ALE-Administrator benachrichtigt.

6.3.3 ALE Eingangsverarbeitung

Erreicht ein IDoc die ALE-Schicht, so durchläuft das IDoc folgende Phasen:

Segmentfilterung - siehe Ausgangsverarbeitung -

Feldumsetzung - siehe Ausgangsverarbeitung -

Die Übergabesteuerung liest die IDocs von der Datenbank und leitet diese entweder sofort oder zeitgesteuert an die Anwendung weiter.

Die Verbuchung des eingegangenen IDocs kann auf drei Arten erfolgen: Die standardmäßig durchlaufende Methode ruft einen Funktionsbaustein auf, der das Verbuchen des IDocs übernimmt. Kommt es zu Fehlern, so wird ein Workflow gestartet. Des weiteren ist es möglich, entweder einen Workflow Prozeß zu starten oder ein einzelnes Workitem (siehe Kapitel 5.1) aufzurufen. Die letzten beiden Möglichkeiten sind nur durch Programmierung innerhalb der ALE-Schnittstelle [SAP98d] zu realisieren.

In Abbildung 6.7 ist der eben beschriebene Vorgang des Nachrichtenaustausches zwischen zwei Systemen über ALE grafisch dargestellt.

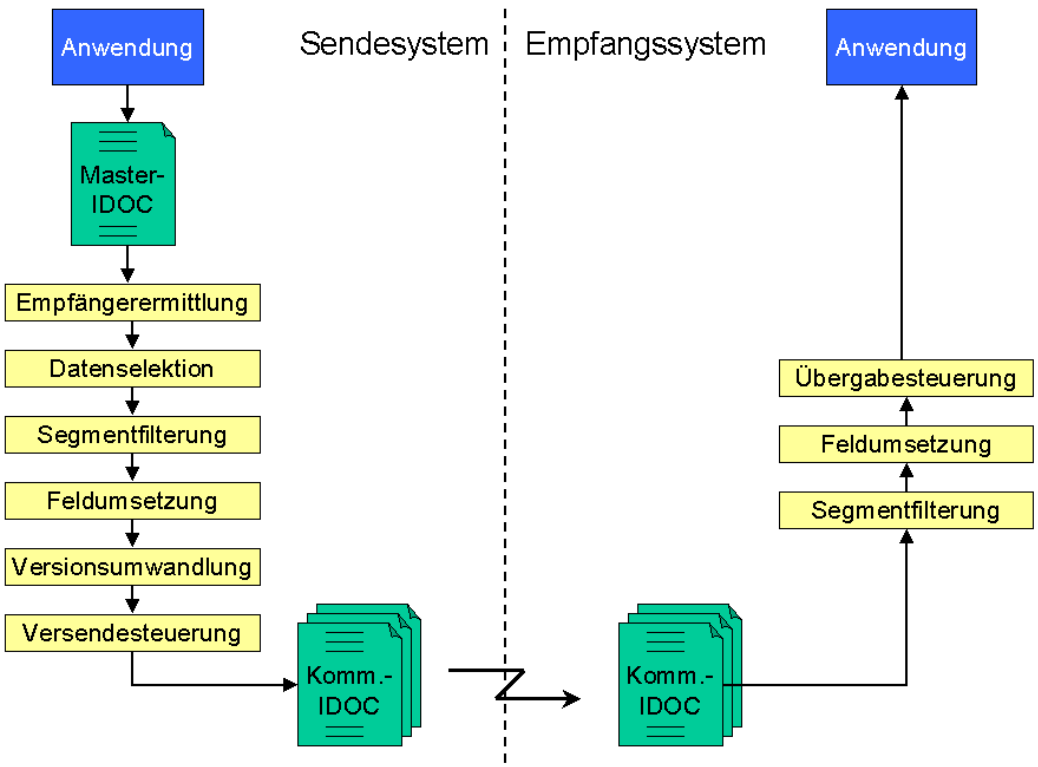

Abbildung 6.7: Nachrichtenaustausch über ALE

6.3.4 Erzeugen eines IDoc von DEBMAS03

Zunächst muß im Customizing ein logisches System angelegt werden. Hierbei handelt es sich nur um einen Tabelleneintrag, der bei der Partnervereinbarung wieder verwendet wird.

Anschließend muß im R/3 System eine Portbeschreibung angelegt werden. Hierbei kann es sich entweder um einen transaktionalen RFC (vgl. Kapitel 6.2.2), eine Datei, CPI-C (vgl. Kapitel 6.2.1) oder um das Internet handeln. Der Port Internet bedeutet an dieser Stelle die Übertragung eines IDoc als E-Mail Anhang im MIME-Format. Zur Erzeugung eines IDoc auf der Festplatte des Applikationsservers ist der Port *Datei* notwendig. Beim Anlegen eines Ports vom Typ *Datei* wird festgelegt, wo im Dateisystem die Kommando-, Ausgangs-, Eingangs- und Statusdateien abgelegt werden.

Nach dem Einrichten des Ports muß die Partnervereinbarung gepflegt werden. Hierbei wird zunächst die Partnernummer und die Partnerart benötigt. Diese beiden Parameter sind mit dem Customizing verknüpft und ermöglichen die Übernahme von Parametern, die bereits im Customizing gesetzt wurden. Soll es sich um eine reine Dateischnittstelle handeln, so wählt man als Partnerart das *Logische System* und gibt als Partnernummer das zuvor im Customizing angelegte logische System ein. Bei der Beschreibung der Partnerart ist der Empfängerport zu spezifizieren. Des weiteren kann der IDOC-Basistyp festgelegt werden, der in diesem Beispiel *DEBMAS03* ist.

Startet man nach dieser Konfiguration aus dem ALE Stammdaten Menü das Programm *Kunde senden* und gibt in der anschließenden Maske *DEBMAS* als Nachrichtentyp und das entsprechende logische System ein, so schreibt das R/3 System ein IDoc vom Typ *DEBMAS03* mit Kundendaten auf die Festplatte des Applikationsservers. Das erzeugte IDoc ist in Abbildung 6.8 zu sehen.

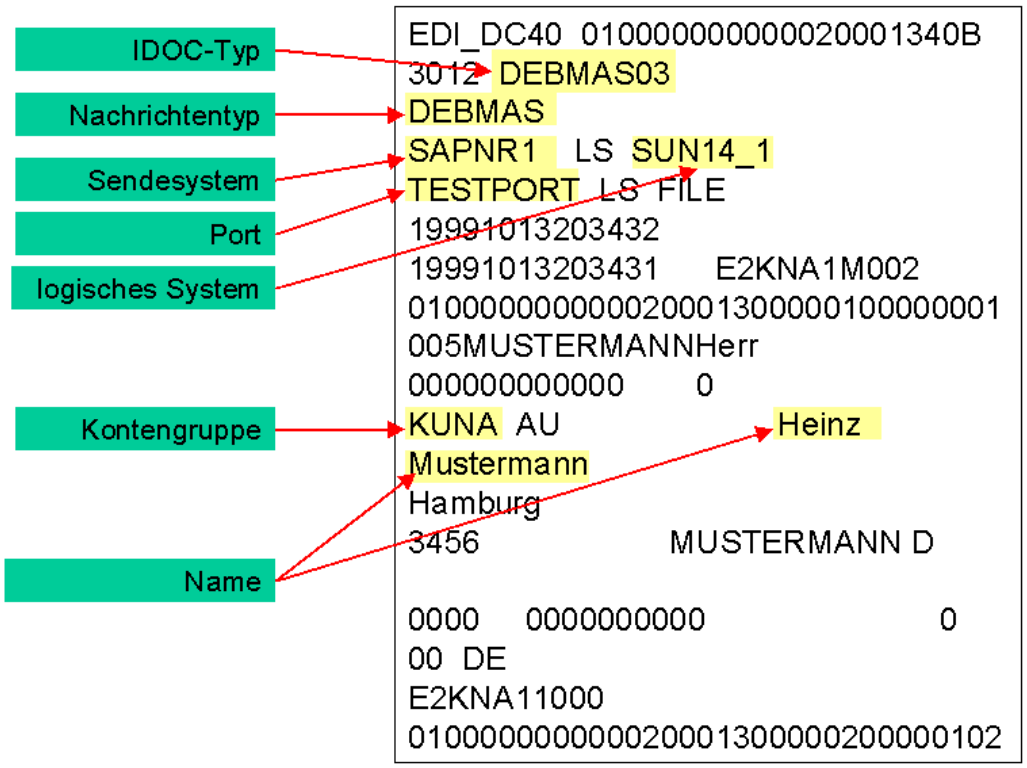

Abbildung 6.8: IDoc vom IDoc-Typ DEBMAS03

Die erzeugte Datei könnte nun per FTP[10] zu einem anderen System transportiert und anschließend importiert werden. Gerade diese Möglichkeit, ein IDoc als Datei zu erzeugen und weiterzuverarbeiten, erlaubt die Anbindung beliebiger Fremdsysteme, da die Struktur des IDocs durch seinen IDoc-Typ bekannt ist und im Fremdsystem gepflegt werden kann.

Auch die Kopplung von EDI-Systemen[11] ist über IDoc möglich, so daß zum Beispiel über ALE und IDoc die Anbindung an Banken geschaffen werden kann.

6.4 Interprogramm Kommunikation über Objektmodelle

Die in Kapitel 6.2 und Kapitel 6.3 vorgestellten Techniken nutzen teilweise objektorientierte Programmiersprachen zur Kommunikation mit betrieblichen Standardsoftwaresystemen,

[10] FTP - File Transfer Protocol
[11] EDI - Electronic Data Interchange

machen jedoch keinen Gebrauch von den Objekten, wie Kunden, Rechnungen, etc., die in der Standardsoftware enthalten sind. Diese Techniken nutzen objektorientierte Technologie nur als Mittel, um Funktionsaufrufe durchzuführen, anstatt Nachrichten an Objekten der Standardsoftware zu senden.

In diesem Abschnitt werden zwei Techniken vorgestellt, die mit Hilfe von objektorientierten Technologien auf Objekte eines Standardsoftwaresystems zugreifen und eine Interprogramm Kommunikation ermöglichen.

6.4.1 BAPI - Business Application Programming Interface

Durch den Ansatz des *Business Framework* will die SAP AG weg von der monolithischen Architektur des R/3 Systems und hin zu einer komponentenbasierten Architektur. Ziel ist es unter anderem, die Kopplung von verschiedenen Softwarekomponenten zu vereinfachen. Um diese Kopplung zu ermöglichen, wurden im R/3 System die SAP-Business-Objekte (vgl. Kapitel 5.1.3) erschaffen, die reale Objekte (z.B. Mitarbeiter, Kunden, Kundenaufträge, etc.) im R/3 System abbilden. Die Methoden dieser Business-Objekte sind die BAPIs[12]. Es kann zu einem Business-Objekt mehrere BAPIs geben. Es handelt sich somit bei BAPIs nicht um *eine* Schnittstelle des R/3 Systems, sondern um *eine* Schnittstelle *eines* Business-Objekts.

Die Einführung objektorientierter Technologien durch BAPIs und Business-Objekte öffnet das R/3 System nach außen für den Einsatz objektorientierter Methoden. Durch das Kapseln von Daten und Prozessen des R/3 Systems wird von den R/3 spezifischen Gegebenheiten abstrahiert.

Die ersten BAPIs wurden in R/3 Release 3.1 freigegeben. In R/3 Release 3.1 gab es ca. 150, in Release 4.5 sind es mittlerweile über 1000 Business-Objekte. Aus dieser Entwicklung ist die wachsende Bedeutung der Schnittstelle zu ersehen.

Die BAPIs sind im R/3 System als Funktionsbausteine integriert, die mittels des integrierten *Function Builders* entwickelt werden. Ein Funktionsbaustein wird wiederum einem BAPI zugeordnet und anschließend an ein Business-Objekt im BOR[13] gebunden. Alle diese Funktionsbausteine unterstützen den *Remote Function Call* (vgl. Kapitel 6.2.2).

Durch diese Trennung der Schnittstelle eines Business-Objektes und des dahinterliegenden Funktionsbausteins ist es der SAP möglich, Änderungen in den Funktionsbausteinen vorzunehmen, ohne die externen Anwendungen zu beeinträchtigen. Durch diese Transparenz können in neuen R/3 Versionen nötige Änderungen vorgenommen und mögliche neue optionale Parameter hinzugefügt werden; die bestehenden Anwendungen bleiben kompatibel.

Ist ein BAPI von der SAP freigegeben worden, so ist dies mindestens in den nächsten drei funktionalen Versionen weiterhin verfügbar. Sollten Änderungen notwendig sein, die nicht abwärtskompatibel sind, so erhält das neue BAPI den Namen des alten BAPIs mit einer Nummer hinter dem Namen. Das alte BAPI wird nun in der neuen Version als auslaufend (obsolet) gekennzeichnet und behält (s.o.) in zwei weiteren Versionen seine Gültigkeit.

Durch die beschriebene Trennung von *Implementation* und *Definition* von BAPIs ergeben sich

[12] BAPI - Business Application Programming Interface
[13] BOR - Business Object Repository

zwei Zugriffsmöglichkeiten. Die erste Möglichkeit besteht darin, direkt auf den Funktionsbaustein mittels RFC zuzugreifen. Diese Alternative ist bereits in Kapitel 6.2.2 beschrieben worden. Die zweite Möglichkeit ist der Methodenaufruf der Schnittstelle eines Business-Objektes, also einem *BAPI*. Dieser Zugriff, von der SAP als objektorientierter Zugriff [SAP98a] beschrieben, ist von verschiedenen Plattformen und mit unterschiedlichen Programmiermethoden möglich:

- Nutzung der BAPI C++ Klassenbibliothek
- Nutzung der BAPI Java Klassenbibliothek
- Zugriff über BAPI ActiveX Control (siehe Kapitel 6.4.2)
- Zugriff über COM/DCOM (siehe Kapitel 6.4.2)
- Nutzung von ObjectBridge von Visual Edge unter Verwendung von CORBA

Programmierumgebungen von IBM (Access Builder for SAP), Inprise (Delphi Connect) und Microsoft (Visual Studio) können als zusätzliche Umgebungen und Tools für die Anbindung genutzt werden.

Bis R/3 Version 4.0 wurden BAPIs immer synchron verarbeitet. Da jedoch eine synchrone Verbindung gerade in Lastzeiten zu Performanzeinbußen führt, ist die Empfehlung der SAP AG, bei häufig benutzten Schnittstellen mit großen Datenvolumen, eine asynchrone Kopplung zu benutzen. Hierzu wird die ALE Schnittstelle (vgl. Kapitel 6.3) mit IDocs genutzt. Auch bei einer ständigen Kopplung zweier Systeme soll auf die ALE und IDocs zurückgegriffen werden, damit bei einem Ausfall der Verbindung oder eines Systems, die Verfügbarkeit beider Systeme nicht beeinträchtigt wird. Zur Integration von BAPIs und ALE wird auf den Abschnitt ALE-Programmierung der SAP Online Hilfe [SAP98d] verwiesen.

Um den Kommunikationspartner zu ermitteln, wird auf das Verteilungsmodell (vgl. Kapitel 6.2.2 und Kapitel 6.3) zurückgegriffen. Bei synchronen Aufrufen wird der Empfänger anhand der *RFC Destinationen* ermittelt; bei asynchronen Aufrufen werden die *logischen Systeme* genutzt.

Eine weitere Neuerung in R/3 Version 4.0 ist die Änderung des Transaktionsmodells für BAPIs. Im R/3 System ist eine Transaktion gleichbedeutend mit einer Verabeitungseinheit oder auch *Logical Unit of Work (LUW)*. Um die Kombination mehrerer BAPI-Aufrufe in einer LUW zu ermöglichen, mußte der Zeitpunkt des *Commit of Work* geändert werden. Bis Version 4.0 riefen BAPIs selbst am Ende des Funktionsbausteins ein *Commit Work* auf. Seit Version 4.0 muß die Anwendung explizit ein Commit Work aufrufen, um die Verbuchung anzustoßen.

Im Folgenden wird die Nutzung von BAPI mit der BAPI Java Klassenbibliothek gezeigt. Beispielhaft werden die Daten eines Debitors aus dem R/3 System ausgelesen.

Als Generator für die notwendigen Klassen (z.B. jboDebtor, jboPurchaseOrder) wird der *SAP-Assistant* genutzt. Dieses Werkzeug verbindet sich mit dem R/3 System und listet alle zur Verfügung stehenden Business-Objekte und die zugehörigen BAPIs auf. Die für die eigene Applikation nötigen Objekte und ihre Methoden werden ausgewählt und anschließend erzeugt

der *Code-Generator* die entsprechenden Klassen. Der Quellcode dieser Klassen enthält Hinweise, wie ein Objekt angelegt werden kann. Zusätzlich gibt der *SAP-Assistant* Aufschluß über die Attribute eines Objektes und die Rückgaben einer Methode, um ein aufwendiges Suchen in den Quellcodes der Klassen zu vermeiden.

Der Aufruf des BAPIs (der Methode) „getDetail“ an einem Business Objektes der Klasse „jboDebtor“ erfordert die folgenden Schritte:

1. Der Kommunikationsaufbau erfolgt analog zu einem JRFC. Die Schritte 1 und 2 sind durchzuführen.
2. Das Objekt *myDebtor* wird mittels des Konstruktor der Klasse „jboDebtor“ erzeugt. Als Parameter ist die *DEBTORID* (eindeutiger Schlüssel des Debitors im R/3 System) anzugeben.
3. Die Methode „GetDetail“ wird mit dem Parameter „GetDetailParams“ aufgerufen. Das Objekt „GetDetailParams“ muß zuvor mit dem Buchungskreis (*Companycode*) erzeugt werden.
4. Durch den Aufruf der Methode „getReturn“ kann die Rückgabe, die in dem Objekt „GetDetailParams“ enthalten ist, des R/3 Systems ausgelesen werden.

Im Anhang B ist der Java Quellcode zu finden, der ein Objekt *Debtor* erzeugt und die Methode *getdetail* aufruft. Zum Starten der Applikation muß, wie bei einem JRFC (vgl. Kapitel 6.2.2), der Interpreter mittels

```
java -Xbootclasspath:"C:\SAP\JRFC\Client\classes;
d:\jdk1.2\jre\lib\rt.jar" MyClient
```

gestartet werden, um den ORB der Firma Orbix zu aktivieren und zusätzlich die *Java Runtime Environment* zu setzen.

Wie auch bei *RFCs* ist es möglich, eigene Funktionsbausteine zu entwickeln, um Funktionalitäten zu implementieren, die nicht im ausgelieferten Standard enthalten sind. Hierzu sei auf den BAPI-Programmierleitfaden [SAP98b] verwiesen.

6.4.2 Nutzung von OLE, ActiveX, COM und DCOM mit Standardsoftwaresystemen

Als Betriebssysteme für den Präsentationsserver von Standardsoftwaresystemen haben sich in den letzten Jahren die Betriebssysteme der Firma Microsoft durchgesetzt. Die Produkte *Windows 95*, *Windows 98* und *Windows NT* sind zum Zeitpunkt dieser Arbeit die meist genutzten Betriebssysteme der Firma Microsoft[14].

Zur Kommunikation zwischen verschiedenen Anwendungen in einer Windows-Umgebung wurden von der Firma Microsoft verschiedene Technologien entwickelt. Zu diesen Technologien

[14] Das Produkt *Windows 2000* wird in naher Zukunft ausgeliefert und voraussichtlich die anderen Produkte sukzessive verdrängen.

zählen *OLE*, *ActiveX*, *COM* und *DCOM*. Im Folgenden werden die einzelnen Begriffe gegeneinander abgegrenzt.

Object Linking and Embedding (OLE) wurde entwickelt, um zusammengesetzte Dokumente erstellen und bearbeiten zu können. Eine Integration von Excel-Tabellen in Word-Dokumente wurde beispielsweise durch OLE realisiert [Cha96]. Diese Technologie ist unter dem Namen *OLE 1* als erstes in die Windows-Systeme integriert worden. Aus *OLE 1* wurde die *COM-Technologie* entwickelt, die von Microsoft folgendermaßen beschrieben wird [Micb]:

> The Component Object Model (COM) is a software architecture that allows applications to be built from binary software components. COM is the underlying architecture that forms the foundation for higher-level software services, like those provided by OLE. OLE services span various aspects of commonly needed system functionality, including compound documents, custom controls, interapplication scripting, data transfer, and other software interactions.

Als Erweiterung zu *COM* wurde *DCOM* entwickelt, welches die Nutzung beziehungsweise Verteilung von *COM-Komponenten* über ein Netzwerk erlaubt. Microsoft beschreibt *DCOM* als [Micc]:

> The Distributed Component Object Model (DCOM) is a protocol that enables software components to communicate directly over a network in a reliable, secure, and efficient manner. Previously called "Network OLE," DCOM is designed for use across multiple network transports, including Internet protocols such as HTTP. DCOM is based on the Open Software Foundation's DCE-RPC spec and will work with both Java applets and ActiveX components through its use of the Component Object Model (COM).

ActiveX wurde erstmals 1996 in Verbindung mit Internet-Technologien als neue Technologie von Microsoft propagiert. Da die meisten Internet-Technologien jedoch mit der *COM-Technologie* zusammenarbeiteten, wurden Anwendungen, die *OLE* nutzen, im Folgenden als Anwendungen mit *ActiveX* tituliert [Mica]:

> ActiveX controls are among the many types of components that use COM technologies to provide interoperability with other types of COM components and services. ActiveX controls are the third version of OLE controls (OCX), providing a number of enhancements specifically designed to facilitate distribution of components over high-latency networks and to provide integration of controls into Web browsers. These enhancements include features such as incremental rendering and code signing, to allow users to identify the authors of controls before allowing them to execute.

Im R/3 System werden die OLE-Anwendungen[15] in der Tabelle *TOLE* registriert. Die Tabelle enthält neben den Namen der Anwendungen die OLE-Klassenidentifikation (CLSID), die in der Registrierungsdatenbank des Windows-System wiederzufinden ist. Weitere Felder

[15] Innerhalb des R/3 Systems wird der Begriff ActiveX (noch) nicht benutzt.

zur Behandlung und Identifikation von OLE-Anwendungen sind ebenso enthalten. Agiert der SAPGUI als OLE-Client, so werden aus dieser Tabelle die Parameter ausgelesen, um die Anwendung zu starten und die Interaktion zu beginnen. Soll der SAPGUI als OLE-Server agieren, so werden in der Registrierungsdatenbank die entsprechenden Einträge für die verschiedenen Komponenten eingefügt. Ein Beispiel für das Agieren des SAPGUIs als OLE-Server ist im Anhang C[16] zu finden. Es handelt sich hierbei um ein Visual Basic Makro, welches die Anmeldung am R/3 System durchführt und anschließend alle Debitoren des R/3 Systems in eine Excel-Tabelle überträgt. Das Auslesen der Debitoren erfolgt über einen *RFC* (siehe Kapitel 6.2.2) mit dem Funktionsbaustein `RFC_CUSTOMER_GET`. Die genutzte Komponente ist `SAP.Functions`, die das Objekt `Connection` und die Methode `RFC_CUSTOMER_GET` implementiert.

Die Nutzung von DCOM-Komponenten mit dem R/3 System ist über das *SAP Automation Paket* (vgl. Kapitel 6.2.2) möglich. In diesem Paket ist der *R/3 DCOM Connector* enthalten, der sowohl das Verwalten von R/3 Destinationen übernimmt als auch einen *Object Builder* enthält, der dynamisch Bibliotheken für die benötigten Business Objekte (vgl. Kapitel 5.1.3) oder die RFC-Funktionsbausteine erstellt. Der *Microsoft Transaction Server (MTS)* agiert als Anbieter für die Bibliotheken. Somit müssen die Bibliotheken nicht auf jedem einzelnen Personalcomputer installiert, sondern können zentral von dem MTS abgerufen und verwaltet werden.

Durch die Nutzung des *R/3 DCOM Connectors* und des *MTS* können Anwendungen für einen *PC Client* und für einen Webbrowser entwickelt werden. Die Entwicklung für einen Webbrowser muß jedoch mit *Active Server Pages (ASPs)* erfolgen, welches ein hohes Sicherheitsrisiko darstellt [Kos99].

[16] übernommen aus [CDI98]; gegenüber dem Original wurden einige Anpassungen und Fehlerberichtigungen vorgenommen

Kapitel 7

Zusammenfassung und Ausblick

Nach einer kurzen Zusammenfassung (vgl. Kapitel 7.1) werden die in dieser Arbeit vorgestellten Techniken und Konzepte zur Adaption von betrieblichen Standardsoftwaresystemen in einer Taxonomie klassifiziert und zusammengefaßt (vgl. Kapitel 7.2). In Kapitel 7.3 folgen abschließend eine Bewertung und ein Ausblick auf weitere Arbeiten und offene Punkte.

7.1 Zusammenfassung

Zu Beginn dieser Arbeit wird der Zusammenhang zwischen *Business (Re-)Engineering* und der *Einführung* von Standardsoftwaresystemen in Kapitel 2.1 untersucht. Standardsoftwaresysteme bieten allgemeingültige Geschäftsprozesse als Ausgangspunkt. Die spezifischen Geschäftsprozesse eines Unternehmens müssen auf die Verträglichkeit mit diesen allgemeingültigen Geschäftsprozessen hin untersucht werden. Die Einführung einer Standardsoftware geht somit mit dem *Business (Re-)Engineering* Hand in Hand. Standardsoftwaresysteme werden gewählt, um alte Geschäftsprozesse zu verwerfen und neue, optimierte Geschäftsprozesse in einem Unternehmen einzuführen (vgl. Kapitel 2.1.2). Als Methode zur Einführung eines Standardsoftwaresystems wird in Kapitel 2.3 *AcceleratedSAP* vorgestellt.

In Kapitel 3.1 wird zunächst die Architektur eines Standardsoftwaresystems beschrieben, um in Kapitel 3.2 Ansätze zur Adaption von Standardsoftware entlang der Systemarchitektur aufzuzeigen. Da eine Klassifikation entlang der Systemarchitektur verschiedene Methoden mischt und andere ähnliche Methoden in unterschiedliche Kategorien zusammenfaßt, wird in Kapitel 3.3 eine Klassifikation in drei Kategorien entwickelt:

Konfiguration: „Welche Einstellungen müssen gemacht werden?“

Erweiterung: „Welche Einstellungen können zusätzlich vorgenommen werden?“

Kopplung: „Wie können externe Systeme angebunden werden?“

Diese Klassifikation bildet die weitere Gliederung dieser Arbeit mit den gleichnamigen Kapiteln.

7.2 Taxonomie der Techniken und Konzepte

Die verschiedenen Techniken und Konzepte zur Adaption von betrieblicher Standardsoftware werden in den Kapiteln 4 bis 6 am Beispiel des R/3 Systems der SAP AG untersucht. In der Tabelle 7.1 sind die verschiedenen Techniken abschließend als Taxonomie zusammengefaßt. Die Bedeutung der Spalten ist wie folgt:

Anforderung enthält verallgemeinerte Anforderungen, die in einem betrieblichen Standardsoftwaresystem abgebildet werden sollen.

Verallg. Technik oder Konzept abstrahiert von einer bestimmten Standardsoftware und beschreibt abstrakt, welche Techniken oder Konzepte zur Adaption zur Verfügung stehen müssen.

Kategorie ordnet die verschiedenen Techniken und Konzepte in die in Kapitel 3.3 entwickelten Kategorien ein.

R/3 Technik bzw. Konzept beschreibt die Technik oder das Konzept des R/3 Systems, um die Anforderung in das R/3 System abzubilden.

Kapitel enthält den Verweis auf das entsprechende Kapitel dieser Arbeit.

7.3 Bewertung und Ausblick

Ziel dieser Arbeit ist es, die Techniken und Konzepte zur Adaption eines Standardsoftwaresystems zu untersuchen. Die wichtigsten Techniken werden am Beispiel des R/3 Systems beschrieben. Es ist jedoch im Rahmen einer solchen Arbeit nicht möglich, alle Techniken und Konzepte zu erfassen. Die mehrere tausend Seiten lange Dokumentation des R/3 Systems, welche vor dem Erscheinen als CD-ROM in „Metern“ gemessen wurde, gibt Aufschluß über die Komplexität eines solchen Softwaresystems. Inbesondere die einzelnen Module dieser Standardsoftware werden im Rahmen dieser Arbeit nicht genauer untersucht. Es wird ein Schwerpunkt auf die Basisfunktionalität und die sich daran angliedernden Techniken gelegt.

Als Abschluß sollen offene Punkte zusammengestellt und ein Ausblick auf mögliche zukünftige Arbeiten gegeben werden:

- Innerhalb dieser Arbeit wird das R/3 System der SAP AG mit dem *Release* 4.0B als Beispiel für ein betriebswirtschaftliches Standardsoftwaresystem genutzt. Ein Vergleich der Techniken und Konzepte zur Adaption von anderen Standardsoftwaresystemen (beispielsweise von *Oracle*, *Baan*, *Navision* oder *PeopleSoft*) kann die in Kapitel 7.2 ausgearbeitete Taxonomie erweitern und möglicherweise neue Kategorien aufdecken. Insbesondere der Vergleich der verschiedenen Techniken der unterschiedlichen Hersteller, um Adaptionen im Softwaresystem durchzuführen, kann Schwächen der verschiedenen Systeme aufdecken.
- Das R/3 System liegt kurz vor Abschluß dieser Arbeit im *Release* 4.6B vor. Insbesondere die Initiative *EnjoySAP* hat grundlegende Veränderungen des SAPGUIs zur Folge. Das

Anforderung	**Verallg. Technik oder Konzept**	**Kategorie**	**Technik bzw. Konzept des R/3 Systems**	**Kapitel**
Anpassung betriebswirtschaftlicher Strukturen	interaktives Vorgehen mit Tabellen, Masken, etc.	Konfiguration	Customizing	Kapitel 4.1
Dokumenten-Output-Steuerung	Textmanagement mit Formularen	Konfiguration	SAPscript	Kapitel 4.2
Anpassung des Berichtswesens	Datenselektion und -präsentation	Konfiguration	Reports	Kapitel 4.3
Abbilden von Benutzern, Rollen, Aufgaben	Benutzerverwaltung und Benutzerrechte	Konfiguration	Benutzerverwaltung und Benutzerrechte	Kapitel 4.4
Automatisieren von Arbeitsprozessen	Workflow-Management-Systeme	Erweiterung	SAP Business Workflow	Kapitel 5.1
Anpassen an unternehmensspezifische Gegebenheiten	integrierte Programmierumgebung, Schnittstellen	Erweiterung	Customer- und User-Exits	Kapitel 5.2
Automatisches Prüfen und Ersetzen von Benutzereingaben	Regeln zur Validierung und Substitution	Erweiterung	Validierung und Substitution	Kapitel 5.3
Anpassen der Benutzeroberfläche, Personalisierung	Ändern der Menüstruktur, Generierung von Favoritenlisten, etc.	Erweiterung	Personalisierung	Kapitel 5.4
Anbindung von Kunden über einen Webbrowser	Erstellung von Webseiten mit einer Kopplung zur Standardsoftware	Erweiterung / Kopplung	Internet Transaction Server (ITS) / RFC, BAPI	Kapitel 5.5 / Kapitel 6.2, Kapitel 6.4
Altdatenübernahme und Massendatenverarbeitung	Massendaten-Export und -Import mit sequentiellen Dateien oder elektronischen Formularen	Kopplung	Batch-Input, ALE-IDoc	Kapitel 6.1, Kapitel 6.3
Anbindung anderer Systeme	Schnittstellen des Applikationsservers	Kopplung	CPI-C, RFC, ALE-IDoc, BAPI, Batch-Input, (D)COM	Kapitel 6
Integration in die Windows-Oberfläche	Schnittstellen des Präsentationsservers	Kopplung	OLE, ActiveX (Microsoft) über den SAPGUI	Kapitel 6.4

Tabelle 7.1: Taxonomie der Techniken und Konzepte zur Adaption

Customizing-Menü[1] wurde überarbeitet und neue Funktionalitäten sind hinzugefügt worden. Ein Vergleich der unterschiedlichen *Releases* im Hinblick auf die Entwicklung der Techniken und Konzepte zur Adaption kann Aufschluß über die zeitliche Entwicklung von Adaptionstechniken geben.

- Die Initiative *mySAP.com* integriert die Geschäftsprozesse des R/3 Systems in das Internet. Hinter dieser Initiative verbirgt sich der Gedanke des *Outsourcings*, bei dem das R/3 System nicht wie bisher beim Kunden betrieben und administriert wird, sondern bei der SAP AG verbleibt. Der Kunde führt über das Internet die Transaktionen im R/3 System durch. Die Techniken und Konzepte der Adaption für den Kunden sind bei dieser Initiative anderen Rahmenbedingungen unterworfen. Eine neue Untersuchung und Entwicklung der Taxonomie für die Konzepte und Techniken zur Adaption in diesem Umfeld kann in einer zukünftigen Arbeit aufschlußreich sein.

[1] Customizing im Sinne der SAP AG

Anhang A

Quellcode eines Java RFC

```
import com.sap.rfc.*;
import com.sap.rfc.exception.*;

public class MyClient {
    public String rfcHost="sap.sts.tu-harburg.de";
    public String r3Host="orb.sts.tu-harburg.de";
    public String client="007", user="xxx", password="xxx", language="DE";
    public String custNo="*", custName="*";
    public int r3SysNo = 00;

    private FactoryManager facMan = null;
    private IRfcConnection connection = null;
    private IRfcModule module = null;
    private MiddlewareInfo mdInfo = null;

    public boolean connect() {
        //alte Verbindungen aufraeumen
        cleanUp();
        try {
            if( null == mdInfo ) {
                // MiddlewareInfo aufbauen
                mdInfo = new MiddlewareInfo();
            }
            mdInfo.setMiddlewareType(MiddlewareInfo.middlewareTypeOrbix);
            mdInfo.setOrbServerName(rfcHost);
            // Factory Manager erzeugen mit MiddlewareInfo
            facMan = FactoryManager.getSingleInstance();
            facMan.setMiddlewareInfo(mdInfo);
        }
        catch (JRfcBaseRuntimeException je) {
            displayMsg("Fehler im Factory Manager! Unerwarteter JRFC" +
                       "runtime Fehler:\n" + je.toString ());
            return false;
```

```
        }
        connection = logon();
        if( connection == null ) return false;
        // RFC mit RFC_CUSTOMER_GET
        try{
            module = facMan.getRfcModuleFactory()
                      .autoCreateRfcModule(connection, "RFC_CUSTOMER_GET");
        }
        catch (JRfcBaseException je) {
            displayMsg("Unerwarter JRFC Fehler beim Erzeugen des " +
                       "Modules:\n" + je.toString());
            return false;
        }
        displayMsg("R/3 Verbindung aufgebaut.");
        return true;
    }

    private IRfcConnection logon() {
        UserInfo userInfo = new UserInfo(user, password, client, language);
        ConnectInfo connectInfo = new ConnectInfo();
        IRfcConnection connection = null;
        connectInfo.setRfcMode(ConnectInfo.RFC_MODE_VERSION_3);
        connectInfo.setDestination("xxx");
        connectInfo.setHostName(r3Host);
        connectInfo.setSystemNo(r3SysNo);
        connectInfo.setLoadBalancing(false);
        connectInfo.setCheckAuthorization(true);
        try{
            connection = facMan.getRfcConnectionFactory()
                         .createRfcConnection(connectInfo, userInfo);
            connection.open();
        }
        catch (JRfcRfcConnectionException re){
            displayMsg("Unerwarterer RFC Fehler beim Oeffnen der " +
                       "Verbindung:\n" + re.toString());
            return null;
        }
        catch (JRfcBaseRuntimeException je) {
            displayMsg("Unerwarteter JRFC runtime Fehler beim Oeffnen " +
                       "der Verbindung:\n" + je.toString());
            return null;
        }
        return connection;
    }

    public void getData() {
        StringBuffer strbMessage = null;
```

```
    if( module == null ) {
        displayMsg("Verbindung zu R/3 nicht aufgebaut!");
        return;
    }
    SetParamValues();
    try {
        int rc = module.callReceive();
    }
    catch (JRfcRfcConnectionException re) {
        displayMsg("Unerwarter RFC Fehler beim Aufruf von " +
                   "RFC_CUSTOMER_GET:\n" + re.toString());
        return;
    }
    catch (JRfcBaseException je) {
        displayMsg("Unerwateter JRFC Fehler beim Aufruf von " +
                   "RFC_CUSTOMER_GET.\n" + je.toString());
        return;
    }
    GetParamValues();
}

private void SetParamValues() {
    try {
        module.getSimpleImportParam("KUNNR").setString(custNo);
        module.getSimpleImportParam("NAME1").setString(custName);
    }
    catch (JRfcBaseRuntimeException je) {
        displayMsg("Unerwarteter JRFC runtime Fehler beim Setzen " +
                   "der Parameter" + je.toString());
        return;
    }
}

private void GetParamValues() {
    displayMsg("Start: GetParamValues()\n");
    ITable theTable = null;
    try {
        theTable = module.getTableParam("CUSTOMER_T");
    }
    catch (JRfcBaseRuntimeException je) {
        displayMsg("Unerwarteter JRFC Fehler beim Erhalten der " +
                   "Tabelle:\n" + je.toString());
        return;
    }
    if( theTable == null ) {
        displayMsg("Rückgabetabelle ist leer!");
        return;
```

```
        }
        try {
            ComplexInfo complexInfo = theTable.getComplexInfo();
            int nFieldCount = complexInfo.getCount();
            //Spaltenüberschriften ausgeben
            for (int nColumn = 0; nColumn < nFieldCount; nColumn++) {
                SimpleInfo fieldInfo ;
                //Get individual field description
                fieldInfo = (SimpleInfo)complexInfo.getFieldInfo (nColumn);
                displayMsg(fieldInfo.getFieldName() + "\t");
            }
            displayMsg("\n");
            int nRowCount = theTable.getRowCount();
            for (int nRow = 0; nRow < nRowCount; nRow++) {
                IRow theRow = theTable.getRow(nRow);
                for (short nColumn = 0; nColumn < nFieldCount; nColumn++) {
                    displayMsg(theRow.getSimpleField(nColumn)
                    .getString() + "\t");
                }
                displayMsg("\n");
            }
        }
        catch (JRfcBaseException je) {
            displayMsg("Unerwarter JRFC Fehler bei der Rückgabe:\n" +
                       je.toString());
            return;
        }
    }

    public void cleanUp() {
        try {
            if( connection != null ) connection.close();

        }
        catch (JRfcBaseRuntimeException je) {
            displayMsg("Unerwarteter JRFC runtime Fehler beim Schliessen " +
                       "der Verbindung:" + je.toString());
            return;
        }
    }

    void displayMsg (String msg) {
        System.out.println(msg);
    }

    public static void main(String args[]) {
        System.out.println("Los geht's:\n");
```

```
        MyClient client = new MyClient();
        client.connect();
        client.getData();
    }
}
```

Anhang B

Quellcode eines BAPI Aufrufes unter Java

```
import com.sap.rfc.*;
import com.sap.rfc.exception.*;
import com.sap.bapi.*;
import com.sap.bapi.Debtor.*;

public class GetDebtorBapi {
    public String rfcHost="sap.sts.tu-harburg.de"
    public String r3Host="192.168.0.10";
    public String client="007", user="xxx", password="xxx", language="DE";
    public String custNo="*", custName="*";
    public int r3SysNo = 00;
    public jboDebtor myDebtor;

    private FactoryManager facMan = null;
    private IRfcConnection connection = null;
    private MiddlewareInfo mdInfo = null;

    public boolean connect() {
        //alte Verbindungen aufraeumen
        cleanUp();
        try {
            if( null == mdInfo ) {
                // MiddlewareInfo aufbauen
                mdInfo = new MiddlewareInfo();
                }
                mdInfo.setMiddlewareType(MiddlewareInfo.middlewareTypeOrbix);
                mdInfo.setOrbServerName(rfcHost);
                // Factory Manager erzeugen mit MiddlewareInfo
                facMan = FactoryManager.getSingleInstance();
                facMan.setMiddlewareInfo(mdInfo);
        }
```

```
        catch (JRfcBaseRuntimeException je) {
                displayMsg("Fehler im Factory Manager! Unerwarteter JRFC" +
                           "runtime Fehler:\n" + je.toString ());
                return false;
        }
        connection = logon();
        if( connection == null ) return false;
        displayMsg("R/3 Verbindung aufgebaut!\n");
        return true;
        }

    private IRfcConnection logon() {
        UserInfo userInfo = new UserInfo(user, password, client, language);
        ConnectInfo connectInfo = new ConnectInfo();
        IRfcConnection connection = null;
        connectInfo.setRfcMode(ConnectInfo.RFC_MODE_VERSION_3);
        connectInfo.setHostName(r3Host);
        connectInfo.setSystemNo(r3SysNo);
        connectInfo.setLoadBalancing(false);
        connectInfo.setCheckAuthorization(true);
        try{
            connection = facMan.getRfcConnectionFactory()
                          .createRfcConnection(connectInfo, userInfo);
                connection.open();
        }
        catch (JRfcRfcConnectionException re){
            displayMsg("Unerwarterer RFC Fehler beim Oeffnen der " +
                       "Verbindung:\n" + re.toString());
            return null;
        }
        catch (JRfcBaseRuntimeException je) {
            displayMsg("Unerwarteter JRFC runtime Fehler beim Oeffnen " +
                       "der Verbindung:\n" + je.toString());
            return null;
        }
        return connection;
    }

    public void cleanUp() {
        try {
            if( connection != null ) connection.close();
        }
        catch (JRfcBaseRuntimeException je) {
            displayMsg("Unerwarteter JRFC runtime Fehler beim Schliessen " +
                       "der Verbindung:" + je.toString());
                return;
        }
```

```
}

public void getData() {
    jboKey objectKey = jboDebtor.getEmptyObjectKey();
    objectKey.setKeyfield("DEBTORID","HEINZ");
    try {
            myDebtor = new jboDebtor(objectKey, connection);
    }
    catch(JRfcBaseException re) {
            System.out.println("Fehler beim Erzeugen des " +
                               "Debitor-Objektes.");
            return;
    }
    jboDebtorGetDetailParams getdetailParams =
                            new jboDebtorGetDetailParams();
    getdetailParams.setCompanycode("DE01");
    try {
            myDebtor.getdetail(getdetailParams);
    }
    catch(JRfcBaseException be) {
            System.out.println("Aufruf von jboDebtor.GetDetail" +
                               "fehlgeschlagen!");
            return;
    }
    if(getdetailParams.getReturn().getType().equals("S")) {
            displayMsg("Aufruf von Debtor.GetDetail " +
                       "erfolgreich!\n\n");
            displayMsg("KundenNr.: " + getdetailParams.getGeneraldetail()
                                       .getCustomer() + "\n");
            displayMsg("Name:      " + getdetailParams.getGeneraldetail()
                                       .getName() + "\n");
            displayMsg("Strasse:   " + getdetailParams.getGeneraldetail()
                                       .getStreet() + "\n");
            displayMsg("PLZ:       " + getdetailParams.getGeneraldetail()
                                       .getPostlCode() + "\n");
            displayMsg("Ort:       " + getdetailParams.getGeneraldetail()
                                       .getCity() + "\n");
    }
    else {
            displayMsg("Aufruf von Debtor.GetDetail " +
                       "fehlgeschlagen!\n\n");
            displayMsg("Rueckgabeparameter:\n");
            displayMsg("Type:      " + getdetailParams.getReturn()
                                       .getType() + "\n");
            displayMsg("Code:      " + getdetailParams.getReturn()
                                       .getCode() + "\n");
            displayMsg("Message:   " + getdetailParams.getReturn()
```

```
                                        .getMessage() + "\n");
                displayMsg("LogNo:     " + getdetailParams.getReturn()
                                        .getLogNo() + "\n");
                displayMsg("LogMsgNo:  " + getdetailParams.getReturn()
                                        .getLogMsgNo() + "\n");
                displayMsg("MessageV1: " + getdetailParams.getReturn()
                                        .getMessageV1() + "\n");
                displayMsg("MessageV2: " + getdetailParams.getReturn()
                                        .getMessageV2() + "\n");
                displayMsg("MessageV3: " + getdetailParams.getReturn()
                                        .getMessageV3() + "\n");
                displayMsg("MessageV4: " + getdetailParams.getReturn()
                                        .getMessageV4() + "\n");
        }
    }

    void displayMsg (String msg) {
        System.out.print(msg);
    }

    public static void main(String[] args) {
        GetDebtorBapi dummy = new GetDebtorBapi();
        if (dummy.connect()) {
        dummy.getData();
        }
        else {
             System.out.println("Programmabruch! Keine Verbindung!");
        }
    }
}
```

Anhang C

Quellcode eines RFC unter VisualBasic als Excel-Makro

```
Sub AlleDebitoren()

'Variablen Deklarationen
Dim oFunctionCtrl As Object
Dim oConnectiorn As Object
Dim Kunde, Kunden As Object
Dim Result As Boolean

' R/3 rufen und anmelden
Set oFunctionCtrl = CreateObject("SAP.Functions")
Set oConnection = oFunctionCtrl.Connection
If oConnection.Logon(0, False) = False Then
    MsgBox "R/3 Anmeldung fehlgeschlagen"
    Set oConnection = Nothing
    Set oFunctionCtrl = Nothing
    Exit Sub
End If

'RFC Aufruf
Result = oFunctionCtrl.RFC_CUSTOMER_GET(KUNNR:="*",
                        NAME1:="*", Customer_t:=Kunden)
If Result <> True Then
    MsgBox ("RFC fehlgeschlagen")
    oFunctionCtrl.Connection.logoff
    Set oConnection = Nothing
    Set oFunctionCtrl = Nothing
    Exit Sub
End If

' Excel-Tabelle füllen
i = 2
```

```
For Each Kunde In Kunden.Rows
    Cells(i, 1) = Kunde("KUNNR")
    Cells(i, 2) = Kunde("ANRED")
    Cells(i, 3) = Kunde("NAME1")
    Cells(i, 4) = Kunde("PFACH")
    Cells(i, 5) = Kunde("STRAS")
    Cells(i, 6) = Kunde("PSTLZ")
    Cells(i, 7) = Kunde("ORT01")
    Cells(i, 8) = Kunde("TELF1")
    Cells(i, 9) = Kunde("TELFX")
    i = i + 1
Next

'von R/3 Abmelden
oFunctionCtrl.Connection.logoff
Set oConnection = Nothing
Set oFunctionCtrl = Nothing

End Sub
```

Glossar

A

ABAP Advanced Business Application Programming. Die Programmiersprache des R/3 Systems.

Adaption Adaption bezeichnet das Anpassen, Erweitern, Konfigurieren, etc. von Standardsoftware.

AGate Application Gateway. Das AGate ist eine der beiden Komponenten des ITS. Das Application Gateway kommuniziert mit dem R/3 System und dem WGate.

ALE Application Link Enabling. ALE dient der Verteilung von Anwendungen. Die Kommunikation zwischen den Anwendungen kann über IDocs oder BAPIs erfolgen.

Applikationsserver In der dreistufigen Client/Server-Architektur ist der Applikationsserver die mittlere Schicht.

ASAP Accelerated SAP bzw. „As Soon As Possible “.

ASAP Roadmap Die ASAP Roadmap ist die „Landkarte“, in der die Phasen der R/3 Einführung mit ASAP zusammengefaßt sind.

Aufbauorganisation Die Aufbauorganisation ist eine Zusammenfassung von Informationen, die die strukturelle und personelle Umgebung des Unternehmens umfassend und dynamisch darstellen [SAP98d].

B

BAPI Business Application Programming Interface (BAPIs) sind Methoden der Business-Objekte.

Batch-Input Im R/3 System bezeichnet Batch-Input den Import/Export von großen Datenmengen über einen Hintergrundprozeß.

Buchungskreis Ein Buchungskreis ist die kleinste organisatorische Einheit des externen Rechnungswesens, für die eine vollständige, in sich abgeschlossene Buchhaltung, abgebildet werden kann [SAP98e].

Business-Objekt Repräsentant eines zentralen betriebswirtschaftlichen Objektes der realen Welt [SAP98e].

C

CCMS Das Computing Center Management System (CCMS) dient der Überwachung, Steuerung und Konfiguration des R/3 Systems.

COM Component Object Model. Ein von der Firma Microsoft definierter Standard.

CORBA Common Object Request Broker Architecture.

CPI-C Common Programming Interface for Communication. CPI-C wurde von IBM entwickelt und ist eines der von R/3 genutzten Protokolle.

Customer-Exit Vorgedachte Erweiterung des SAP-Standards.Potentielle Kundenanforderungen, die nicht als Standard ausgeprägt sind, werden als 'leere Modifikationshülse' im Standard vorgedacht und können beim Kunden mit kundenspezifischer Logik ausgestaltet werden [SAP98e].

Customizing Das Anpassen der unternehmensneutral ausgelieferten Funktionalität an die spezifischen betriebswirtschaftlichen Anforderungen des Kunden.

D

Datenbankserver Der Datenbankserver ist im R/3 System ein relationales Datenbankmanagementsystem. In der dreistufigen Client/Server-Architektur ist der Datenbankserver die unterste Schicht.

DCOM Distributed Component Object Modell. Ein von der Firma Microsoft definierter Standard. DCOM erweitert COM über Rechnergrenzen hinweg.

DDIC Data Dictionary (DDIC). DDIC ist ebenso einer der Standardbenutzer des R/3 Systems (siehe Kapitel 4.4).

DynPro Dynamisches Programm; - siehe Screen -.

E

Einzelschrittaufgabe Tätigkeit, die durch eine Objektmethode beschrieben wird. Einzelschrittaufgabe wird als Oberbegriff für Standardaufgabe und Kundenaufgabe verwendet. Eine Einzelschrittaufgabe ist im Gegensatz zu einer Mehrschrittaufgabe zu sehen, die durch eine Workflow-Definition beschrieben wird [SAP98e].

EPK Ereignisgesteuerte Prozeßketten (siehe Kapitel 2.1.2).

F

Form-Routinen Form-Routinen sind Unterprogramme in ABAP/4, die mit Prozeduren aus Pascal vergleichbar sind. Parameterübergabe via *Call-by-Reference* und *Call-by-Value* sind möglich. Die Rückgabe von Funktionswerten ist jedoch nicht möglich.

Funktionsbaustein Funktionsbausteine sind Unterprogramme, die von allen ABAP-Programmen aufgerufen werden können. Funktionsbausteine verfügen über definierte Schnittstellen.

G

Gesellschaft Eine Gesellschaft ist die kleinste organisatorische Einheit, für die nach der jeweiligen Handelsgesetzgebung ein Einzelabschluß zu erstellen ist. Eine Gesellschaft kann einen oder mehrere Buchungskreise umfassen [SAP98e].

H

Hot-Packages Die Hot-Packages (HP) der SAP AG sind Erweiterungen, Aktualisierungen und auch Fehlerkorrekturen der R/3 Software.

HTML-Business HTML-Business ist eine Erweiterung von HTML. Die HTML-Business Ausdrücke werden zur Laufzeit vom ITS durch HTML-Ausdrücke ersetzt.

HTML-Templates HTML-Templates werden vom ITS genutzt, um für IACs HTML-Seiten mit den entsprechenden R/3 Daten zu generieren. HTML-Templates enthalten eine Kombination aus HTML und HTML-Business.

I

IAC Internet Application Components (IACs) bestehen aus HTML-Templates, Service-Beschreibungen und einem R/3 Programm (Report, Transaktion, Funktionsbaustein, etc.).

IDoc Intermediate Documents (IDocs) sind das Standardformat der SAP AG zum Austausch von elektronischen Daten. IDoc ist das Pendant des R/3 Systems zu EDI. Ein IDoc ist die Ausprägung eines IDoc-Typs.

IDoc-Typ Ein IDoc-Typ bestimmt das Aussehen der abgeleiteten IDocs.

Implementation Assistant Der Implementation Assistant ist eines der Werkzeuge zur R/3 Einführung mit ASAP (siehe Kapitel 2.3).

Include Includes sind spezielle Unterprogramme von ABAP. Includes ermöglichen verschiedenen Anwendungen auf einen gemeinsamen Datenbestand zuzugreifen. Hierzu wird der Include in den einzelnen Programmen mit dem gleichnamigen Include-Befehl aufgerufen.

ITS Der Internet Transaction Server ermöglicht die Anbindung eines R/3 Systems an das World Wide Web. Der ITS bereitet Anfragen eines Webbrowsers auf und leitet diese an ein R/3 System weiter. In umgekehrter Richtung übernimmt der ITS die Aufbereitung der R/3 Daten für den HTTP-Server.

J

JRFC Java RFC. Nutzung von RFCs in Java.

L

Legal Changes Patches Die Legal Changes Patches (LCPs) aktualisieren die R/3 Software im Hinblick auf rechtliche Änderungen. LCPs beziehen sich zumeist auf die R/3 Komponente HR.

M

Mandant Definition im Glossar der SAP Online Hilfe: Eine für sich handelsrechtlich, organisatorisch und datentechnisch abgeschlossene Einheit innerhalb eines R/3-Systems mit getrennten Stammsätzen und einem eigenständigen Satz von Tabellen.
Anmerkung: Ein Mandant besitzt auf der Datenbank keinen eigenen Satz von Tabellen, jedoch arbeitet der Benutzer immer an einem Mandanten zur Zeit. Die Daten von anderen Mandanten sind zu diesem Zeitpunkt nicht zugreifbar.

Mehrschrittaufgabe Tätigkeit, die durch eine Workflow-Definition beschrieben wird. Mehrschrittaufgabe wird als Oberbegriff für Workflow-Muster und Workflow-Aufgabe verwendet. Eine Mehrschrittaufgabe beschreibt den formalen Rahmen (Schnittstelle, auslösende Ereignisse) für eine Workflow-Definition. Sie ist im Gegensatz zu einer Einzelschrittaufgabe zu sehen, die durch eine Objektmethode beschrieben wird [SAP98e].

Modulpool Modulpool ist eine Klasse von ABAP-Programmen. Ein Screen gehört zu einem Modulpool. Die Zusammenfassung von mehreren Screens in einem Modulpool ergibt eine ABAP-Anwendung.

N

Native SQL Native Structured Query Language. Native SQL ist einer der beiden SQL-Levels in ABAP und ist abhängig von dem verwendeten Datenbanksystem.

O

OLE Object Linking and Embedding. Ein von der Firma Microsoft definierter Standard.

Open SQL Open Structured Query Language. Open SQL ist einer der beiden SQL-Levels in ABAP und ist unabhängig von dem verwendeten Datenbanksystem.

ORB Object Request Broker.

P

Präsentationsserver Im R/3 System ist der SAPGUI der Präsentationsserver. In der dreistufigen Client/Server-Architektur ist der Präsentationsserver die oberste Schicht.

R

R/2 Realtime-System, Version 2. R/2 ist der Vorläufer des R/3 System und wurde für Mainframes entwickelt.

R/3 Realtime-System, Version 3.

Report Reports sind die kleinsten ABAP Programme. Reports dienen der Generierung von Berichten. Die Programmierung von interaktiven Reports ist möglich, jedoch werden im allgemeinen Screens benutzt.

RFC RFC steht für Remote Function Call und ist von der SAP AG definierter Standard. Der Remote Function erlaubt das entfernte Aufrufen von ABAP-Programmen und ist mit dem Remote Procedure Call (RPC) von UNIX zu vergleichen. Anmerkung: RFC ist in der Informatik ebenso wie RPC ein geläufiger Begriff. Im Rahmen dieser Arbeit soll RFC jedoch in dem von SAP erdachten Sinne gebraucht werden.

S

SAP Systeme, Anwendungen und Produkte in der Datenverarbeitung.

SAP Session Manager Der SAP Session ist ein Navigationswerkzeug des R/3 Systems. Er erlaubt das Starten von R/3 Funktionen, die baumartig im SAP Session Manager angezeigt werden. Zusätzlich wird durch eine Ablage dem Benutzer die Möglichkeit gegeben, wichtige R/3 Funktionen schnell zugreifbar zu machen.

SAPGUI SAP Graphical User Interface. Der SAPGUI ist die Oberfläche des R/3 Systems. Der Benutzer interagiert mit dem SAPGUI, der die einzelnen Bildschirmdialoge verarbeitet und mit dem R/3 System kommuniziert.

SAPNet Das SAPNet ist über http://sapnet.sap-ag.de zu erreichen und bietet die Möglichkeit nach Hot-Packages und Legal Changes Patches zu suchen. Installationhinweise und Fehlermeldungen sind ebenso zu finden.

SAPscript Die Programmierung von SAPscript dient der Erstellung von Texten für verschiedene Anwendung.

SAP@Web Studio Das SAP@Web Studio unterstützt den Programmierer bei der Entwicklung von IACs. Es generiert HTML-Templates und Service-Beschreibungen.

Screen Ein Screen ist ein ABAP Programm, das aus einem Bildschirmbild und der darunterliegenden Ablauflogik besteht. Im deutschen Sprachgebrauch wird DynPro synonym benutzt.

Service-Beschreibung Die Service-Beschreibung enthält Informationen, die zum Ablauf einer Web-Transaktion benötigt werden. Jede Web-Transaktion besitzt eine eigene Service-Beschreibung, in der unter anderem das zu rufende ABAP-Programm festgelegt ist. Eine globale Service-Beschreibung existiert zusätzlich auf jedem ITS und enthält zum Beispiel den Namen des R/3 Systems, das angesprochen wird.

SPAM Der SAP Package Manager (SPAM) dient dem Einspielen von Updates, wie Hot-Packages.

Standardsoftware Standardanwendungssoftware sind von Softwarehäusern und Hardwareherstellern für einen anonymen Markt entwickelte Programmsysteme zur Lösung von Anwendungsproblemen [Sch90].

Substitution Substitution ermöglicht das Ersetzen von Benutzereingaben. Benutzereingaben werden hierzu gegen boolesche Aussagen überprüft. Durch Substitution wird entweder die Benutzereingabe verändert oder die Eingaben unverändert auf die Datenbank geschrieben.

T

Taxonomie Lehre von der Klassifikation von Objekten [Sch97].

U

User-Exit Zeitpunkt in einem SAP-Programm, zu dem ein kundeneigenes Programm aufgerufen werden kann. Im Gegensatz zu Customer-Exits kann der Entwickler über User-Exits auf Programmteile und Datenobjekte des Standards zugreifen. Beim Upgrade muß jeweils geprüft werden, ob die Erweiterung durch einen User-Exit zum Standard paßt [SAP98e].

V

Validierung Validierung ermöglicht das Überprüfen von Benutzereingaben gegen boolesche Aussagen. Nach der Validierung werden Daten auf die Datenbank geschrieben und/oder Nachrichten ausgegeben.

W

Web-Reporting Das Konzept um R/3 Reports auf einem Webbrowser anzeigen zu können, heißt Web-Reporting. Hierbei werden vorgefertigte WebRFC genutzt, um Reports für den Webbrowser aufzubereiten.

Web-Transaktion Eine Web-Transaktion ist eine R/3 Transaktion, die mittels des ITS für einen Webbrowser aufbereitet wird. Siehe auch IAC.

WebRFC WebRFC nutzt den Remote Function Call (RFC), um HTML-Seiten mit dem ITS dynamisch aus Funktionsbausteinen zu genieren.

WGate Web Gateway. Das WGate ist eine der beiden Komponenten des ITS. Das WGate ist an einen HTTP-Server gebunden und kommuniziert mit dem AGate.

Work-Prozeß Work-Prozesse werden von SAP Dispatcher erzeugt. Dialog-Work-Prozesse bearbeiten zum Beispiel die Bildschirmdialoge.

Workflow-Manager Gesamtheit der vom SAP Business Workflow zur Laufzeit bereitgestellten Programme und Funktionen zur Steuerung und zur Ablaufkontrolle eines Workflows.

Workflow-Muster Von SAP ausgelieferte Mehrschrittaufgabe, die unverändert von Kunden verwendet werden darf. Workflow-Muster (WS) können nicht modifiziert werden und sind mandantenunabhängig, mit uneingeschränktem Gültigkeitszeitraum und planvariantenunabhängig [SAP98e].

Literaturverzeichnis

[Bal97] Balzert, Helmut: *Lehrbuch der Software-Technik.* Spektrum, 1997. – ISBN 3–8274–0089–9

[Bau99] Bauer Gruppe: *BatchMan - Job-Management unter SAP R/3 - Benutzerhandbuch.* 1999

[BE99] Buck-Emden, Rüdiger: *Die Technologie des SAP R/3 Systems.* vierte. Addison-Wesley, 1999. – ISBN 3–8273–1379–1

[BMS99] Berthold, Andreas ; Mende, Ulrich ; Schuster, Herbert: *SAP Business Workflow.* erste. Addison-Wesley, 1999. – ISBN 3–8273–1483–6

[Bra99] Brand, Hartwig: *SAP R/3 - Einführung mit ASAP.* erste. Addison-Wesley, 1999. – ISBN 3–8273–1380–5

[BRJ99] Booch, Grady ; Rumbaugh, James ; Jacobsen, Ivar: *The Unified Modeling Language User Guide.* erste. Addison-Wesley, 1999. – ISBN 0–201–57168–4

[CDI98] CDI: *SAP R/3 - Basissystem.* erste. Addison-Wesley, 1998. – ISBN 3–8273–1463–1

[Cha96] Chappell, David: *Understanding ActiveX and OLE.* Microsoft Press, 1996. – ISBN 1–57231–216–5

[CK99] Curran, Thomas A. ; Keller, Gerhard: *SAP R/3 Business Blueprint.* erste. Addison-Wesley, 1999. – ISBN 3–8273–1396–1

[CW85] Cardelli, Luca ; Wegner, Peter: On Understanding Types, Data Abstraction and Polymorphism. In: *Computing Surveys* 17 (1985), December, Nr. 4

[Fla97] Flanagan, Davis: *JavaScript, Das umfassende Referenzwerk.* OReilly, 1997. – ISBN 3–930673–56–8

[Gei95] Geiger, Kyle: *Insight ODBC - Entwicklerleitfaden zum Indeustriestandard für Datenbank-Schnittstellen.* Microsoft Press, 1995. – ISBN 3–86063–359–7

[Gun96] Gundavarm, Shishir: *CGI Programming on the World Wide Web.* OReilly, 1996. – ISBN 1–56592–168–2

[GW97] Gawecki, Andreas ; Wienberg, Axel: *STML Developer's Guide.* Burchardstraße 19, 20095 Hamburg, Germany: Higher-Order GmbH, 1997

[GW98] Gawecki, Andreas ; Wienberg, Axel: *Report on the Tycoon-2 Programming Language*. Burchardstraße 19, 20095 Hamburg, Germany: Higher-Order GmbH, 1998

[Ham90] Hammer, Michael: Reengineering Work: Don't Automate, Obliterate. In: *Harvard Business Review* (1990), Jul-Aug, S. 104–112

[HC95] Hammer, Michael ; Champy, James: *Reengeneering the Corporation - A Manifesto for Business Revolution*. Nicholas Brealy Publishing Ltd., 1995. – ISBN 1–85788–056–0

[IBM96] IBM: *The Official CPI-C Home Page*. 1996. URL `http://www.networking.ibm.com/app/aiwconf/cpic.htm`

[JBS97] Jablonski, Stefan ; Böhm, Markus ; Schulze, Wolfgang: *Workflow-Management. Entwicklung von Anwendungen und Systemen.* dpunkt-Verlag, 1997. – ISBN 3–920–99373–X

[JV87] Jessen, Eike ; Valk, Rüdiger: *Rechensysteme*. erste. Springer Verlag, 1987. – ISBN 3–540–16383–2

[KM97] Kristol, D. ; Montulli, L.: *HTTP State Menagement Mechanism - RFC 2109*. c/o Corporation for National Research Initiatives, 1895 Preston White Drive, Suite 100, Reston VA20191-5434, U.S.A.: Network Working Group - IETF, 1997. URL `http://www.ietf.org/rfc/rfc2109.txt`. Request for Comments: 2109

[Kos99] Kossel, Axel: Ein waches Auge - Kalkuliertes Risiko beim Internet-Zugang. In: *c't - Magazin für Computertechnik* (1999), Nr. 3

[Lom97] Lomax, Paul: *Learning VBScript*. OReilly, 1997. – ISBN 1–56592–247–6

[Mat98] Matzke, Bernd: *ABAP/4*. zweite. Addison-Wesley, 1998. – ISBN 3–8273–1372–4

[Mica] Microsoft Corporation: *ActiveX Controls*. URL `http://www.microsoft.com/com/tech/activex.asp`

[Micb] Microsoft Corporation: *The Component Object Model (COM)*. URL `http://www.microsoft.com/com/tech/com.asp`

[Micc] Microsoft Corporation: *The Distributed Component Object Model (DCOM)*. URL `http://www.microsoft.com/com/tech/dcom.asp`

[Mic98] Microsoft Corporation: *Microsoft Visual Basic 6.0 Programmers Guide*. 1998

[Obj99] Object Management Group (OMG): *The Common Object Request Broker: Architecture and Specifikation*. October 1999. URL `http://www.omg.org/corba`. V2.3.1

[PB96] Pfleger, Charles ; Becker, Paul: *Security in computing*. Prentice Hall, 1996. – ISBN 0–13–337486–6

[Por98] Porter, Michael E.: *Competitive Advantage: Creating and Sustaining Superior Performance*. Free Press, Juni 1998. – ISBN 0–6848–4146–0

[Pro99] PROJECT, Java A.: *JSSI*. 1901 Munsey Drive, Forest Hill, MD 21050-2747, U.S.A: The Apache Software Foundation, 1999. URL `http://java.apache.org`

[SAP98a] SAP AG: *BAPI-Benutzerhandbuch*. 1998

[SAP98b] SAP AG: *BAPI-Programmierleitfaden*. 1998

[SAP98c] SAP AG: *SAP Einführungsleitfaden*. Juni 1998

[SAP98d] SAP AG: *SAP Online Hilfe, Release 4.0B*. Juni 1998

[SAP98e] SAP AG: *SAP Online Hilfe, Release 4.0B - Glossar*. Juni 1998

[SAP99a] SAP AG: *AcceleratedSAP - Concept Check Tools*. April 1999

[SAP99b] SAP AG: *AcceleratedSAP - Implementation Assistant*. April 1999

[SAP99c] SAP AG: *AcceleratedSAP - Question and Answer Database*. April 1999

[SAP99d] SAP AG: *SAP Help - HTML-Business Language Description*. 1999

[SAP99e] SAP AG: *SAP Help - IAC Implementation Guide*. 1999

[SAP99f] SAP AG: *SAP Help - Programming WebRFCs*. 1999

[SAP00a] SAP AG: *SAP AG [AcceleratedSAP]*. 2000. URL `http://www.sap-ag.de/germany/products/imple/asap/`

[SAP00b] SAP AG: *SAP E-Business Solutions [AcceleratedSAP]*. 2000. URL `http://www.sap-ag.de/service/asapover.htm`

[Sch90] SCHEER, August-Wilhelm: *Wirtschaftsinformatik*. dritte. Springer-Verlag Berlin, Heidelberg, 1990

[Sch97] SCHNEIDER, Prof. Dr. Hans-Jochen (Hrsg.): *Lexikon Informatik und Datenverarbeitung*. vierte. R.Oldenbourg Verlag, 1997. – ISBN 3–486–22875–7

[SG96] SHAW, Mary ; GARLAN, David: *Software Architectures: perspectives on an emerging discipline*. Prentice Hall, 1996. – ISBN 0–13–182957–2

[SSP99] SIVER, Ellen ; SPAINHOUR, Stephen ; PATWARDHAN, Nathan: *Perl in a Nutshell*. OReilly, 1999. – ISBN 1–56592–286–7

[Sun88] Sun Microsystems Inc.: *RPC: Remote Procedure Call - RFC 1050*. 1988. URL `http://www.ietf.org/rfc/rfc1050.txt`. Request for Comments: 1050

[Sun99] Sun Microsystems Inc.: *JavaServer Pages Specifikation - Version 1.1*. November 1999

[Wah98] WAHLEN, Jens: *Entwurf einer objekt-orientierten Sprache mit statischer Typisierung unter Beachtung kommerzieller Relevanz*, Universität Hamburg, Fachbereich Informatik, Diplomarbeit, 1998

[Wor99] The World Wide Web Consortium: *HTML 4.01 Specification*. 1999. URL `http://www.w3c.org/TR/html401/`

Erklärung

Hiermit erkläre ich, daß ich die vorliegende Diplomarbeit selbständig durchgeführt und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Hamburg, den 9. März 2000

(André Wittenburg)