Vollständiger Abdruck der von dem Fachbereich Maschinenwesen
der Universität Duisburg - Essen
zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs

genehmigten Dissertation.

Thema:

Konzept zur Unternehmensplanung, -steuerung und –verwaltung durch objekt- und komponentenorientierte Entwicklung

Autor:

Jens Holl

Geburtsort: Wurzen

Universität Duisburg - Essen

Fachbereich 12 – Maschinenbau

Vorsitzender:	Prof. Dr.-Ing. H. Vinck
Prüfer der Dissertation:	Prof. Dr.-Ing. H.-J. Stracke
	Prof. Dr.-Ing. R. Koch

Tag der mündlichen Prüfung:

Mittwoch, 17.12.2003

1 EINLEITUNG

Der steigende Informationsumfang und der hohe Bedarf an Informationen beim Wertschöpfungsprozess in einem Unternehmen verlangen nach Maßnahmen zur Bewältigung dieser Herausforderungen. Die Zusammenhänge zwischen der Produktivität und den Informationen zu den Prozessen in einem Unternehmen nehmen ständig zu. Durch die Globalisierung und der damit verbundenen Zunahme der Konkurrenz sind die Unternehmen einem starken Innovations- und Kostendruck ausgesetzt. Um wettbewerbsfähig zu bleiben, sind die Unternehmen gezwungen, ihre betrieblichen Abläufe immer stärker datentechnisch zu begleiten, zu planen und zu steuern. Diese Aufgabe übernehmen Systemlösungen, wie PPS (Produktion, Planung und Steuerung), ERP (Enterprise Resource Planning) und PDM (Product Data Management) bzw. PLM (Product Lifecycle Management).

Da der starke Kostendruck im Gegensatz zum Innovationsbedarf steht, sind gerade kleine, mittelständische Unternehmen (< 100 Mitarbeiter), z.B. als Lohnfertiger im Sondermaschinenbau tätig, meistens nicht in der Lage, Investitionen für Systemlösungen oder sogar Systemkombinationen zu verkraften, obwohl diese die Grundvoraussetzung für Neuerungen bilden. Hinzu kommt, dass insbesondere mittelständische Unternehmen in der Regel über enorme dynamische, flexible und individuelle Geschäftsprozesse verfügen, die nur schwer mit konventionellen Lösungen abzubilden sind. Da aber nur eine Software effektiv genutzt werden kann, die auch das Unternehmen in allen Abläufen und Prozessen real widerspiegelt, werden viele dieser Unternehmen spezielle und individuelle Softwarelösungen benötigen.

Diese speziellen Lösungen können entweder mit einer konventionellen Software und einem enormen Anpassungsaufwand oder durch individuelle Lösungen bereitgestellt werden. Die Entscheidung, ob eine konventionelle oder eine individuelle Software zum Einsatz kommt, hängt vor allem von der Finanzkraft des Unternehmens, der zur Verfügung stehenden Implementierungszeit für die Software, der Investitionssicherheit und, nicht zuletzt, von der geforderten Funktionalität für das Systems ab.

1.1 MOTIVATION DIESER ARBEIT

Der hier formulierte Bedarf mittelständischer Unternehmen an einer durchgängigen Systemlösung für den gesamten Bereich der Produktion veranlasste den Verfasser gemeinsam mit einem mittelständischen Unternehmen eine eingehende Beurteilung der auf dem Markt angebotenen Systeme durchzuführen. Das hier beteiligte Unternehmen gilt nur als so genannter Stellvertreter für alle Branchen mit komplizierten Geschäftsabläufen und

deshalb wurde die Befragung so allgemeingültig wie möglich formuliert. Das Ergebnis des langwierigen Beurteilungsvorganges stellt sich wie folgt dar.

- Unvorhersehbare Kosten durch notwendige Anpassungen
- Kein konkreter Zeitrahmen für die Anpassungsdauer
- Hohe Anschaffungskosten für Hard- und Software
- Ungewissheit über Eignung der geforderten Funktionalitäten im Geschäftsalltag
- Hoher Schulungsaufwand
- Zweifel über Einfachheit der Bedienung
- Teilweise softwaretechnisch veraltete Systeme
- Hohe Support- und Updatekosten
- Deklarierte Kundenorientierung kann nicht realisiert werden

Diese Untersuchungsergebnisse verlangen eindeutig nach einer individuellen Software-Lösung. Hierbei ist sicherzustellen, dass einerseits vorhandene EDV[1]-Teillösungen soweit wie möglich integriert werden können und andererseits muss gewährleistet sein, dass der Fertigungsablauf auf das jeweilige Produkt abgestimmt werden kann, die Produkteigenschaften korrekt dem Kundenwunsch entsprechen und jeder Vorgang individuell geplant, gesteuert und abgewickelt wird.

Die vorliegende Arbeit zeigt ein Konzept, wie eine durchgehende Systemlösung individuell entwickelt werden kann, dabei auf aktuellen Technologien basiert, eine innovative Softwarearchitektur nutzt und darüber hinaus konsequent vorhandene Standards einhält.

1.2 Aufbau der Arbeit

Der erste Teil der Arbeit zeigt die aktuelle Situation in den Unternehmen und die an sie gerichtete Forderung softwaretechnisch zu modernisieren um wettbewerbsfähig zu bleiben. Die ermittelten Defizite einer umfassenden Analyse der vorhandenen konventionellen Systeme führen dann zu den Anforderungen für die Entwicklung eines spezifischen Software-Systems zur Abwicklung der unternehmensweiten Geschäftsprozesse für kleine und mittelständische Unternehmen.

Die anschließende Einführung in die neuen Entwicklungstechnologien und die relevanten softwaretechnischen Grundlagen schaffen die Voraussetzung für das technische Verständnis des nachfolgenden Konzeptes.

[1] Elektronische Datenverarbeitung

Folgend wird das Konzept auf der Basis einer objekt- und komponentenorientierten Software-Entwicklung hergeleitet und ausführlich beschrieben.

Die Machbarkeit des Konzeptes beweist sich an einem praktischen Beispiel eines mittelständischen Maschinenbau-Unternehmens, welches als Lohn- und Einzelfertiger tätig ist.

Nach einer kurzen Zusammenfassung befinden sich im Anhang Auszüge des Programmcodes, Teile von Datenbankanweisungen und diverse Flussdiagramme.

2 IST-SITUATION IN MITTELSTÄNDISCHEN UNTERNEHMEN

2.1 ALLGEMEINES

In der Industrie wird der geschäftliche Erfolg durch die im Unternehmen eingesetzten Computer-Anwendungen wesentlich beeinflusst. Um allen Anforderungen gerecht zu werden, ist es sicherlich notwendig, dass die Geschäftsprozesse (Auftragsabwicklung, Fertigungsplanung, etc.) mit den bestmöglichen Randbedingungen, wie Flexibilität, Praxisnähe, Zukunftssicherheit, schnelle und effiziente Informationshandhabung usw. optimal abgebildet werden.

Die datentechnische Infrastruktur der Unternehmen basiert heute auf Intra- und Internetlösungen, d.h. LANs[2] kommunizieren über das Internet; ein Beispiel hierfür ist das Virtual Private Network (VPN). Vor einigen Jahren waren solche Aussagen noch Visionen. Heute sind stetige Internet-Verbindungen und damit der zeitlich unbegrenzte Zugriff auf Firmendaten Standard. Bei der aktuellen Entwicklungsgeschwindigkeit sind auch die teilweise noch niedrigen Übertragungsraten zu vernachlässigen.

EDM[3] bzw. PDM[4] sind hier zwei der Begriffe, die heute und künftig das Umfeld in Unternehmen bestimmen. Sie beschreiben den Fluss und Status der unterschiedlich anfallenden Objekttypen (Artikel, Zeichnungen, Programme, Dokumente, Pläne, Prozesse, Kontakte etc.). Unterschiede zwischen EDM und PDM gibt es praktisch nicht, selten wird ein PDM-System an der Produktkonfiguration erkannt, die meisten EDM-Systeme besitzen diese Funktion ebenfalls, also sind die Unterschiede zwischen EDM und PDM, zumindest in dieser Arbeit, zu vernachlässigen. PDM-Systeme treten bei mittelständischen Unternehmen häufig in Konkurrenz zu PPS- / ERP-Systemen. Zum besseren Verständnis werden die Inhalte der hier genannten Systeme kurz definiert:

Definition und Funktion der EDM/PDM-Systeme:

Den folgenden Formulierungen liegen die Aussagen des Fraunhofer-Instituts für Arbeitswirtschaft und Organisation zu Grunde. PDM-Systeme sind das Fundament der technischen und administrativen Informationsverarbeitung. Sie stellen Schnittstellen zu CAD[5]- und PPS-Systemen und anderen Applikationen bereit. Weiterhin unterstützen sie einen

[2] Local Area Network

[3] Engineering Daten Management

[4] Product Daten Management

[5] Computer Aided Design (computerunterstütztes Konstruieren)

unternehmensweiten durchgängigen Informationsfluss. Fehlerquellen, wie inkonsistente und nicht aktuelle Informationsbestände, werden auf ein Minimum reduziert. PDM-Systeme verwalten technische Daten und Dokumente eines Produktes über dessen gesamten Lebenszyklus. Sie ermöglichen dem Anwender sich durch komplexe Produkt- und Dokumentenstrukturen zu navigieren, um die von ihm gewünschten Daten schnell und komfortabel zu finden und zur weiteren Verarbeitung zu nutzen. Für die Abbildung von Routineabläufen, wie sie im Freigabe- und Änderungswesen die Regel sind, stellen die meisten PDM-Systeme Workflowmechanismen zur Verfügung. Zur Sicherstellung des Zugriffsschutzes auf Daten ermöglichen sie die unternehmensspezifische Festlegung von Zugangsberechtigungen. Für die Belange des Qualitätsmanagements besteht die Möglichkeit, alle produktbezogenen Aktionen und Änderungen zu protokollieren und somit die Rückverfolgung mittels einer Produkthistorie zu gewährleisten. [01]

Diese Eigenschaften und Merkmale beschreiben Mindestanforderungen für PDM-Systeme, die heute auf dem freien Markt erhältlich sind. Es ist dennoch zu erkennen, dass PDM allein nicht einen kompletten Produktzyklus beschreiben kann. Es fehlen Komponenten, die als Teilmengen aber in PPS- und ERP-Systemen enthalten sind. Dazu zählen unter anderem Artikelbestände, Arbeitsvorbereitungstools, Vertriebsfunktionen oder Rückmeldungen aus der Produktion. Um eine durchgängige Systemlösung zu erhalten, müssen PPS und ERP mit PDM verknüpft werden. Der Aufwand und die Komplexität des Verbundes einer derartigen Systemlösung überfordert in der Regel jedes mittelständisches Unternehmen sowohl finanziell als auch in Bezug auf Bedienung, Pflege und Wartung des Systems.

Die Absichten der klein- und mittelständischen Industrie, ihre Produktion zu optimieren, d.h. Kosten zu senken, Durchlaufzeiten zu minimieren und flexibel auf Marktansprüche zu reagieren, wird durch die jeweils eingesetzte Software bestimmt. Möchte ein Unternehmen des Mittelstandes sein Datenmanagement automatisieren und durchgängig die EDV einsetzen, sind marktübliche PDM-, ERP- bzw. PPS-Systeme die Wahl.

Definition PPS/ERP:

Der Ausschuss für wirtschaftliche Fertigung (AWF) verwendet in seinen begrifflichen Definitionen für CIM-Bausteine (Computer Integrated Manufacturing) folgende Definition [02]: PPS bezeichnet den Einsatz rechnergestützter Systeme zur organisatorischen Planung, Steuerung und Überwachung der Produktionsabläufe von der Angebotserstellung bis zum Versand unter Mengen-, Termin- und Kapazitätsgesichtspunkten. ERP hingegen unterstützt sämtliche betriebswirtschaftliche Aufgaben eines Unternehmens. [03]

Die Anschaffung der erwähnten Systemprogramme ist nur mit einem enormen Kosten- und Zeitaufwand realisierbar. Die hierfür notwendigen Investitionsmittel übertreffen erfahrungsgemäß die Finanzkraft der mittelständischen Unternehmen und stellen somit ein

Handicap für Neuerungen dar. Konventionelle PPS-Systeme sind hochkomplex und für viele Unternehmen des Mittelstandes überdimensioniert. Die Vielfalt der Funktionalitäten ist unbestritten, muss jedoch vom Kunden mit bezahlt werden, ob sie nun erforderlich ist oder nicht. Ein weiteres Dilemma sind die Datenbank- und Anwendungsabhängigkeit. Konventionelle Systeme basieren auf einer bestimmten Datenbank und arbeiten mit integrierten Text- und Kalkulationsprogrammen, die zum System gehören, was wiederum bedeutet, dass modulare Erweiterungen von der gekauften Software abhängig sind.

Die Abhängigkeit einer Anwendung von einer bestimmten Datenbank ist jedoch nicht immer ein negativer Faktor, denn Datenbanken sind prädestiniert für die Ablage der Geschäftslogiken. Anwendungen, die auf diese Architektur bauen, besitzen logischerweise diese Einschränkung. Eine Schlussfolgerung daraus beinhaltet, dass der Einsatz von EDV-Systemen grundsätzlich notwendig, aber nicht immer in allen Dimensionen nötig ist. An dieser Stelle sind die Produzenten der Individualsoftware gefragt. Sie sind in der Lage Lösungen kundenspezifisch zu erstellen und auf Besonderheiten im jeweiligen Unternehmen einzugehen. Diese Art von Dienstleistung wurde vor wenigen Jahren noch als sehr vermessen angesehen und wurde selten realisiert.

Fazit:

Ein Zitat von Prof. Scheer, Institut für Wirtschaftsinformatik in Saarbrücken, unterstreicht die bisherigen Erkenntnisse: *„Hinter der heutigen Softwareentwicklung steht, dass die große Zeit der monolithischen und proprietären Systeme (ERP, PPS) zu Ende geht. Mit dem Internet hat ein massiver Technologieumbruch begonnen, vergleichbar dem Übergang von der Mainframe- zur Client/Server-Architektur. An die Stelle der integrierten Standardanwendungspakete treten Softwareeinheiten für einzelne Anwendungsfunktionen, die zu größeren Systemen zusammenmontiert werden*." [04]

Die folgende Arbeit soll ein Konzept beschreiben, wie in einem Unternehmen individuelle Systemlösungen objekt- und komponentenbasierend entwickelt und eingesetzt werden können. Dazu wird zunächst eine Reihe von kleinen mittelständischen Unternehmen mit weniger als 100 Mitarbeitern analysiert. Der Schwerpunkt der IST-Analyse liegt in der Ermittlung der Abwicklung der jeweiligen Geschäftsprozesse und der an ihr beteiligten Mitarbeiter mit den hier jeweils eingesetzten Softwaresystemen.

2.2 Übersicht über die befragten Kleinunternehmen

Im Rahmen einer Befragung von ca. 15 kleinen mittelständischen Unternehmen bezüglich der Nutzung der Informationstechnologie in der Verwaltung, der Arbeitsvorbereitung, der Konstruktion und der Fertigung wurde besonders deutlich, dass die einzelnen Unterneh-

men die Informationstechnologie in einzelnen Bereichen bereits nutzen, aber eine durchgängige und unternehmensweite Nutzung nirgends zu erkennen war. Einen Auszug aus der Befragung der Unternehmen zeigt die Abbildung 2-1, wobei zur Darstellung der Problematik bewusst Unternehmen aus unterschiedlichen Branchen mit verschiedener Fertigungsart ausgewählt wurden. Die Aussagen der restlich befragten Unternehmen sind deckungsgleich mit den hier vorgestellten Ergebnissen.

Die Tabelle zeigt, dass keines der Unternehmen einen Verantwortlichen für den EDV-Einsatz abstellt, vielmehr wird diese Aufgabe von irgendeinem Mitarbeiter aus der Verwaltung oder der Arbeitsvorbereitung neben seiner regulären Tätigkeit übernommen. Zwangsläufig kann es dann auch kein durchgängiges, zukunftsorientiertes und geschlossenes Konzept für die Nutzung der Informationstechnologie geben. Diese Tatsache spiegelt sich auch in den hier vorgenommenen Ausgaben für den jeweiligen EDV-Einsatz eindeutig wider. Darüber hinaus existieren zwischen den tatsächlich eingesetzten Software-Produkten kaum Schnittstellen, so dass die jeweils erzielten Daten von Hand in die nachfolgenden Systeme eingegeben werden. Dies wiederum führt zwangsläufig zu Übertragungsfehlern, was die Akzeptanz bei einigen Mitarbeitern erheblich mindert.

Mit einem durchgängigen Konzept kann dieses Defizit behoben werden. Des Weiteren werden durch die Realisierung dieses Konzeptes die Durchlaufzeiten erheblich reduziert und damit die Transparenz wesentlich gesteigert. Die Entwicklungskosten können durch die Schaffung eines überwiegend allgemeingültigen Konzeptes auf verschiedene Anwender verteilt und damit auch für das einzelne Unternehmen wirtschaftlich gestaltet werden.

Kriterium	**Kandidat A**	**Kandidat B**	**Kandidat C**	**Kandidat D**
Branche	Maschinenbau	Härtetechnik	Teppich-Herstellung	Druckerei
Fertigungsart	Lohn- und Einzelfertiger	Auftragsbez. Serien	Auftragsbez. Unikate	Einzelauftrag mittl. Serien
Umsatz	5.000.000	3.500.000	1.500.000	1.800.000
EDV-Kosten / Jahr	10.000	25.000	2.000	40.000
Mitarbeiter insges.	70	35	13	15
Mitarb. Verwaltung	13	3	1	1
Mitarb. Arbeitsvorber.	6	12	1	1
Mitarb. Konstruktion	1,5	0	1	2
Mitarb. Fertigung	50	20	10	11
Mitarb. EDV-Einsatz	0	0	0	0
Software Verwaltung	Word, Access	COMET	Word, Excel	Lektor
Software Konstruktion	AutoCAD	-	Designer	QuarkPress
Software Berechnung	-	Eigene	-	-
Software Fertigung	-	Excel	-	-
Durchlaufzeit	max. 2 Monate	max. 2 Wochen	max. 4 Wochen	mehrere Tage
Transparenz	4	4	2-3	2-3

Abbildung 2-1 Befragung von Klein-Unternehmen

Unter dem Begriff Transparenz wird hier verstanden, die Zurückverfolgung und Durchschaubarkeit von innerbetrieblichen Prozessen und Zuständen, wie z.B. Fertigungszustände, Auftragsstände, Stati und Historien von Dokumenten, Forderungen, Verbindlichkeiten, Auftrags-Prioritätenlisten, Lieferrückstände oder Terminverfolgung usw..

2.3 Bearbeitung der Geschäftsprozesse

Die Abbildung 2-2 zeigt die momentane Situation bezüglich der eingesetzten Software entlang der Wertschöpfungskette in einem Kleinunternehmen, wobei aus Vereinfachungsgründen die einzelnen Unternehmensbereiche nur grob gegliedert wurden.

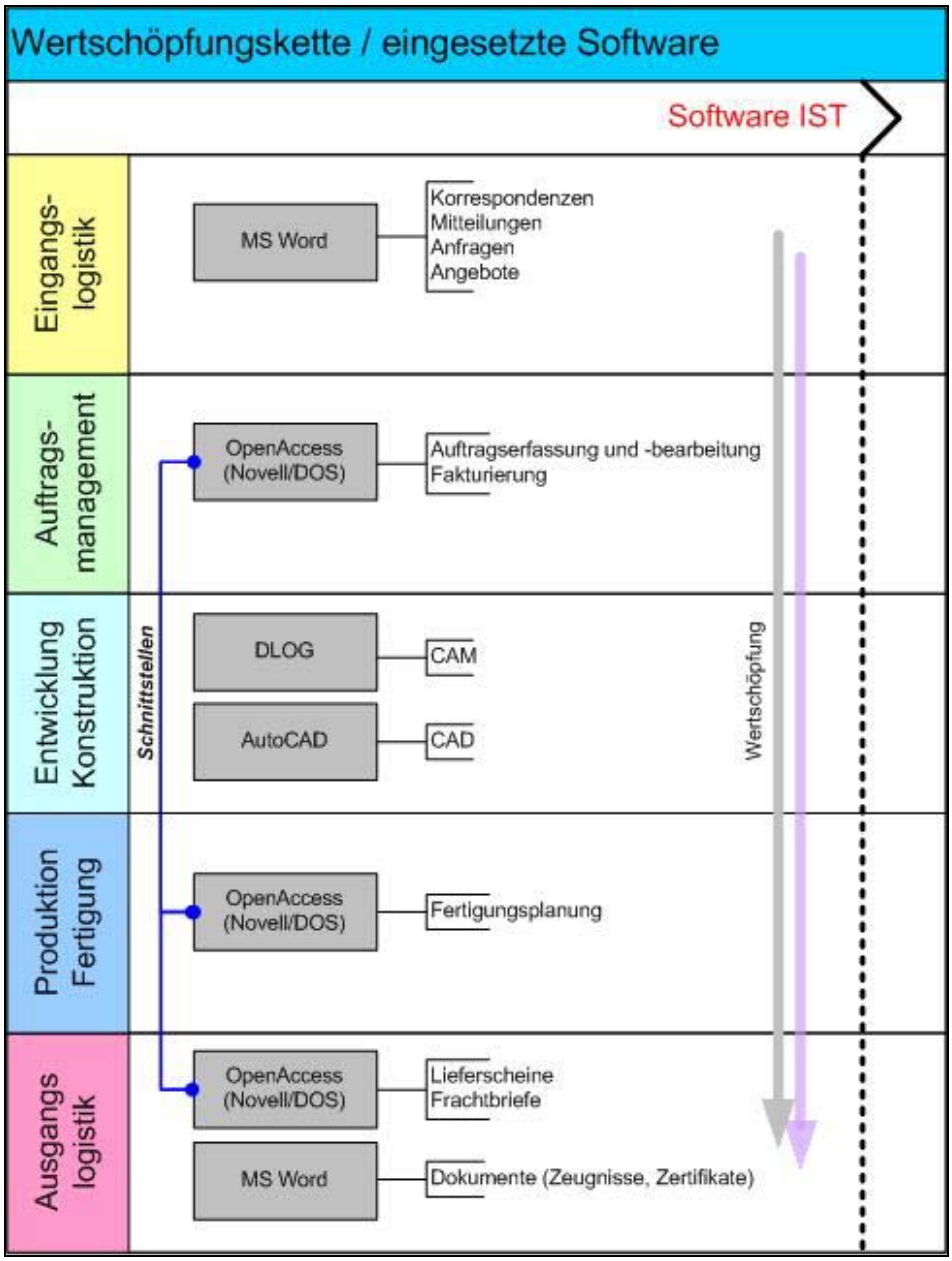

Abbildung 2-2 Flussdiagramm (IST-Zustand)

Die im Beispiel-Unternehmen eingesetzte Software entlang der Wertschöpfungskette trägt maßgeblich zur Transparenz der innerbetrieblichen Prozesse bei. Wie in der Abbildung 2-2 erkennbar, existieren für jeden Bereich unterschiedliche Softwarelösungen. Die Möglichkeit der Kommunikation bzw. des Datenaustausches zwischen den nichtverwandten Anwendungen fehlt, d.h. es existiert keine gemeinsame Datenbasis. Nur durch mehrfache Datenpflege in den verschiedenen Programmen sind diese Lücken im Informationsfluss zu schließen. Hinzu kommt, dass die Informationen in der Ein- und

Ausgangslogistik sowie in dem Bereich der Entwicklung und Konstruktion, speziell CAM[6], nicht in Datenbanken, sondern nur in Verzeichnissen abgelegt werden. Dieser Zustand lässt für spätere Recherchen nur eine zeitintensive und unvollständige Informationsbeschaffung zu.

Dieser Umstand beeinflusst die Effektivität der Geschäftsprozesse negativ, infolgedessen sinkt die Leistungsfähigkeit des Unternehmens. Das ganze Dilemma wird noch deutlicher, wenn die Bearbeitung eines einzelnen Geschäftsprozesses näher betrachtet wird, wie die Abbildung 2-3 für den Geschäftsprozess Auftragsbearbeitung zeigt.

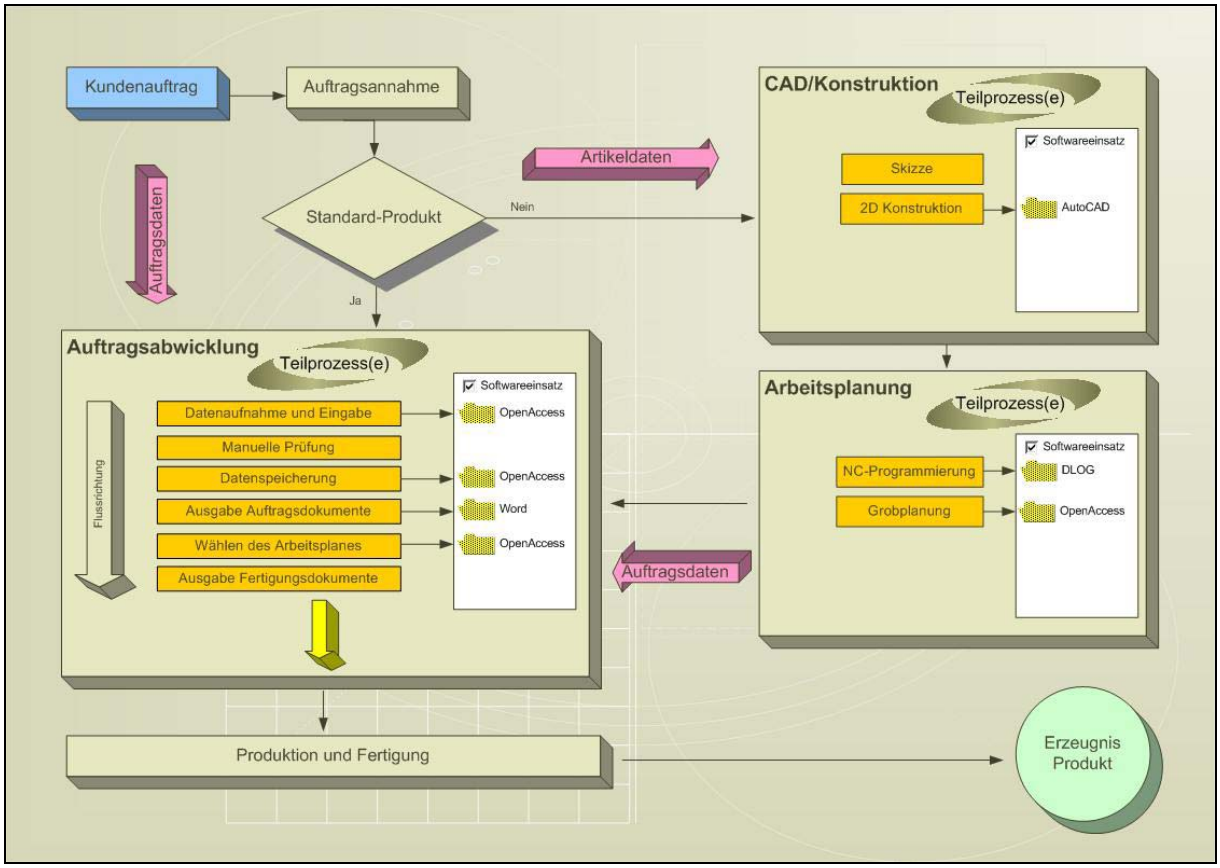

Abbildung 2-3 Geschäftsprozess Auftragsbearbeitung (IST-Zustand)

Bei der Betrachtung der Auftragsbearbeitung kann die eingesetzte Software den Prozess begleiten. Da aber die Auftragsbearbeitung keinen direkten Zugriff auf den Artikelstamm besitzt und somit Zweideutigkeiten nicht ausgeschlossen werden können, steigt die Fehlerwahrscheinlichkeit. Die fehlende gemeinsame Datenbasis fordert einen erheblichen Mehraufwand. Ferner werden die Änderungen am Artikel in der Auftragsbearbeitung nicht bemerkt und Konflikte im weiteren Verlauf sind vorprogrammiert. Weiterhin können keine

[6] Computer Aided Manufacturing

Rückschlüsse auf artikelabhängige Dokumente (Zeichnungen, Zeugnisse oder Protokolle) gezogen werden, da die Verknüpfung Auftrag-Artikel fehlt. Auftragsbezogene Dokumente, wie z.B. die Auftragsbestätigungen, können nur unabhängig von der Datenbank in einer Textverarbeitung erzeugt werden. Status-Informationen zu laufenden Aufträgen sind nur bedingt vollständig, da durch die Eigenständigkeit der Auftragsbearbeitung die Aktualität und der Informationsgehalt leidet.

2.4 Technischer Stand konventioneller Systeme

Die heute eingesetzten Systeme basieren überwiegend auf Windows oder Unix, sind modular aufgebaut und der Datenbestand wird in relationalen Datenbanken abgelegt. Die Technologien, mit der PDM-, PPS oder ERP-Systeme erstellt werden, sind, bedingt durch ihre Historie, unterschiedlich. Die Entwicklungen der Softwarehersteller beinhalten zu 80 bis 90 Prozent die Weiterentwicklung und Wiederverwendung ihrer vorhandenen Software. [05] Die jeweils zu Beginn einer Software-Entwicklung zur Verfügung stehenden Techniken sind auf Grund schnell fortschreitender Innovationen schnell überholt und der Einsatz neuer Technologien wird häufig hinausgezögert, um bei einem eventuell notwendigen Update möglich Probleme zu umgehen.

So werden z.B. Datenbankanwendungen mit Werkzeugen erstellt, die nicht von den Technologieentwicklern stammen, sondern von Datenbankentwicklern. In der Folge kommt es zu einem großen zeitlichen Versatz bezüglich der Anwendung moderner Technologien, die gerade bei produzierenden Unternehmen nicht gewünscht ist. Wenn Systemhäuser ihre Werkzeuge selbst entwickeln, ist der Zeitversatz noch größer, denn die Entwicklungswerkzeuge müssen wiederum an die Datenbankwerkzeuge und an die neuen Technologien angepasst werden. Hat ein Softwarehersteller nicht das Vermögen, sich ständig mit der Entwicklung auseinander zu setzen und die entsprechenden Abteilungen mit einzubeziehen, werden neue Softwareprodukte immer technologisch veraltet sein.

Es gibt nur wenige Hersteller, die eine akzeptable Lösung für solche Probleme bereitstellen und technologisch moderne Software anbieten. Mit diesen Problemen muss sich heute jeder Softwareproduzent auseinandersetzen. Wird diese Tatsache auf komplexe systemübergreifende Anwendungen wie z.B. PPS-, PDM- und ERP-Systemen repliziert, wird deutlich, wie schwer Support, Wartung und Versionisierung zu verwirklichen sind und eigentlich eine unlösbare Aufgabe darstellen.

Die Schnittstellen, die den Datenaustausch zwischen verschiedenen Systemen realisieren, sind vielfältig, kompliziert und schlecht dokumentiert. Es existieren keine Standards,

wie einheitliche Typsysteme oder Datenzugriffsmethoden. Des Weiteren sind unterschiedliche APIs[7] und minderwertige Komponenten Bestandteil der Softwaresysteme. Es werden schnell die Grenzen deutlich, welche die Erstellung komfortabler Software unmöglich machen.

Große Softwareanbieter besitzen meistens eigene Standards, Richtlinien und Werkzeuge, d.h. eine Fremdsoftware kann zwar eingebunden werden, aber nicht nach allgemeinen Standards, sondern nur basierend auf den Schnittstellen des jeweiligen Produzenten. Dass dieser Fakt Auswirkung auf Entscheidungen beim Softwareeinsatz des Anwenders hat, ist die logische Konsequenz.

Fallbeispiel:

In einem Unternehmen ist ein PPS-System eines großen Softwareanbieters im Einsatz. Es besteht hier die Forderung, eine Dokumentenverwaltung (PDM) zu integrieren, die aus verschiedenen Gründen nicht vom Anbieter des PPS-Systems stammt. Da Redundanzen unerwünscht sind, muss die Datenbasis der beiden Systeme identisch sein. Die CAD-Software des Unternehmens liefert die Information für die neue Dokumentenverwaltung und der Informationsaustausch zwischen PPS und PDM muss bilateral ablaufen, d.h. es werden die Informationen über diverse Stati und Bezeichnungen des PPS-Systems auch in der Dokumentenverwaltung benötigt. Daraus folgt, dass drei unterschiedliche Softwareprodukte von drei verschiedenen Anbietern kommunizieren müssen. Der Anspruch an die Programmierer ist enorm, sie müssen alle drei Schnittstellen beherrschen und diese verknüpfen. Dass dabei Probleme auftreten, ist bei den verschiedenen Standards unvermeidlich, allein die Konvertierung der Datentypen ist beispielhaft umständlich und aufwendig. Von einer wirklichen Integration kann insgesamt nicht ausgegangen werden.

Es existieren zwar Standard-Schnittstellen, wie DDE (Dynamic Data Exchange), OLE (Object Linking and Embedding) oder COM (Component Object Model), diese sind zum Teil aber nicht mehr zeitgemäß und besitzen viele Nachteile. Stellvertretend für diese Nachteile sind gemäß [06] zu nennen:

- keine Standards für die Verknüpfung binärer Objekte,
- extreme Abhängigkeiten von Programmiersprache,
- Entwicklungsumgebung und Betriebssystem,
- folgenreiche Änderungen der Schnittstellen (Versionskonflikte) und
- die Existenz unterschiedlicher Typsysteme (String1 ungleich String2).

[7] Application Programming Interface

COM-Komponenten sind zwar effektiver und komfortabler als andere klassische Standard-Schnittstellen, aber aus programmiertechnischer Sicht ist selten ein guter Umgang mit ihnen möglich, da sie, wie erwähnt, system-, programmiersprachen- und versionsabhängig sind. Hinzu kommt, dass COM selbst auf anderen Objektmodellen basiert und eine von vielen APIs (MFC[8], Win32[9]) ist. Die Fakten machen deutlich, dass bei der Softwareentwicklung grundlegende Änderungen erforderlich sind und sich daraus die Anforderungen an eine neue Software formulieren lassen. Diese Ansprüche sind hoch und stellen für die Zukunft eine klare Aufgabe. Generell entstehen durch diese Defizite prinzipielle Entwicklungsprobleme, deren Eliminierung Bestandteil der künftigen Aufgabe ist.

2.5 Zusammenfassung der Defizite

Zusammengefasst liegen bezüglich des Software Engineering die folgenden Probleme vor:

- Kommunikationsprobleme mit dem Anwender
- schwierige Modellbildung für Software
- leichte Modifizierbarkeit von Software
- Änderung der Anforderungen während der Entwicklungszeit
- Portabilitätsprobleme
- Umgang mit der Explosion der Variantenvielfalt
- Verknüpfung mit der Abbildung von bestehenden Arbeitsabläufen (Akzeptanz beim Anwender)
- Mangel an Standards, Methoden und Werkzeugen

[8] Microsoft Foundation Class

[9] Windows 32-Bit Application Programming Interface

3 ANFORDERUNGEN

Die Anforderungen an neue Systeme lassen sich in zwei Kategorien einteilen, zum einen in den softwaretechnischen und zum anderen in den funktionellen Teil. Beide Kategorien lassen sich aus den Defiziten der heutigen Situation mittelständischer Unternehmen und dem technischen Stand konventioneller Systeme ableiten.

3.1 SOFTWARETECHNISCHE ANFORDERUNGEN

Heute zu entwickelnde Anwendungen dürfen keine monolithischen Eigenschaften aufweisen, sondern sind komponentenorientiert aufzubauen. Die Vergangenheit und die Gegenwart zeigen, dass Komplettlösungen Komponenten beinhalteten, die durch andere Softwareproduzenten besser realisiert wurden. Die Kombinationen von unterschiedlichen Systemen, wie PDM, ERP und PPS sind bis heute nicht einfach zu realisieren, besonders dann, wenn die Systeme von unterschiedlichen Herstellern stammen.

So sind die Entwickler neuer Systeme aufgefordert, die definierten Standards von Grund auf einzuhalten und einem durchdachten Programmierparadigma zu folgen. Kreativität oder Innovationen bei Entwicklungen dürfen auch künftig nicht vernachlässigt werden, sie stehen nicht im Konflikt mit der Einhaltung von Richtlinien und Standards. Halten sich die Softwarearchitekten und Produzenten an diese Vorgaben, werden die gesteckten Ziele und damit eine bessere Kundenzufriedenheit erreicht.

3.1.1 Qualität der Software

Ein hoher Qualitätsstandard bei Systemsoftware wird nicht nur vom Anwender erwartet, auch beim Service und Support sowie bei der Weiterentwicklung macht sich eine hochwertige Software bezahlt. So müssen alle Komponenten, die ein System ausmachen, ein Mindestmaß an Qualität bereitstellen. Eine konsequente Anwendung und Einhaltung der Regeln bei der Softwareentwicklung ist selbstverständlich. Die Bausteine, wie Datenbanken, Framework und Betriebssysteme, sind die Fundamente, die die Sicherheitsbasis bilden und die notwendigen Werkzeuge bereitstellen. Die Datenbanktechniken sind ausgereift und zuverlässig, die aktuellen Betriebssysteme werden ständig auf Fehler geprüft und bei Bedarf korrigiert und das Framework bietet Mittel, die einen hohen Sicherheitsstandard im Programmcode garantieren. Die Voraussetzungen, einem qualitativ hohen Anspruch zu genügen, sind gegeben.

3.1.2 Verfügbarkeit und Sicherheit

Die Sicherheit von Datenbankanwendungen wird durch den Datenbankzugriff geregelt, dieser erfordert eine strenge Authentifizierung und Autorisierung. Laufen Datenbanken auf zuverlässiger Hardware und wurden spezielle Hardwarekomponenten (z.B. Raid[10]) eingesetzt und bei der Installation der Netzwerkkomponenten ebenfalls auf Qualität geachtet, sind Systeme hochverfügbar und sicher. Die Sicherheitsaspekte der Anwendung werden im Programmcode und die lokale Sicherheit im Betriebssystem implementiert.

Die Robustheit und Stabilität der Datenbanken wird in der Regel nicht in Frage gestellt. Diese Eigenschaften gelten mittlerweile auch für Betriebssysteme, die ordentlich installiert und konfiguriert sind. Meist sind es Anwendungen die auf den Rechnern laufen, die die Instabilitäten verursachen. Die Ursachen für Instabilität, wie unbefugter Speicherzugriff, Speicherlecks, falsche Typzuweisung, ungültige Argumente und Laufzeitfehler, gehören bei strikter Einhaltung vorgegebener Richtlinien der Vergangenheit an.

3.1.3 Bedienung

Der Aufbau und der Umgang mit der Anwendung richten sich in erster Linie nach der Zielgruppe. Technische Anwendungen verlangen ein Mindestmass an Grundwissen und bestimmte Fähigkeiten, sie können daher nicht im ähnlichen Stil erstellt werden wie "Freizeitanwendungen", müssen aber dennoch für Benutzer mit unterschiedlichen Voraussetzungen verständlich sein.

3.1.4 Wartbarkeit und Erweiterbarkeit

Ist Software transparent, steigert das ihre Wartbarkeit und Erweiterbarkeit. Um Transparenz zu schaffen, muss das Software- und Datenbankdesign durchdacht und logisch aufgebaut sein. Diese Aussage bezieht sich generell auf die Entwicklung, beim Einsatz der Software spielt hingegen die Art und Weise der Konfiguration eine wesentliche Rolle. So darf es beispielsweise nicht sein, dass Einstellungen an vielen Stellen vorgenommen werden müssen, die eigentlich zentral erfolgen können. Die benutzerdefinierten Einstellungen und Optionen sollten sich auf sinnvolle Konfigurationen beschränken.

[10] Redundant Array of Independent Disks

3.1.5 Investitionsschutz

Die ständige Konsolidierung des IT[11]-Marktes durch Übernahmen und Insolvenzen geht zu Lasten der Investitionssicherheit. Kunden fürchten um die Planungssicherheit und Zukunftsfähigkeit der eingesetzten Software, da beispielsweise die Weiterentwicklung und das Fortbestehen des Supports gefährdet sind. Demzufolge müssen die konsequente Einhaltung aller notwendigen Standards und die Unabhängigkeit von einer bestimmten System-Plattform gewährleistet werden. Die Entscheidung für eine bestimmte Hardware wird unabhängig von der Softwareentscheidung getroffen. Auch künftig muss es dem Kunden freistehen, die Hardware auszutauschen oder zu erweitern. Die Entwicklung mit standardisierten und objektorientierten Sprachen trägt zur geforderten Systemoffenheit bei, sie vereinfacht die spätere Integration von Fremdsystemen (CAD, NC[12] usw.). Weitere Vorteile sind die Ausrichtung des Frameworks und der Entwicklungswerkzeuge auf die Internetdomäne, welche die Zukunft der IT-Landschaft prägt. Wann immer Anpassungen oder Erweiterungen für das System geplant sind, können entsprechende Leistungen auch unabhängig vom Hersteller oder Integrations-Partner auf dem Markt erworben werden. Investitionen können dann entsprechend den tatsächlichen Bedürfnissen gestaltet werden, da so entwickelte Software flexibel, anpassungsfähig und skalierbar ist.

Anpassungen und Erweiterungen können dann realisiert werden, wenn die Unternehmensentwicklung es verlangt. Die eventuell vorhandenen Altsysteme können nach Bedarf oder auch sukzessive abgelöst werden.

3.2 Anforderungen an den Funktionsumfang

Der grundlegende Bereich der funktionalen Anforderungen stammt aus dem Umfeld der CAD-Datenverwaltung und angrenzender Unternehmensbereiche. Diese Anforderungen sind demnach identisch mit den Anforderungen, die an konventionelle PDM-Systeme gestellt werden. Da die Erfüllung dieses Teilbereiches der funktionalen Anforderungen heute im Bereich der PDM-Systeme zum Stand der Technik gehört, erfolgt an dieser Stelle keine detaillierte Aufzählung der einzelnen Aufgaben. Diese Anforderungen lassen sich direkt aus dem Leistungsumfang der PDM-Systeme ableiten.

Um jedoch alle Geschäftsprozesse eines Unternehmens abzubilden, werden weitere Komponenten benötigt. So sind der betriebswirtschaftliche und fertigungstechnische Aspekt noch zu implementieren. Da die Basiskomponenten eines PDM-Systems nur einen Teil

[11] Information und Telekommunikation

[12] Numeric Control

des Funktionsumfanges abdecken, der benötigt wird, müssen Überlegungen angestellt werden, welche weiteren Komponenten zwingend notwendig sind. Die Entscheidungen über den Komponentenumfang sind Teil des Pflichtenheftes bei der Erstellung der individuellen Software. In den folgenden Abschnitten werden einzelne Komponenten, die gefordert werden, kurz erläutert.

3.2.1 Integration der betriebswirtschaftlichen Aspekte

Jedes Unternehmen benötigt mindestens Kunden und Lieferanten, hat einen Vertrieb und einen Einkauf. Welche weiteren Module benötigt werden, ist unternehmensspezifisch. Im Folgenden werden die für diese Arbeit relevanten Module kurz beschrieben, da diese in der Regel im Lastenheft des Unternehmens fixiert sind.

Firmen- und Personalstamm (Kunden, Lieferanten, Mitarbeiter, Benutzer): Der Firmenstamm beinhaltet einen Kunden-, Lieferanten-, Mitarbeiter- und Benutzerstamm. Die Kunden und Lieferanten sind für vertriebstechnische Vorgänge zwingend notwendig. Der Mitarbeiterstamm dient zur Verwaltung des Personals im Unternehmen und die Benutzer sind Teilmenge der Mitarbeiter, die in Gruppen aufgeteilt sind und unterschiedliche Aufgabengebiete und dazugehörige Rechte im System besitzen.

Angebotserstellung: Aktionen und Informationen in der Vorverkaufsphase können mit Angeboten im System festhalten werden. Bittet ein Kunde um Informationen zu Produkten oder Dienstleistungen, können diese in Form eines Angebotes formuliert werden. Meist beinhalten die Angebote konkrete oder von Variablen abhängige Preise. Aufwandbezogene Angebote können auf der Basis eines detaillierten Arbeitsplanes ermittelt und im Angebot mitgeteilt werden.

Auftragserfassung, Auftragsabwicklung: Die Erfassung eines Auftrages ist der Grundstein für alle weiteren Geschäftsprozesse. Erst wenn ein Auftrag erstellt wird, werden Folgeaktionen angestoßen. Ein Auftrag setzt Eigenschaften und Voraussetzungen, die für weitere Prozesse im Betrieb notwendig sind. Erst nach Erfüllung dieser Bedingungen ist es möglich Prozesse anzustoßen, die zur Durchführung des Auftrages (Produktion) benötigt werden. Der Auftrag beinhaltet mindestens:

- welche Leistung zu erbringen ist
- wann die Leistung zu erbringen ist
- welche Höhe die Auftragskosten haben
- für wen die Leistung zu erbringen ist

Die Bearbeitung von Vorgängen, wie das Schreiben eines Lieferscheins, einer Rechnung oder Gutschrift, erfolgt auftragsorientiert. Ausgehend vom Kunden werden die Belege mit

den wichtigsten Daten automatisch ausgefüllt. Die Rechnungen und Lieferscheine werden überwiegend automatisch erstellt und für die Finanzbuchhaltung gebucht, sollten aber noch editierbar sein. Eine flexible Rabatt- und Preisfindung benötigen meist Handelsunternehmen, da z.B. bei einem Lohn- und Einzelfertiger die Preise über die Lohnarbeit und den Materialaufwand berechnet werden. Abweichende Liefer- und Rechnungsempfänger sind zwar nicht immer selbstverständlich, sollten aber zum Standard gehören. Übersichten über die Auftragsstände sowie Auftragseingangs- und Umsatzstatistik sind sinnvolle Ergänzungen, ebenso die direkte Verbindung zum Fertigungsauftrag/Werkstattauftrag.

Einkauf und Bestellwesen: Der Einkauf und das Bestellwesen (Beschaffung) stehen in gegenseitiger Abhängigkeit und sind deshalb meist in einem Modul vereint. Die Hauptmerkmale werden sowohl durch auftragsbezogene Disposition als auch freie Bestellungen, die den Bedarf des Unternehmens decken, dargestellt. Die Bestellhistorie und detaillierte erweiterte Statistiken sind in der Regel Bestandteil dieses Moduls. Übernahmemöglichkeiten aus den Auftragsdaten und die Terminüberwachung für Bestellungen sind häufig Standard. Für das Qualitätsmanagement, hier speziell für die Lieferantenbewertung, bieten der Einkauf und das Bestellwesen optimale Grundlagen. Das Bestellwesen unterstützt die Verwaltung des Einkaufs und der Lagerbestände. Mit diesem Baustein können Lagerbestände geprüft, mehrere Lieferanten zu einem Artikel hinterlegt und manuell oder automatisch Bestellungen ausgelöst werden.

Buchungen, Inventar, Firmeneinsatz, Kasse: Die Buchungserfassung bietet die Möglichkeit, getrennt nach Kontenklassen, Belege (Ein- und Ausgangsrechungen) zu buchen und auszuwerten. Sie bietet Konten-, Summen- und Saldenlisten sowie Bilanzen und Statistiken für selbstdefinierte Intervalle und Kriterien. Es sind Exportmechanismen für Standardüberweisungen oder die Ausgabe von Schecks vorgesehen. Die Unterstützung von Buchungen für Kassenbelege wird automatisch durch die Kontenklassenzuordnung bereitgestellt. Einige Komponenten, wie Firmeneinsatz- und Inventarbuchung, sind optional möglich und dienen zur überschlägigen Bilanzerstellung kleiner Unternehmen.

3.2.2 Integration der fertigungstechnischen Aspekte

Um den Bedürfnissen eines Unternehmens aus der Sicht der Fertigung gerecht zu werden, müssen Komponenten vorhanden sein, die diese Aufgaben unterstützen. So sind die Verwaltung von Kostenstellen, Arbeitsgängen, Maschinen bzw. Arbeitsorten und darauf aufbauend Arbeitspläne, Fertigungsaufträge (Betriebsaufträge), Betriebsdatenerfassung sowie Grob- und Feinplanung erforderlich. Durch eine schematische Gliederung der Arbeitsgänge, Maschinen/Arbeitsorte und Kostenstellen wird eine realitätsnahe hierarchische Strukturierung der Arbeitspläne und Fertigungsaufträge ermöglicht. Diese intelligente Gestaltung gestattet eine Preisfindung, die nah an der Praxis liegt. Durch die hierarchi-

sche Strukturierung wird gleichzeitig die Grundlage für eine eindeutige Arbeitsgangidentifizierung für die Betriebsdatenerfassung und Feinplanung geschaffen.

Kostenstellen: Die Erfassung und Verwaltung der einzelnen Kostenstellen im Unternehmen, z.B. Maschinen, Abteilungen, Personen etc., ist zur Planung und Fakturierung zwingend notwendig. Es existieren Kostenstellengruppen, die beliebig Maschinen und/oder Abteilungen (Arbeitsorte) zusammenfassen können und dadurch individuelle Stunden- und Rüstsätze ermöglichen.

Arbeitsgänge: Arbeitsgänge sind freidefinierbare Tätigkeiten, die an den Maschinen oder in den Abteilungen durchgeführt werden. Diesen Arbeitsgängen werden vor- oder benutzerdefinierte Texte zugeordnet, welche die Arbeitsgänge beschreiben.

Maschinen/Arbeitsorte: Arbeitsorte oder Maschinen sind Teilmengen der einzelnen Kostenstellen, d.h. eine Kostenstelle muss mindestens eine Maschine oder einen Arbeitsort beinhalten.

Arbeitspläne: Die Arbeitspläne bilden den Grundstein für alle folgenden Funktionen und Komponenten des Fertigungsbereiches. Da die Kalkulation und die Herstellung der Artikel fast identisch aufgebaut sind, dient der Arbeitsplan gleichzeitig als Kalkulationsgrundlage. Dies minimiert parallel die Fehlerquote bei der Angebotserstellung.

Fertigungsaufträge: Die Fertigungsaufträge (FA) stellen Instanzen der Arbeitspläne dar, die zusätzlich fertigungsspezifische Merkmale aufweisen. Ein wichtiges Merkmal ist die Möglichkeit der kompletten Modifikation des Fertigungsauftrages, da auftragsabhängige Konstellationen in der Fertigung alltäglich sein können. Diese spezifischen Situationen beeinflussen dabei nicht den Arbeitsplan, dieser ist datentechnisch losgelöst vom Fertigungsauftrag.

Betriebsdatenerfassung (BDE): Ein wichtiges Werkzeug zur Erreichung eines stets aktuellen Datenbestandes ist die Betriebsdatenerfassung. Diese Dateneingabeart sorgt für einen einfachen und schnellen Datenabgleich durch relevante Mitarbeiter. Sie melden Arbeitsgänge an und ab, unterbrechen diese oder nehmen sie wieder auf. Auf diese Weise vereinfacht sich die Kapazitätsplanung (Feinplanung), da immer die aktuellen Stati der Arbeitsgänge in der Datenbank vorliegen.

Feinplanung: Die Feinplanung ist aus der Sicht der Fertigung das wichtigste Instrument zur detaillierten Verwaltung der Produktion, weiterhin stellt sie den Informationspool über den kompletten Produktionsprozess dar. Es werden Daten tabellarisch und grafisch interpretiert, die Arbeitsgänge, Kommissionen, Kostenstellen, Maschinen bzw. Abteilungen abhängig vom gewählten Zeitraum anzeigen. Die Feinplanung bietet das Verschieben und

Splitten von Arbeitsgängen sowie kompletter Fertigungsaufträge an, weiterhin werden alle Vorgänge, die den Status betreffen (anmelden, abmelden usw.), unterstützt. Alle relevanten Informationen stehen sofort zur Verfügung, z.B. Infos zur Kommission eines Arbeitsganges oder Belegungsinformationen von Maschinen/Abteilungen bzw. Kostenstellen. Eine häufig geforderte Information ist die Belegung einer speziellen Maschine in einer Kalenderwoche mit den dazugehörigen Kommissionen auf einen Blick oder die Verteilung der Arbeitsgänge in einer Abteilung ohne Zeitbegrenzung.

3.2.3 Integration zusätzlich notwendiger Komponenten

Um dieses System abzurunden bzw. Informationslücken zu schließen, sind Lagerverwaltung, Knowledgemanagement, Workflow, Zeiterfassung und Qualitätsmanagement vorteilhaft.

Lagerverwaltung: Eine rationalisierte Lagerverwaltung mit den grundsätzlichen Funktionen, wie Lagerein- und -ausgang, Reservierung, Chargen und Mindestbeständen sind für Beschaffungsvorgänge erforderlich. Werden Werkzeuge sowie Hilfs- und Betriebsstoffe mit einbezogen, wirkt sich dies positiv auf eine Nachkalkulation aus, vorausgesetzt, die Ausgabe der Werkzeuge bzw. Hilfs- und Betriebsstoffe erfolgt kommissionsabhängig. Darüber hinaus leistet eine Lagerverwaltung aussagekräftige Überblicke über Bestand und Werte der im Lager befindlichen Objekte.

Knowledgemanagement: Hervorragend für die Wissensverbreitung im Unternehmen ist eine Knowledgebase (Wissensdatenbank). Hier werden nach Bedarf alle auftragsunabhängigen Informationen gespeichert, um sie zu späteren Zeitpunkten abrufen zu können. Die Klassifizierung der Einträge nach Typ und Bereich bietet eine schnelle Suche sowie eine kompakte Informationsquelle.

Workflow: Der Workflow ist in heutigen Anwendungen kaum noch wegzudenken und zum Pflichtteil eines PDM-Systems geworden. Workflows werden in der Regel als Aufgabensteuerung angesehen, sie beschreiben und definieren den Aufgabenfluss in einem Unternehmen. Die Richtlinien- und Aufgabendefinition sowie deren Verteilung sind optional einstellbar. Bei der Zielgruppe dieser Arbeit, den Lohn- und Einzelfertigern, wirkt sich der vordefinierte Fluss der Aufgaben eher arbeitsverzögernd als –beschleunigend aus. Gerade in mittelständischen Unternehmen sind Mitarbeiter, die ständig nur einen Bereich im Unternehmen abdecken, selten. Meist sind diese Mitarbeiter vielseitig und in jeder Abteilung einsetzbar oder sogar in mehreren Bereichen gleichzeitig tätig. Hier ein Flussdiagramm für den Workflow zu definieren, welches allgemeine Gültigkeit besitzt, ist schwierig und der Aufwand für die Aktualisierung nicht gerechtfertigt. Deshalb muss diese Komponente

auch freie Workflows gestatten. Selbstverständlich sind feste Verläufe für Aufgaben definierbar.

Zeiterfassung: Eine klassische optionale Komponente ist das Zeiterfassungsmanagement. Jedes Unternehmen hat eine andere Vorstellung, wie Arbeitszeit, Urlaub, Krankheit oder sonstige Abwesenheiten erfasst werden sollen. Meist reichen herkömmliche Stempeluhren. Auswertungen im gewissen Umfang sind auch mit nicht computerbasierenden Zeiterfassungssystemen möglich. Wer jedoch Zeiten EDV-technisch erfassen möchte und erweiterte Auswertungsmöglichkeiten verlangt, kommt um die Zeiterfassung über Computerterminals nicht herum. Die Methode der Erfassung, ob Barcode, Magnetstreifen, Infrarot oder Funk, ist entsprechend den Umgebungsbedingungen und dem finanziellen Rahmen zu wählen. Als robust, zuverlässig und optimal für Werkstattumgebungen hat sich die Barcode-Methode erwiesen.

Qualitätsmanagement: Viele Kunden verlangen schon seit einiger Zeit Zertifizierungen, Zeugnisse oder Nachweise der Lieferanten, um dem Qualitätsstandard, dem sie selbst unterliegen, z.B. der DIN EN ISO 9001:2000, gerecht zu werden.

Einige wichtige Anmerkungen zum Qualitätsmanagement müssen vorab formuliert werden, um seine Bedeutung zu unterstreichen. Im Rahmen der unternehmerischen Sorgfaltspflicht definiert sich die Qualitätspolitik im Hinblick auf die Qualitätssicherung und damit die Zufriedenheit der Kunden und die Erfüllung ihrer Erwartungen. Das Qualitätsmanagement-Handbuch stellt für das Unternehmen eine angemessene Beschreibung des Qualitätsmanagementsystems dar. Seine Anwendung gewährleistet, dass die organisatorischen, kaufmännischen und technischen Tätigkeiten, die Auswirkung auf die Ausführungsqualität haben, geplant, gesteuert und überwacht werden und dass vertraglich vereinbarte Anforderungen erfüllt werden.

Es erfüllt die Anforderungen der DIN EN ISO 9001:2000 und die Bestimmungen vertragsgemäß anzuwendender nationaler Regelwerke und Vorschriften. Im Konfliktfall zwischen den Anforderungen des Qualitätsmanagement-Handbuches gemäß DIN ISO 9001:2000 und anderen Qualitätsmanagement-Regelwerken gelten die Anforderungen der DIN EN ISO 9001:2000. Dadurch sind externe Richtlinien von Kunden nicht ausgeschlossen. Der QM-Beauftragte ist verantwortlich für die Planung, Überwachung und Korrektur des Qualitätsmanagementsystems. Er ist befugt und hat die organisatorische Freiheit, Qualitätsmanagementprobleme zu identifizieren, entsprechende Maßnahmen vorzuschlagen und die Durchführung dieser Maßnahmen zu überwachen. Dazu gehören Steuerung des Vorgehens bei Behandlung von Qualitätsabweichungen, Erstellung, Genehmigung, Verteilung und Pflege des Qualitätsmanagement-Handbuches sowie ständige Unterrichtung der Geschäftsführung über wesentliche Qualitätsmanagementprobleme. Mit

der Beurteilung der Ergebnisse interner Qualitätsaudits durch die Geschäftsführung und die Veranlassung sich daraus ergebender Maßnahmen ist eine Bewertung des Qualitätsmanagementsystems durch die Geschäftsführung gegeben.

Eine Unterstützung des QM im System ist demzufolge angebracht. Mit relativ wenig Aufwand können so die Daten eingepflegt werden, die zur Auswertung und Beurteilung des QS notwendig sind. Wird beispielsweise das Eintreffen einer Lieferung im System registriert, können in diesem Schritt auch gleichzeitig Aussagen über Pünktlichkeit, Qualität und Vollständigkeit der Lieferung getroffen werden. Bei der Beobachtung eines Lieferanten und dessen Lieferungen können schon nach wenigen Wochen Benotungen durchgeführt werden. Der Zeitaufwand für eine Lieferantenbewertung ist minimal und die Ausbeute sehr effektiv. Das Anwendungsgebiet des Qualitätsmanagementsystems ist unerschöpflich, hier zu entscheiden, wie weit gegangen werden soll, ist nicht trivial und hängt von vielen Faktoren ab. Die Entscheidungen diesbezüglich fällt der Beauftragte im Unternehmen.

3.3 Fazit der Anforderungen

Die Zusammenfassung der bisher formulierten Anforderungen liefert das folgende Bild: Zur Abdeckung aller Anforderungen ist ein System erforderlich, welches über die Funktionalitäten verfügt, die in Form von drei unabhängigen Systemfamilien angeboten werden. Es werden Funktionalitäten von PDM-, PPS- und ERP-Systemen benötigt. Daneben betrifft ein wesentlicher Bereich der Anforderungen softwaretechnische Aspekte, welche für die Auswahl der Basistechnologie des Konzepts eine ausschlaggebende Bedeutung haben. Die große Zahl der verschiedenartigen Anforderungen macht deutlich, dass es sich bei dem zu entwickelnden Konzept um ein äußerst komplexes System handeln wird. Die Frage ist, auf welcher Basis dieses Konzept umgesetzt wird. Prinzipiell besteht die Möglichkeit, ein System aus einer der oben genannten Kategorien PDM, PPS oder ERP auszuwählen, und dieses System durch individuelle Anpassungen um die erforderlichen Funktionen zu erweitern. Dem gegenüber steht die Alternative einer komplett neu entwickelten Applikation, welche exakt die gestellten Anforderungen erfüllt. Um diese Auswahl zu unterstützen, werden zunächst die Hauptanforderungen in einer Tabelle den einzelnen Systemen gegenüber gestellt. Die Bewertung der Erfüllung der Anforderungen erfolgt auf der Basis von umfangreichen Recherchen des Verfassers.

Für die Beurteilung wird ein Drei-Punkte Schema verwendet mit den Bewertungen:

- 2 = Gut erfüllt
- 1 = Bedingt / teilweise erfüllt
- 0 = Nicht erfüllt

Die Auswertung dieses Vergleichs zeigt eindeutig, dass ein individuelles Konzept die beste Möglichkeit darstellt, um eine umfassende Lösung aller Anforderungen zu erreichen. Besonders auch mit Blick auf die begrenzten finanziellen Möglichkeiten eines mittelständischen Unternehmens stellt eine Individualsoftware die einzige mit vertretbarem Aufwand zu realisierende Alternative dar. Da im Allgemeinen unter Hinweis auf die anzustrebende Standardisierung stets von der Verwendung individuell erstellter Software abgeraten wird, werden in der Folge einige relativierende Fakten zum Thema Individualsoftware aufgezeigt.

Anforderungen	**PPS**	**PDM**	**ERP**	**Konzept**
Einfachheit in Installation und Bedienung	0	1	1	2
Nutzung von Standards	0	0	0	2
Stabilität und Sicherheit	2	2	2	2
Gegebene Freiheitsgrade, offene Schnittstellen und Sprachpluralität	1	2	1	2
Ableitung der Stammdaten aus Realität	1	2	1	2
Grosse Funktionsvielfalt, Leistungsmerkmale, hoher Informationsgehalt	2	1	2	2
Durchgehende und lückenlose Widerspiegelung der Geschäftsprozesse	1	1	1	2
Technologisch aktuell zum Zeitpunkt der Einführung	0	0	0	2
Problemlose Integration in bestehende IT-Landschaft	0	2	1	2
Einfache Informationssuche auch über hierarchische Datenstrukturen	0	2	0	2
Einfache Versionsupdates	1	1	1	0
Support und Erweiterbarkeit	1	1	1	2
Investitionssicherheit	1	1	1	1
Finanzierbarkeit für den Mittelstand	0	2	1	2
Beständigkeit des Softwarelieferanten	1	1	1	1
Summe:	11	19	14	**26**

Abbildung 3-1 Anforderungs- und Entscheidungsmatrix

Die Befürchtungen, dass die Unternehmen, die individuelle Software anbieten, meist nicht marktführend, sondern selbst eher klein und mittelständisch sind und schnell vom Markt verzehrt werden, treffen nicht zu. Die Veränderung des Technologiemarktes in der heutigen Zeit macht auch vor großen Marktführern nicht Halt, d.h. ein großer Anbieter kann genau so schnell vom Markt verschwinden wie ein solider kleiner oder mittelständischer Dienstleister. Demzufolge spielen Größe und Marktposition des Unternehmens keine wesentliche Rolle. Als weiteres Kriterium kann die Zeitspanne von der Lösungsentscheidung bis zum Kompletteinsatz genannt werden. Individuelle Lösungen benötigen einen größeren Zeitraum, jedoch werden die Unterschiede zur konventionellen Softwareeinführung immer geringer, da der Anpassungsaufwand dieser Standardsysteme enorm ist und die Technologien der Individualentwicklungen ausgereifter und effektiver werden. Im Endeffekt ist die Entscheidung abhängig von der Branche und deren internen Betriebsabläufen, d.h. ob der Kunde überhaupt individuelle Software für die Prozessabbildung benötigt oder ob die internen Abläufe einem De-Facto-Standard entsprechen, wie beispielsweise bei Groß- oder Einzelhändlern. Des Weiteren hängen die Entscheidungen vom finanziellen Aufwand und - wie schon erwähnt - vom vorgesehenen Zeitrahmen ab.

Bevor das Konzept zur objekt- und komponentenorientierten Entwicklung vorgestellt wird, werden die eingesetzten Technologien und Grundlagen erläutert.

4 SOFTWARETECHNOLOGIEN

Die zuvor beschriebenen vielfältigen Probleme, die bei der Betrachtung der heute verfügbaren Systeme offenkundig werden, sind zu einem großen Teil darin begründet, dass diese Systeme auf der Basis veralteter Technologien entwickelt wurden, oder dass die gewählte Architektur den gewachsenen Anforderungen nicht gerecht wird. Ein Konzept, welches unter Berücksichtigung aller hier aufgestellten Anforderungen ein einheitliches System hervorbringt, welches auch für mittelständische Unternehmen geeignet ist, erfordert die Verwendung geeigneter Technologien aus dem Bereich der Softwareentwicklung. Daher wird in der Folge der Stand der Forschung auf dem Gebiet der Softwareentwicklung eingehend untersucht, um die für das Konzept geeigneten Verfahren und Werkzeuge zu bestimmen.

4.1 DATENBANK

Da für die Realisierung des Konzepts die Verwendung einer Datenbank zur sicheren Verwaltung der Informationen erforderlich ist, werden zunächst die Grundlagen der Datenbanken vorgestellt.

Prinzipiell existieren vier unterschiedliche Datenbankmodelle: das hierarchische Modell, das Netzwerkmodell, das relationale Modell und das objektorientierte Datenbankmodell. Die in Unternehmen eingesetzten Softwaresysteme basieren hauptsächlich auf relationalen Datenbanken, diese haben sich im Laufe der Zeit durch ihre Stabilität, Robustheit, Geschwindigkeit und Sicherheit durchgesetzt. Die Informationen werden in Tabellen als Datensatz (Zeile in einer Tabelle) gespeichert. Durch zusätzliche Verknüpfungen werden hierarchische Strukturen und Beziehungen abgebildet.

Die Datenmanipulation findet mittels gespeicherter Prozeduren, Triggern und benutzerdefinierter Funktionen statt. Die Vorteile dieser Vorgehensweise liegen klar auf der Hand: Prozeduren und Funktionen sind Strukturierungsmittel für größere Anwendungen, nur vom DBMS (Datenbankmanagementsystem) abhängig und nicht von externen Programmiersprachen oder Betriebssystemumgebungen. Die Optimierung und Ausführung der Prozeduren liegt unter zentraler Kontrolle des Datenbankmanagementsystems. Gespeicherte Prozeduren und Funktionen sind hochperformant und in der Geschwindigkeit nicht zu übertreffen. [26]

Trigger sind immer an eine bestimmte Tabelle gebunden. Sie werden implizit von der Datenbank im Anschluss an INSERT-, UPDATE- oder DELETE- Anweisungen aufgerufen. Trigger können mit Events in der allgemeinen objektorientierten Programmierung vergli-

chen werden. Sie sind in der Lage, bei bestimmten Zustandsänderungen einer Tabelle oder Feldern einer Tabelle Ereignisse auszulösen. Die Syntax von Triggern gleicht der Prozeduren- oder Funktionensyntax. Trigger sind prädestiniert, Plausibilitäts- und Integritätsprüfungen vorzunehmen und/oder relevante Tabellen mit zu aktualisieren. [27]

Der Datenaustausch zwischen Datenbank und Anwendung erfolgt über eine entsprechende Klasse, die speziell für Datenkommunikation optimiert ist und im Verlauf dieser Arbeit noch eingehend beschrieben wird.

Im Folgenden werden wichtige Begriffe und allgemeingültigen Definitionen erläutert [25].

Normalisierung: Überführung komplizierter Tabellen in einfache Beziehungen durch Aufteilung der Attribute einer Tabelle auf mehrere Tabellen. Ziel sind stabile und flexible Datenstrukturen, die bei Erweiterungen möglichst wenig geändert werden müssen. Außerdem dürfen durch Änderung der Daten keine Anomalien entstehen.

Inkonsistenzen: Die Dateninkonsistenz ist eine Widersprüchlichkeit, die aus den Informationen der Datenbank resultiert (eine PLZ, der zwei Orte zugeordnet sind).

Redundanzen: Redundanzen sind wiederholte Speicherungen der gleichen Information in der Datenbank.

Integrität: Unsinnige und widersprüchliche Datenwerte müssen erkannt und abgewiesen werden. Überprüfung und Gewährleistung der Gültigkeit bestimmter Regeln, die darauf zielen, dass die Datensätze der eingehenden Relationen zueinander passen, müssen gegeben sein. Wird z.B. auf eine Postleitzahl in der Tabelle ein Stadtteil referenziert, muss gewährleistet sein, dass dieser Wert dort auch existiert.

1. Normalform ENF: Eine Relation ist in der Ersten Normalform, wenn jeder Attributwert atomar ist. D.h., die Tabelle muss zweidimensional sein; jede Zeile enthält Daten, die zu einem Objekt gehören; jede Spalte ist von derselben Art (Zeile 1 – Spalte 1 ist Name und Zeile 2 – Spalte 1 ist Name); Spalten haben eindeutige Namen; Zeilen sind eindeutig. So ist z.B. der Attributwert (Max Mustermann, Waldweg 12, 10000 Berlin) nicht atomar, da er eine vollständige Adresse enthält, die in mehrere Attribute aufgeteilt werden kann. Die Überführung in die ENF bedeutet Splitten des Attributwertes Adresse in mehrere Attribute, wie Vorname, Name, Strasse, PLZ, Ort.

2. Normalform ZNF: Eine Relation ist in der Zweiten Normalform, wenn sie in der ENF ist und jedes Nicht-Schlüsselattribut von jedem Schlüsselkandidaten vollständig funktional abhängig ist. Eine Tabelle der ENF muss einen eindeutigen Primärschlüssel besitzen und alle weiteren Attribute müssen von diesem abhängig sein. Eine Tabelle mit einem Primärschlüssel und einem Nicht-Schlüssel-Attribut ist also automatisch in der 2NF.

3. Normalform DNF: Eine Relation ist in der Dritten Normalform, wenn sie in der Zweiten Normalform ist und jedes Nicht-Schlüssel-Attribut von keinem Schlüsselkandidaten transitiv abhängig ist. Dieser Zustand wäre erreicht, wenn die Attributwerte Ort und PLZ in eine separate Tabelle ausgelagert werden und über ein Schlüsselfeld in Beziehung stehen. Vorname, Name, Strasse und PLZID beschreiben die erste Tabelle und PLZID, PLZ und Ort die zweite Tabelle, der Attributwert PLZID dient als Schlüsselfeld.

Die höheren Normalformen (4NF, 5NF,…) werden selten benötigt und deshalb an dieser Stelle nicht weiter erörtert.

4.2 Allgemeine Entwicklung im Bereich des Software-Engineering

Mit den ständig komplexer gewordenen Softwaresystemen hat sich eine neue wissenschaftliche Disziplin entwickelt, welche sich thematisch mit den Problemstellungen im Umfeld der Softwareentwicklung beschäftigt. Das Softwareengineering ist per Definition „die Anwendung eines systematischen, disziplinierten, quantifizierbaren Ansatzes auf die Entwicklung, Betrieb und Wartung von Software, d.h. die Anwendung der Ingenieurskunst auf Software." [IEEE „Standard Glossary of Software Engineering Terminology", 1990]

Im Laufe der Zeit haben sich verschiedene Ansätze und Vorgehensmodelle entwickelt, welche die Grundlage für jede Entwicklung neuer Software darstellen. Dabei ist eine ständige Weiterentwicklung in Richtung auf eine Standardisierung von Software zu beobachten. Mit COM (Component Object Model) wurde beispielsweise die Theorie der objektorientierten Softwareentwicklung weiterentwickelt, und die Möglichkeit geschaffen, ein komplexes Softwaresystem durch die Kombination von unabhängigen in sich abgeschlossenen Komponenten, nach dem Baukastenprinzip zu erstellen. Relativ schnell traten jedoch auch die Schwachstellen dieser Architektur zu Tage, so dass sich die führenden Anbieter von Softwareentwicklungswerkzeugen wie IBM, Microsoft und Sun, um einen gemeinsamen Standard bemühten. Als Ergebnis dieser Überlegungen wurde von Microsoft der .NET-Standard[13] veröffentlicht. Dabei werden mit .NET im Wesentlichen die folgenden Ziele verfolgt:

- Plattformunabhängigkeit
- Einheitliche Typsysteme
- Komplette Netzwerkunterstützung

[13] .NET: sprich: „Dot-Net"

- Einfacher XML[14]-Umgang
- Überschaubare Bibliotheken
- Komfortable Entwicklung internetbasierender Anwendungen

Da die .NET-Technologie für die Entwicklung des Konzeptes verwendet wird, werden in den folgenden Kapiteln die grundlegenden Eigenschaften und Mechanismen vorgestellt, deren Sachverhalte aus verschiedenen Entwicklerdokumentationen entnommen wurden.

4.3 DAS .NET FRAMEWORK

Das zentrale Element der .NET-Technologie ist das so genannte .NET Framework. Dabei handelt es sich um eine Laufzeitumgebung, welche eine Ausführung von Programmen, auch in einer Netzwerkumgebung, verwaltet. Sie vereinfacht und beschleunigt nicht nur die Entwicklung, sondern trägt dazu bei, die bisher häufigsten Programmfehler zu minimieren, wie beispielsweise leere Objektverweise und Typenunverträglichkeiten. In den konventionellen Programmiermethoden war oft der Aufwand für Typkonvertierungen und Referenzzugriffe höher als für die eigentliche Aufgabe des Codes. Ein weiteres Ziel des .NET Framework ist es, einen reibungslosen Betrieb und eine nahtlose Integration vorhandener Anwendungen zu ermöglichen.

Im Gegensatz zu interpretierenden oder skriptbasierenden Sprachen ist die Performance der erstellten Programme sehr hoch. Unterschiede zu maschinennah programmierten Codes sind nur messtechnisch nachweisbar und für den Benutzer selbst nicht erkennbar. Die Programmierumgebung ist für die verschiedensten Anwendungsarten, wie beispielsweise Services, Konsolen-, Windows- und Webbasierende Anwendungen, optimal ausgelegt. Codes, die aus dem .NET Framework stammen, sind untereinander voll integrativ, d.h. Klassenbibliotheken unterschiedlicher Programmiersprachen können in einem Projekt problemlos kombiniert werden.

Die Plattformunabhängigkeit des .NET Frameworks wird am deutlichsten am Beispiel der Webservices veranschaulicht. Ein Webservice stellt im Inter- oder Intranet Dienste für externe Aufrufer bereit. Dieser Dienst basiert vollständig auf offenen Standards, wie HTTP, XML, SOAP/XMLP, TCP/IP oder SMTP. Dadurch wird sichergestellt, dass Client-Applikationen von beliebigen Hardware- oder Betriebssystemplattformen aus die Dienste eines Web-Service-Servers in Anspruch nehmen können. So kann ein Webservice-Entwickler unabhängig vom Webanwendungs-Entwickler programmieren.

[14] Extensible Markup Language

Die Hauptbestandteile des .NET Frameworks sind die Common Language Runtime (CLR) und die .NET Framework-Klassenbibliotheken (Abbildung 4-1).

Die Common Language Runtime kann als Codevermittler und –verwalter zur Ausführungszeit beschrieben werden. [07] Diese Laufzeitumgebung beinhaltet eine große Anzahl von Funktionen, um insgesamt die Sicherheit und Robustheit eines Computerprogramms sicher zu stellen. Eine zentrale Eigenschaft der CLR ist die Unabhängigkeit von einer speziellen verwendeten Programmiersprache. So können neben den in .NET-Framework enthaltenen Programmiersprachen, wie C#[15], Visual Basic oder Java, alle anderen Sprachen verwendet werden, für die ein Compiler zur Verfügung steht. [08] Diese Unabhängigkeit wird dadurch erreicht, dass wesentliche Aufgaben von der Ebene der Programmiersprache auf die Ebene der CLR angehoben werden. Ein typisches Beispiel dafür ist die Typsicherheit. Während bei der herkömmlichen Programmierung die Konvertierung von Datentypen von einer Programmiersprache in eine andere Programmiersprache problematisch ist und stets eine potentielle Fehlerquelle darstellt, wird dieses Problem bei .NET grundsätzlich dadurch vermieden, dass die Typsicherheit an zentraler Stelle der CLR angesiedelt ist.

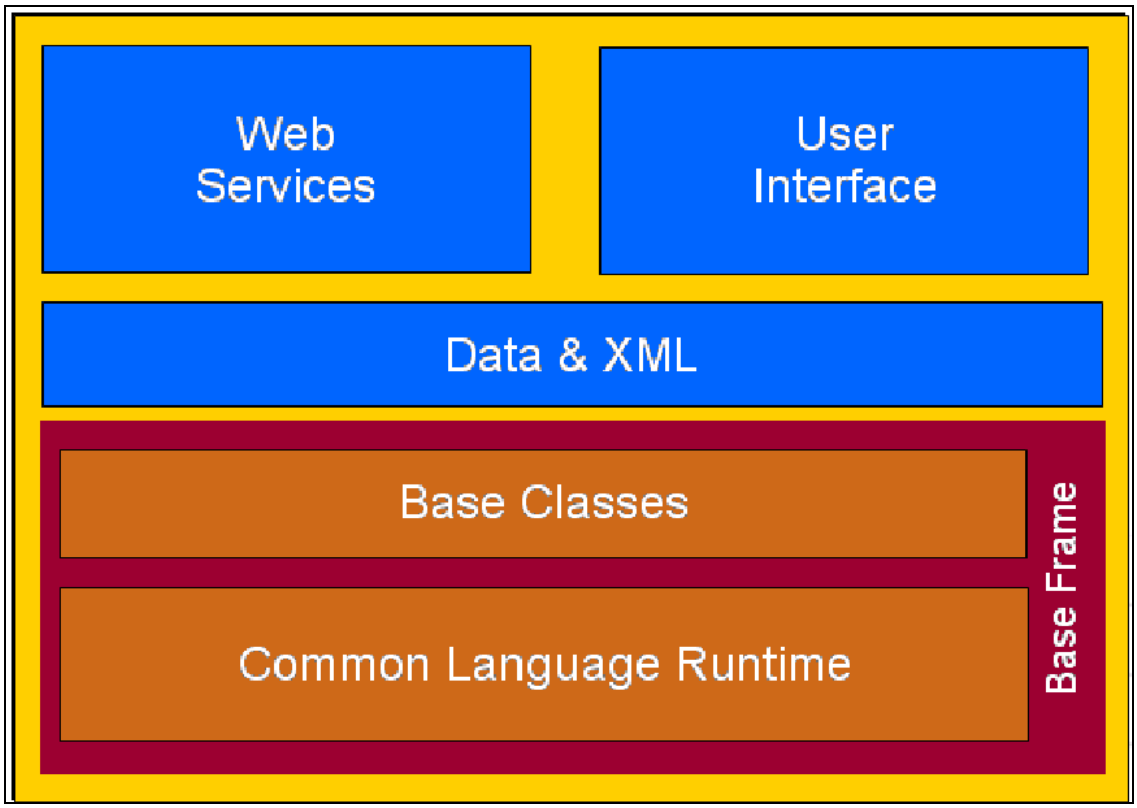

Abbildung 4-1 Architektur des .NET-Framework

[15] C#: sprich: „C-Sharp"

Das .NET Framework kennt grundsätzlich zwei Arten von Code. Der verwaltete Code (Managed Code) wird innerhalb der CLR ausgeführt und überwacht. Daneben wird auch nicht verwalteter Code (Unmanaged Code) unterstützt, der wie bei herkömmlichen Programmiersprachen unabhängig von einer Laufzeitumgebung abläuft. [09] Das .NET Framework bevorzugt den verwalteten Code. Da jedoch einige spezielle Programmarten, wie Compiler oder Hardwaretreiber, direkt ausführbar sein müssen, wird die Verwendung von nicht verwaltetem Code auch unterstützt. Weil diese Sonderfälle im Rahmen der hier relevanten Problemstellung keine Bedeutung haben, wird in der Folge nur der verwaltete Code berücksichtigt. Dabei wird der Quelltext eines Programms zunächst in die Intermediate Language (IL) übersetzt. Zur Ausführungszeit wird der IL-Code durch einen so genannten Just-In-Time-Compiler (JIT) in ausführbaren Binärcode überführt. Bei dieser Übersetzung erfolgt durch den JIT-Compiler ebenfalls die Optimierung für die Zielplattform. [10] Dieses Verfahren gewährleistet die Unabhängigkeit der erstellten Programme von der Hardware- und Betriebssystemumgebung der Zielplattform. Das .NET Framework beinhaltet JIT-Compiler für alle gängigen Betriebssysteme. Auch die Tatsache, dass Quelltexte unterschiedlicher Programmiersprachen zu fast identischen IL-Codes führen, ist ein wichtiger Bestandteil der Portabilität.

Eine detaillierte Beschreibung aller Eigenschaften der CLR kann der einschlägigen Fachliteratur entnommen werden. An dieser Stelle sollen lediglich die herausragenden Eigenschaften vorgestellt werden, die im Kontext des zu entwickelnden Konzepts ausschlaggebend sind. Neben den bereits angesprochenen Mechanismen beinhaltet die CLR vor allem weitergehende Funktionen, um die Sicherheit ausgeführter Programme zu gewährleisten. Diese Sicherheit von .NET basiert auf zusätzlichen Merkmalen, wie Code Signing und Code Access Security. Durch das Code Signing wird sichergestellt, dass der von einer Softwareentwicklungsfirma gelieferte Code auch tatsächlich vom Framework ausgeführt wird. Bei der Funktion Code Access Security handelt es sich um eine Laufzeit-gebundene Engine, welche die Berechtigung des den Code eingebenden Benutzers, des Code-Urhebers und des Codes selbst prüft, und dann entscheidet, ob der Code ausgeführt werden darf oder nicht. [11] Dies bietet wesentlich mehr Sicherheit als der jetzige DOS/Win32-Standard, in dem jeder beliebige Benutzer Codes ausführen kann, welche Schlüsseldateien des Betriebssystems überschreiben und dieses damit lahm legen können.

Den zweiten wesentlichen Bestandteil neben der CLR bildet die Klassenbibliothek von .NET. Diese Bibliothek enthält Basisklassen, die von einem Anwendungsprogramm verwendet werden können. Der Umfang der zur Verfügung stehenden Klassen deckt den gesamten Problembereich der Programmierung ab. Es finden sich geeignete Klassen für die Gestaltung von Benutzeroberflächen oder den Zugriff auf Datenbanken. Bemerkenswert ist, dass auch im Bereich der Klassenbibliothek die Unabhängigkeit von einer spezifischen Programmiersprache konsequent umgesetzt ist. So greifen Programme, die in unter-

schiedlichen Programmiersprachen, wie C# oder VB.NET, geschrieben sind auf exakt dieselben Basisklassen des .NET Framework zu. [12]

Da im Rahmen des Konzeptes in hohem Maße auf die Prinzipien der .NET-Architektur zurückgegriffen wird, werden im Folgenden die charakteristischen Eigenschaften des .NET Frameworks kurz erläutert, die in diesem Projekt relevant sind. Diese Beschreibungen dienen zur Abrundung des Basiswissens, welches zum Verständnis des Konzeptes notwendig ist.

4.3.1 Datenkapselung durch Klassen

Um der objektorientierten Programmierung konsequent nachzugehen und deren Vorteile auf allen Architekturebenen zu nutzen, ist die Kapselung von Daten aus relationalen Datenquellen eine effiziente Methode, sehr leistungsfähige Objekte zu erstellen. Die Objekte werden so programmiert, dass die Tabellenfelder Eigenschaften und das Laden, Validieren und Speichern der Daten Methoden der Klasse darstellen. Nicht nur die einfache Handhabung von Objekten prädestiniert diese Technik, auch können in die datenbankbezogenen Klassen Funktionen und Prozeduren implementiert werden, die Logik in Steuerelementen ausschließen. Die gesamte notwendige Datenbankfunktionalität steckt in den Methoden Open oder Save, Regeln dagegen in Validate. [13]

4.3.2 XML als Datensprache

Binnen kurzer Zeit hat sich XML von einer Sonderspezifikation zu einer Standard-Datensprache entwickelt. XML steht für Extensible Markup Language. XML ist eine Sprache, die Daten und deren Struktur beschreibt. Diese Information liegt in Dokumentform vor und kann in realen oder virtuellen Speichermedien abgelegt werden (Datei oder Stream). XML ist derzeit weit verbreitet, wird in verschiedenen Businessanwendungen und zunehmend auch in allgemeinen Anwendungen bevorzugt. Die Sprache selbst ist textbasierend (HTML-ähnlich), leicht lesbar und plattformunabhängig. Die Einsatzgebiete sind unbegrenzt, ob als Konfigurationsdatei oder als temporärer Datencontainer. Selbst die Abbildung hierarchischer und verschachtelter Tabellen ist möglich (XML-Baum).

Das .NET Framework ist für den Umgang mit XML ausgelegt, mit wenigen Anweisungen werden XML-Streams gelesen oder geschrieben. Spezielle .NET Framework Bibliotheken beinhalten sämtliche XML-Funktionalitäten.

Ein korrektes XML-Dokument beginnt mit der Deklaration, in der die XML-Version angegeben ist, mit der der Inhalt übereinstimmt. Nach der Deklaration folgt zwingend das Stamm- bzw. Dokumentelement. Innerhalb des Dokumentelements befinden sich die

XML-Elemente, sie sind stets durch Start- und Endtags begrenzt. Die folgende Abbildung zeigt eine einfache XML-Datei.

```
<?xml version="1.0" standalone="yes" ?>
- <NewDataSet>
  - <Eigenschaften>
      <Name>StartUp Message</Name>
      <Message>message.txt</Message>
      <Text>Hello World1</Text>
    </Eigenschaften>
  - <Eigenschaften>
      <Name>Final Message</Name>
      <Message>Final.txt</Message>
      <Text>Hello World2</Text>
    </Eigenschaften>
</NewDataSet>
```

Abbildung 4-2 Aufbau einer XML-Datei

Die hier abgebildete Datei beschreibt die Tabelle „Eigenschaften" mit den Feldern „Name", „Message" und „Text", in der zwei Datensätze selektiert wurden. Das Stammelement heißt „NewDataSet", da die XML-Datei aus einem Dataset stammt und standardmäßig den Namen des Datasets übernimmt. [14]

4.3.3 Datasets als Objektmodell

Das Dataset ist eine speicherresidente Darstellung von Daten, es stellt ein konsistentes relationales Programmierungsmodell ungeachtet der Datenquelle bereit. Es kann mit mehreren und unterschiedlichen Datenquellen, mit XML-Daten oder zur Verwaltung lokaler Anwendungsdaten verwendet werden. Das Dataset stellt einen vollständigen Satz von Daten dar, einschließlich verwandter Tabellen, Einschränkungen und Beziehungen zwischen den Tabellen. Die Struktur eines Datasets kann mit der Struktur einer relationalen Datenbank verglichen werden. In der folgenden Abbildung sind die Struktur und einige Eigenschaften (P) und Methoden (M) eines Datasets dargestellt:

- AcceptChanges (M)
- DataSetName (P)
- Relations (P)
- Tables (P)
- Clone (M)
- ReadXML (M)
- WriteXML (M)

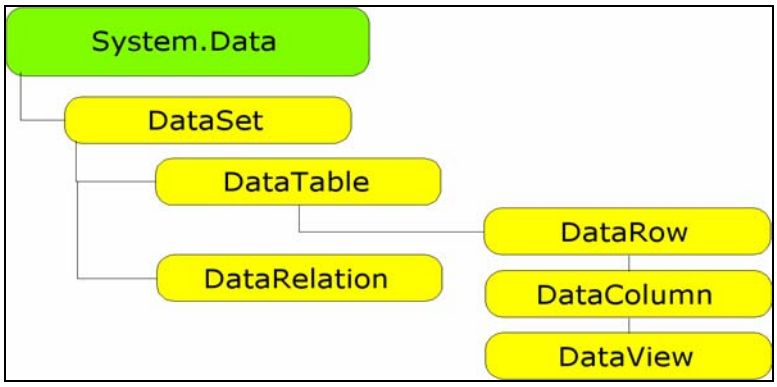

Abbildung 4-3 Auszug Dataset Struktur

Ein Dataset ist eine relationale Datenansicht, die als XML-Stream dargestellt werden kann, diese Eigenschaft macht das Dataset zum universalen Datencontainer. Mit den Methoden ReadXMLSchema und WriteXMLSchema können die Strukturen eines Datasets gelesen und geschrieben werden, d.h. ein Abbild eines relationalen Datenmodells. Es ist möglich, Klassen abzuleiten, mit denen der Zugriff auf Spalten, Zeilen oder Tabellen vereinfacht wird. Die Methoden ReadXML und WriteXML gestatten, XML-Streams zu lesen bzw. zu schreiben. [15] Diese Streams können in XML-Dateien gespeichert und von anderen Komponenten oder XML-fähigen Plattformen einlesen werden. Wie einfach ein Dataset mit Daten gefüllt und als XML-Datei abgelegt werden kann, veranschaulicht der folgende Code.

```
Dim m_con As New SqlClient.SqlConnection("Data Source=localhost;" & _
                                         "Integrated Security=SSPI;" & _
                                         "Initial Catalog=myCatalog")

Dim m_com As New SqlClient.SqlCommand("SELECT ObjektID, KlassenID FROM TB_KUNDEN", m_con)
m_com.CommandTimeout = 30

Dim m_adp As New SqlClient.SqlDataAdapter()
m_adp.SelectCommand = m_com
m_con.Open()

Dim m_ds As DataSet = New DataSet()
m_adp.Fill(m_ds, "Kunden")
m_ds.WriteXml("c:\temp\myKunden.xml")

m_con.Close()
```

Abbildung 4-4 Beispielcode XML-Datei schreiben

Noch einfacher ist das Lesen einer XML-Datei. Bei dieser Methode werden die Strukturen und Daten, die sich in der XML-Datei befinden, in das leere Dataset geschrieben.

```
Private Sub ReadXML(ByVal XMLFilename As String)
    Dim myDataSet As New DataSet()
    myDataSet.ReadXml(XMLFilename, XmlReadMode.Auto)
End Sub
```

Abbildung 4-5 Beispielcode XML-Datei lesen

4.3.4 Unterstützung verteilter Anwendungen (.NET-Remoting)

.NET Remoting stellt den Rahmen für die Kommunikation zwischen Objekten über Anwendungsdomänen hinaus dar. Bei der Übertragung über Channels (Kanäle) werden die Meldungen mit Hilfe von Formatierungsprogrammen codiert und decodiert. Es existieren Codierungen auf binärer Basis bei hohen Performanceansprüchen oder auf XML-Basis, wenn Interoperabilität gefordert wird. Der Sicherheitsaspekt wird über Hooks (Einstiegspunkte) realisiert, diese informieren über Zugriffe, bevor die Streams über die Channels transportiert werden. An dem folgenden einfachen Beispiel (siehe Abbildung) soll veranschaulicht werden, wie eine Remoteanwendung in der Praxis genutzt werden kann. Das Beispiel besteht aus drei Komponenten, dem Remoteserver, dem Remoteobjekt und dem Remoteclient und zeigt, wie Zeiten zwischen Computern synchronisiert werden.

```
Namespace RemotingSamplesObject

    Public Class myObject
        Inherits MarshalByRefObject

        Public Function GetServerTimeStr() As String
            Return Environment.MachineName + " aktuelle Uhrzeit:" & _
                                Date.Now.ToString("dd.MM.yyyy HH:mm")
        End Function

        Public Function GetServerTimeDate() As Date
            Return Date.Now
        End Function

    End Class

End Namespace
```

Abbildung 4-6 Beispielcode Remoteobjekt

Es existiert ein Zeitserver, der die genaue Uhrzeit im Netzwerk angeben soll. Die Clients, also beliebige Arbeitsstationen im Netz, synchronisieren automatisch ihre Uhrzeit mit der des Zeitservers. Das Datum wird zum einen als Type Date und zum anderen als Typ String zurückgegeben. [16]

```
Dim chan As TcpChannel = New TcpChannel(8085)
ChannelServices.RegisterChannel(chan)
Call RemotingConfiguration.RegisterWellKnownServiceType(GetType(RemotingSamplesObject.myObject), _
                                        "SayHello", WellKnownObjectMode.SingleCall)
```

Abbildung 4-7 Beispielcode Remoteserver

```
Dim chan As TcpChannel = New TcpChannel()
Call ChannelServices.RegisterChannel(chan)
Dim obj As Object = Activator.GetObject(GetType(RemotingSamplesObject.myObject), _
                    "tcp://10.0.0.14:8085/SayHello")

If Not obj Is Nothing Then
    Me.Text = obj.GetServerTimeStr
End If

Call ChannelServices.UnregisterChannel(chan)
```

Abbildung 4-8 Beispielcode Remoteclient

Es ist schnell erkennbar, dass mit wenig Programmcode .NET Remoting verwirklicht und Informationen sowie Daten über Anwendungsdomänengrenzen hinweg ausgetauscht werden können.

4.4 Ausgewählte Entwicklungsumgebung

Parallel zur Veröffentlichung des .NET Frameworks erschien die integrierte Entwicklungsumgebung[16] „Microsoft Visual Studio .NET®" zur Erstellung verschiedener .NET Projekte. Sie beinhaltet Vorlagen für Web-, Konsolen-, Windows- und ASP-Anwendungen sowie für Bibliotheken, Steuerelemente und verschiedene Dienste. Die Visual Studio® .NET IDE ist für das .NET Framework optimiert und erlaubt die Nutzung aller .NET-Features mit den folgenden Leistungen:

- Mit dem Visual Studio® .NET können Komponenten erstellt werden, die mit Hilfe von XML-basierenden Webservices auf jede beliebige Plattform zugreifen können.
- Es unterstützt die Entwicklung hoch leistungsfähiger Datenbankanwendungen dank neuer Datenzugriffstechnologien, wie ADO.NET.
- Applikationen für unterschiedliche Plattformen und mobile Geräte können einfach und schnell erstellt werden. Browserbasierende Entwicklungen oder speziell für Handheldgeräte gefertigte Anwendungen stellen kein Problem dar.

[16] engl.: Integrated Development Environment oder IDE

Die .NET-Entwicklungsumgebung stellt also das geeignete Werkzeug zur Erstellung von Programmen auf Basis des .NET Frameworks dar und soll deshalb für das hier angestrebte Konzept zum Einsatz kommen.

5 KONZEPT DER KOMPONENTEN-ENTWICKLUNG

5.1 ALLGEMEINES

Das hier vorliegende Konzept beschreibt sowohl die Architektur als auch die Vorgehensweise einer zu entwickelnden Individualsoftware unter der Berücksichtigung der eingangs aufgestellten Anforderungen. Die Softwarearchitektur, das Programmierparadigma und das Datenbankdesign werden so gestaltet, dass diese Komponenten als Vorlagen für weitere Projekte genutzt und somit die Implementierungszeiten für diese Folgeprojekte komprimiert werden können. Da das Konzept selbst nicht nur auf die Theorie bezieht, sondern unmittelbar auf die praktische Anwendung ausgerichtet ist, wird die Entwicklung projektbezogen vor Ort in einem kleinen mittelständischen Unternehmen durchgeführt, welches auch der spätere Erstanwender ist.

Wie die Analyse gezeigt hat, sind insbesondere die kleinen mittelständischen Betriebe sowohl bezüglich der vorhandenen Finanzkraft als auch durch die ihnen zur Verfügung stehenden Mitarbeiter nicht in der Lage, die vorhandenen Softwaresysteme für eine unternehmensweite Bearbeitung ihrer Geschäftsprozesse zu nutzen. Letzteres wird insbesondere durch die Tatsache verhindert, dass die zur Verfügung stehenden Softwaresysteme sehr starr und unflexibel sind, was wiederum von dem Anwender eine gewisse Anpassungsfähigkeit ihrer Geschäftsprozesse an das System verlangt. Aber gerade diese Anpassungsfähigkeit können diese Unternehmen auf Grund ihrer vorhandenen Mitarbeiterstruktur nicht leisten. Also muss hier ein Weg gefunden werden, der es ermöglicht, das zu entwickelnde Softwareprodukt überwiegend an die vorhandenen Geschäftsprozesse anzupassen und nicht umgekehrt. Des Weiteren ist das Konzept so gestaltet, dass einzelne erzielte Lösungen ohne großen Aufwand auf andere Unternehmen zu übertragen sind.

5.2 AUFBAU UND DURCHFÜHRUNG DES KONZEPTES

Deshalb wird die Entwicklung in einem so genannten Rapid Prototyping-Verfahren durchgeführt, das heißt, die Entwicklung wird modular ausgeführt und jedes Modul wird unmittelbar nach der Fertigstellung im produktiven Betrieb getestet. [17] Durch diese parallele Vorgehensweise können die eventuell auftretenden Mängel somit unmittelbar behoben und die Handhabbarkeit schon frühzeitig optimiert werden. Des Weiteren können durch diese Arbeitsweise die jeweils betroffenen Mitarbeiter bzw. die späteren Anwender sukzessive in die Software eingearbeitet werden, wodurch die Akzeptanz von vorneherein gegeben ist. Jedes einzelne Modul bzw. jede Komponente wird so allgemeingültig wie

möglich konzipiert, um die spätere Übertragbarkeit von vorneherein zu gewährleisten. Die Abbildung 5-1 zeigt die Struktur in softwaretechnischer Hinsicht für die gesamte Anwendung. Die Abbildung verdeutlicht die objektorientierte Software-Entwicklung der einzelnen Komponenten auf drei Ebenen. Ausgehend von der jeweiligen Anwendungsebene wird die Kommunikationsebene und die Datenbankebene mit ihren jeweils notwendigen Schnittstellen konzipiert. Somit ist jederzeit ein durchgängiger Datenfluss gewährleistet. Der hier eingesetzte softwaretechnische Lösungsweg wird später ausführlich vorgestellt.

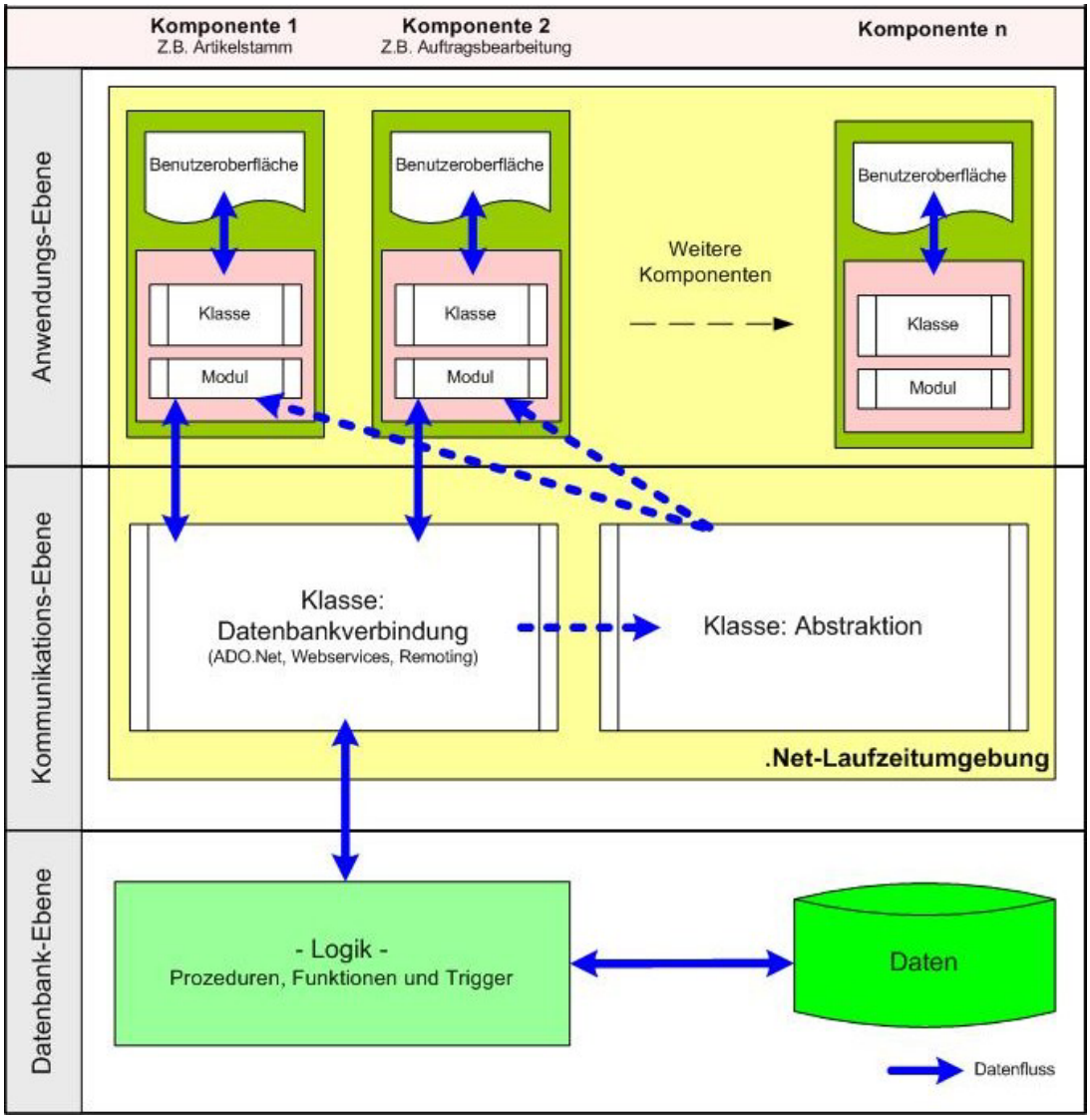

Abbildung 5-1 Softwaretechnische Architektur des Konzeptes

Sicherlich ist die Tatsache, die Entwicklung und die Einführung der Software während des Geschäftsalltages durchzuführen, nicht alltäglich, aber auf diese Weise stehen die neu entwickelten Komponenten immer sofort auf den Rechnern der Anwender zur Verfügung und der Anwender wird unmittelbar in die Entwicklung mit einbezogen. Damit ein derartiges Konzept effektiv koordiniert und mit minimalen Konflikten abgewickelt werden kann,

sind hier erhöhte Anforderungen an das Projektmanagement zu stellen. Deshalb muss vor Beginn der softwaretechnischen Entwicklung des Konzeptes zunächst die Reihenfolge des Ablaufs Abbildung 5-2 für die gesamte Entwicklung definiert werden.

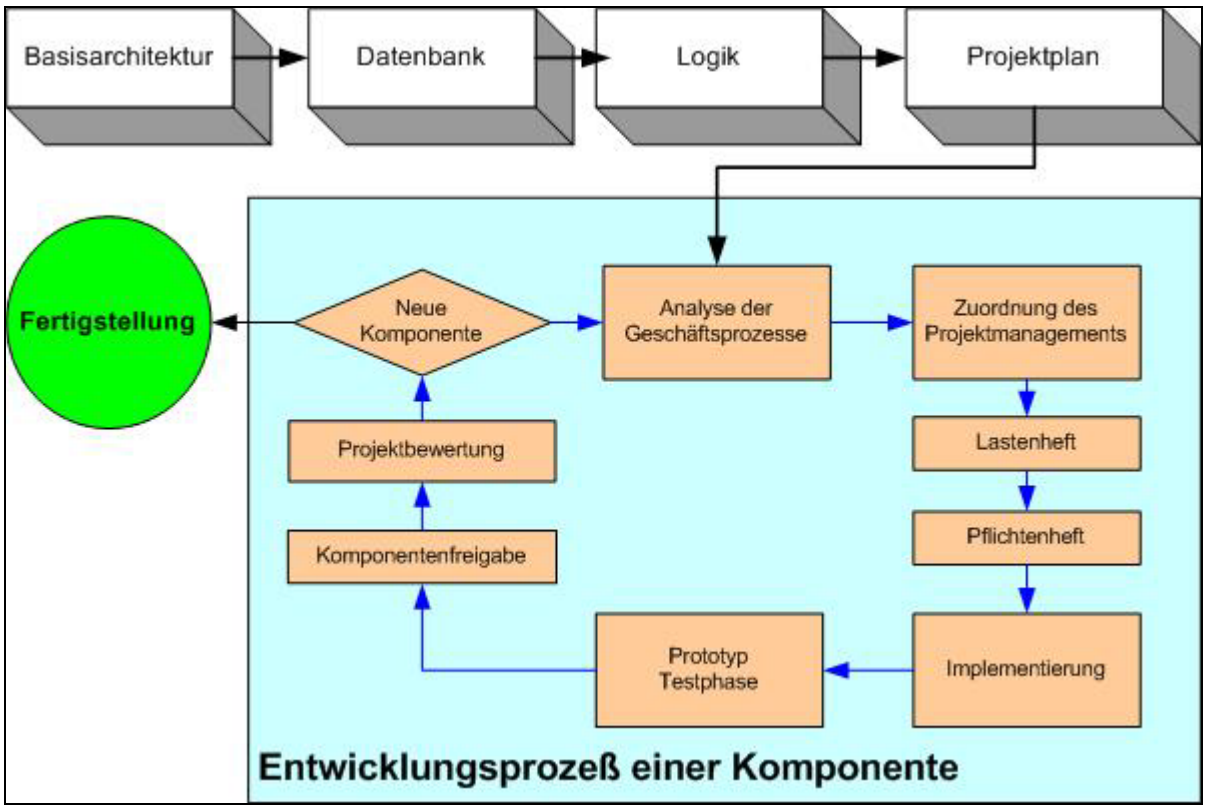

Abbildung 5-2 Ablauf des Konzeptes

In einem ersten Schritt sind die allgemeingültigen Grundlagen zu erarbeiten, die nicht für die spezifische Entwicklung einer einzelnen Komponente, sondern für die Entwicklung des gesamten Konzeptes gelten. Diese Grundlagen sind nur jeweils einmal zu erarbeiten, während der Prozess der Komponentenentwicklung (Punkt 6.) iterativ für jede einzelne gewünschte Komponente zu wiederholen ist. Im Einzelnen werden nun die folgenden Sachverhalte vorgestellt:

1. Das Prinzip der Online-Entwicklung (Kapitel 5.3)
2. Definition der grundlegenden Basisarchitektur (Kapitel 5.4)
3. Festlegung der Entwicklungs-Richtlinien für Datenbank und Logik (Kapitel 5.5)
4. Definition des spezifischen Projektplan (Kapitel 5.6)
5. Klärung der Strukturen des beteiligten Anwenderunternehmens (Kapitel 5.7)
6. Beschreibung des Entwicklungsprozess für eine Komponente (Kapitel 5.8)

- Analyse der Geschäftsprozesse
- Zuordnung/Anwendung des Projektmanagements
- Lastenheft (Was wofür benötigt wird?)
- Pflichtenheft (Wie womit die Aufgabe gelöst wird?)
- Implementierung
- Prototyp / Testphase
- Freigabe der Komponente
- Projektbewertung des Teil-Projektes (Projekttest)

5.3 Die „Online-Entwicklung" und ihre Randbedingungen

Wie schon vorab definiert, geschieht die Programmierung der Software vor Ort beim Kunden und parallel zum Tagesgeschäft. Diese Arbeitsweise wird auch als so genannte „Online-Entwicklung" bezeichnet. Die Vor- und Nachteile dieser Methode lassen sich am besten in einer direkten Gegenüberstellung verdeutlichen.

Nachteile:

- Entwicklungsfehler können das Tagesgeschäft beeinflussen
- Nicht jeder Anwender oder Programmierer kann mit Kritik umgehen
- Häufige Stress-Situationen beim Entwickeln
- Vom Anwender wird neben seiner Tätigkeit zusätzliches Lernen gefordert
- Fehlende Motivation behindert die Entwicklung
- Werden nicht alle Ideen verwirklicht, ist der Anwender enttäuscht
- Unvorhersehbare Arbeitszeiten für Programmierer
- Hohe Kompetenz der Programmierer (alle Architekturebenen)

Vorteile:

- Wirklich kundenorientierte Systemsoftware
- Programmierer lernen endlich die Realität kennen
- Anwender gewinnen Verständnis für Software und deren Probleme
- Benutzer können indirekt mit entwickeln
- Äußerst effektive Fehlererkennung, da Anwender integriert ist
- Effektiverer Umgang mit der Software, da sie mit dem Betrieb wächst
- Anwender erhält eine partielle Schulung nebenbei
- Offene Frage können sofort gestellt werden
- Schnellstmögliche Fehlerbeseitigung wird garantiert
- Software ist immer aktuell

- Einbringung neuer Ideen durch gemeinsame Projektarbeit, dadurch ständige Optimierung der Geschäftsprozesse
- Der wachsende Datenbestand wird vom Entwickler beobachtet, dadurch werden Eingabefehler bzw. unsinnige Eingaben früh erkannt, dies dient zur allgemeinen Prävention
- Der Kunde hat ständig kompetentes IT-Personal vor Ort

Und diese sind bei weitem nicht alle Vorzüge, weil manche Vorteile eher unauffällig oder nicht ersichtlich sind und somit nicht wahrgenommen werden. Es ist also keine Frage, dass gerade für die Entwicklung einer Individualsoftware die ständige Präsenz beim Kunden von Vorteil ist.

Um eine Entwicklung durchführen zu können, die das Merkmal des parallelen Einsatzes entwickelter Komponenten aufweist, müssen die entsprechenden betrieblichen Bereiche auf Kompatibilität überprüft werden. Bereiche, die Schnittstellen aufweisen, wie Kunden oder Lieferanten, sind besonders zu beachten. Hier auftretende Komplikationen sind nachhaltig und extrem sensibel, da sie den Geschäftsprozess direkt beeinflussen. Wird z.B. im Bereich der Planung von Fertigungsaufträgen oder im Bereich der Betriebsdatenerfassung entwickelt, haben auftretende Komplikationen häufig tief greifende Auswirkung auf den Produktionsprozess selbst, wie die nachfolgenden Fallbeispiele zeigen.

Fallbeispiel 1:

Ein im Einsatz befindliches Terminal zum Setzen der Arbeitsgangstati (an- und abmelden, unterbrechen, wieder aufnehmen von AG, Rüsten an- und abmelden) weist einen empfindlichen Fehler auf, sei er software- oder hardwaretechnisch, es stehen Daten, die zur weiteren Planung und Belegung notwendig sind, nicht mehr zur Verfügung. Der Arbeitsvorbereitung fehlt somit der Überblick über den aktuellen Fertigungsstand, der nur mit enormem zusätzlichem Aufwand auszugleichen ist.

Fallbeispiel 2:

Die Werkzeug- und/oder Materialausgabe läuft über eine Datenerfassungsmethode (Barcode, Magnetstreifen, Funk), die Datenübertragung zur Datenbank bleibt jedoch durch einen Fehler aus. Um nicht eine Differenz zwischen Realbestand und Datenbankbestand zu erhalten, müssen die Daten manuell über die Anwendung in die Datenbank eintragen werden, dies ist zwar möglich, aber der zeitliche Aufwand größer. Die Summe der Wartezeiten wirkt sich sehr negativ auf den Fertigungsablauf aus.

Anhand dieser Fallbeispiele wird erkennbar, wie wichtig es ist, vor Beginn der „Online-Entwicklung" eine genaue Analyse der betroffenen Unternehmensbereiche vorzunehmen.

Zur Vermeidung doppelter Arbeit können als Grundlage die Erkenntnisse aus dem Kapitel 5.7 mit herangezogen werden.

Die Probleme mit neuen Komponenten, die bei der "Online-Entwicklung" entstehen können, werden durch die Möglichkeit der sofortigen Fehlereliminierung ausgeglichen. Der Kontakt mit dem Programmierer kann sofort hergestellt und der Fehler weitergegeben werden. Sind die Benutzer mit dieser Vorgehensweise vertraut, steigert das nicht nur die Qualität der Software, sondern auch zusätzlich die Motivation der Anwender. Sie können direkt an der Entwicklung "ihrer" Software teilnehmen. Die Vorschläge für Verbesserungen oder Vereinfachungen und die Fehlererkennung durch die Anwender sind unabdingbar für die Entwicklung. Da zwischen der Komponentenentwicklung und ihrem Einsatz kein Zeitversatz existiert, ist der Programmierer mit dem Thema noch intensiv vertraut und kann schnelle Korrekturen vornehmen und anschließend eine erneute Kompilierung durchführen. Somit ist der Ring „Entwicklung und Einsatz" geschlossen.

Die Erfahrungen haben gezeigt, dass anfänglich diese Schleife mehrmals durchlaufen wird, aber die Anzahl stetig abnimmt und letztlich erstaunlich wenige Durchläufe (ca. 1-4, je nach Schwierigkeitsgrad der Komponenten können auch mehr Durchläufe nötig sein) benötigt werden. In dem Blockdiagramm gemäß Abbildung 5-3 sind die einzelnen Entwicklungsetappen schematisch dargestellt.

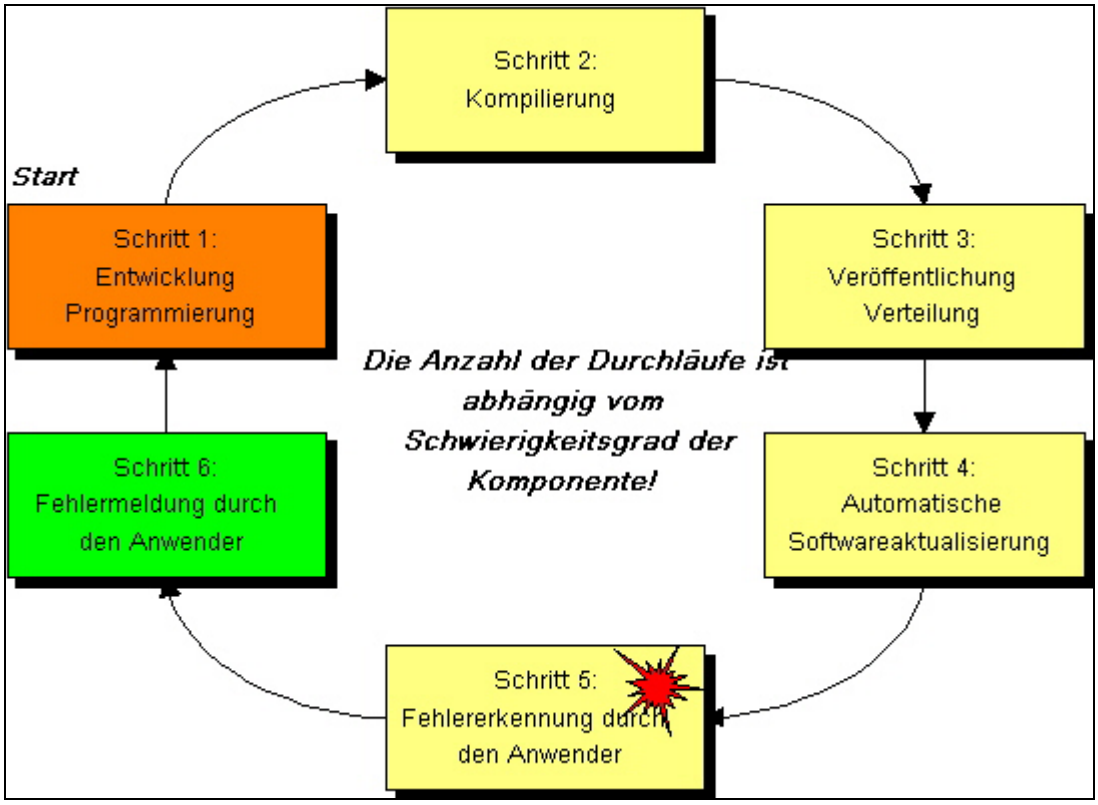

Abbildung 5-3 Kreislauf der "Online-Entwicklung"

Der technische Ablauf von Schritt 2 zu Schritt 4 kann variieren, da es unterschiedliche Möglichkeiten gibt, Software zu veröffentlichen und zu aktualisieren. Die einfachste Variante ist das Kopieren der Release-Komponenten in einen Netzwerkordner, in dem die Benutzer nur Leserechte besitzen. Von diesem Pfad aus kann via Loginscript eine XCopy-Routine angestoßen werden, welche die Komponenten aktualisiert oder ergänzt. Eine komfortablere Alternative ist durch das automatische Update für .NET-Komponenten realisierbar. Hier bemerkt die Anwendung selbst das Vorhandensein einer aktualisierten Komponente und fragt den Benutzer, ob diese jetzt sofort aktualisiert werden soll oder selbsttätig beim Neustart des Programms. Sicher gibt es noch weitere gute Lösungen, meist besitzen diese aber Randbedingungen, die nicht immer erfüllt werden können (z.B. ActiveDirectory[17]).

Die Anwendung der "Online-Entwicklung" bei einigen Komponenten, die sensible Schnittstellen aufweisen, ist im Vorfeld zu untersuchen. Bei einer kompromisslosen Anwendung dieser Methode werden Konfliktsituationen auftreten, die enorme Störungen des Geschäftsprozesses verursachen. So müssen im Vorfeld Berührungspunkte und Gefahrenstellen, die durch die "Online-Entwicklung" entstehen, identifiziert und klassifiziert werden. Diese Vorbetrachtungen ermöglichen dann präventive Maßnahmen sowie prophylaktisches Verhalten der Anwender und Programmierer während der Entwicklung. Beeinträchtigungen des jeweiligen Geschäftsprozesses werden somit voraussehbar oder sogar vermeidbar.

5.4 Das Software-Architekturmodell

Das Fundament des Architekturmodells ist die komponentenorientierte Entwicklung. Das .NET Framework ist nicht nur objektorientiert, sondern auch komponentenorientiert und damit von Grund auf geeignet, innovative Software zu erstellen. Im Mittelpunkt des Komponentenkonzepts stehen die so genannten Assemblies. Eine Komponente stellt sich in diesem Konzept überwiegend als Assembly dar und ist mit ihr gleichzusetzen, somit gilt: **Assembly = Komponente**.

Der Grund für die Benutzung von Assemblies bei einer komponentenorientierten Entwicklung ist die Vereinfachung der Weitergabe und Minimierung der Versionsprobleme. Außerdem ist die zentrale Instanziierung durch eine so genannte Factory-Klasse von entscheidendem Vorteil. Bisher waren Versions- und Weitergabeprobleme in Verbindung mit der Bibliothekenregistrierung durch die Verbreitung von Systemen auf Komponentenbasis

[17] Verzeichnisdienst, Weiterentwicklung der Windows-Domäne

bekannt. Sowohl Endbenutzern als auch Entwicklern bereitete diese Angelegenheit viel Ärger. Eine Vielzahl von Endbenutzern hat die unangenehme Erfahrung gemacht, dass eine vorhandene Anwendung nach dem Installieren einer neuen Anwendung plötzlich nicht mehr funktioniert. Um diese Situation zu umgehen, verbrachten viele Entwickler Stunden, alle relevanten Registrierungseinträge konsistent zu halten. Im .NET Framework wird eine Fülle von Weitergabeproblemen durch die Verwendung von Assemblies behoben. Da Assemblies selbstbeschreibende, von Registrierungseinträgen unabhängige Komponenten sind, ermöglichen sie das Installieren von Anwendungen ohne unerwünschte Auswirkungen. Assemblies vereinfachen außerdem das Deinstallieren und Replizieren von Anwendungen. [18]

5.4.1 Das Architekturmodell der Komponenten

In diesem Konzept besteht ein Assembly vornehmlich aus einer Klasse, einem Interface (Schnittstelle) und aus einer Benutzeroberfläche, auch Userinterface genannt. Die Klasse hat die Datenkapselung zur Aufgabe, das Interface stellt die deklarierten Eigenschaften und Methoden bereit und die Benutzeroberfläche (Benutzercontrol) ist die Schnittstelle zum Anwender für die Dateneingabe bzw. Datendarstellung. Sind zusätzliche Features im Assembly erforderlich, die laut Programmierparadigma weder in die Klasse noch ins Control gehören, werden extra Klassen und/oder Module dem Assembly zugeordnet. Ein Beispiel ist der Zugriff auf Hardwareschnittstellen, wie COM oder LPT, die unter Umständen für die Komponente unerlässlich sind. In dem Fall werden separate Klassen oder Module erstellt, welche die Kommunikation zur Hardwareschnittstelle realisieren. Im Control oder der Klasse des Assembly werden dann diese separaten Klassen oder Module genutzt.

Der Aufbau einer Assembly-Klasse basiert auf der Vorgabe, wie Daten in Objekte zu kapseln sind. Erweiterungen und Modifikationen innerhalb der Klasse machen sie zum gewünschten Assembly konform. Methoden, Eigenschaften, Deklarationen und Validierung bilden die Grundbausteine der Klasse, wobei die Methoden und Eigenschaften um klassenabhängige Methoden und Eigenschaften erweitert werden sowie ergänzend eine Prüfung auf korrekte Initialisierung erfolgt. Diese Klasse repräsentiert die Daten aus der Datenbank. Nachdem die Daten über die Kommunikationsebene geladen werden, stehen sie zur Manipulation zur Verfügung und nicht mehr in Verbindung mit der Datenbank. Innerhalb der Klasse werden die geänderten Informationen geprüft und zum Speichern bereitgestellt. Erst bei einer Aufforderung zum Speichern werden die Informationen in der Datenbank aktualisiert. Die Eigenschaften und Methoden stehen dem Benutzercontrol und dem Interface zur Verfügung.

Die Anwendung der Polymorphie ist für die Komponenten zwingend, da die Factory-Klasse jedes Assembly deklariert, instanziiert und Methoden aus ihm aufruft. Grundsätzlich leistet die Eigenschaft des Polymorphismus folgendes: Es können in unterschiedlichen Klassen gleichnamige Methodenbezeichner (Methodennamen) verwendet werden. Die gleichnamigen Methoden können jedoch unterschiedliche Funktionalitäten besitzen, obwohl sie in gleicher Weise aufgerufen werden. [19] Im Allgemeinen beinhalten gleichnamige Methoden auch den gleichen Inhalt, zumindest in diesem Konzept. Wie der Entwurf einer Klasse aussieht, zeigt die folgende Abbildung. Dieses Schema gilt für jede Klasse, welche die Datenkapselung zur Aufgabe hat. Um den Arbeitsaufwand bei der Neuerstellung von Assemblies zu minimieren, muss eine Art Template mit Standardeigenschaften und Methoden existieren.

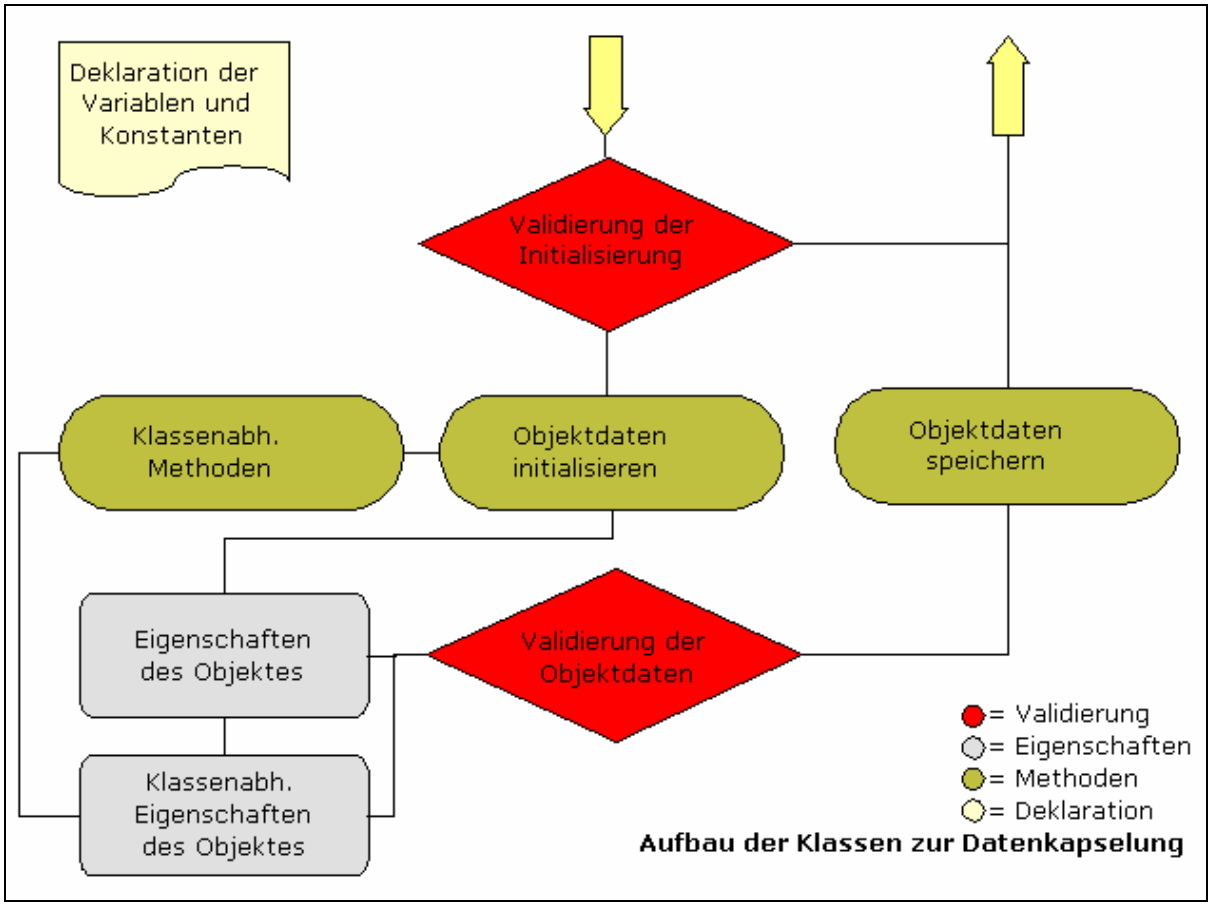

Abbildung 5-4 Klassenentwurf

Die Zugriffe von außen auf die Klasse lassen sich über das Interface separat organisieren. Interfaces sind Verweistypen, die von anderen Typen implementiert werden, um die Unterstützung bestimmter Operationen zu gewährleisten. Eine Schnittstelle kann als Vertrag zwischen einer Klasse oder einer Struktur und dem Klient beschrieben werden und sie können Methoden, Eigenschaften und Ereignisse als Member enthalten. [20]

```
Interface IExample

    Event E()

    Sub F(ByVal Value As Integer)

    Property P() As String

End Interface
```

Abbildung 5-5 Aufbau des Interfaces

Mit geringem Aufwand lassen sich Interfaces implementieren. Auch wenn zum Zeitpunkt der Erstellung des Assembly noch nicht klar ist, in welcher Situation das Interface des Assembly genutzt werden soll, gerechtfertigt ist die Implementierung durch den geringen Aufwand auf jeden Fall. In der Abbildung 5-5 wird veranschaulicht, wie ein Interface die Member Methode (F), Event (E) und Property (P) aufweist.

Für das Benutzercontrol des Assembly gibt es keine Richtlinien oder Vorgaben. Es kann webbasierend oder windowsbasierend realisiert werden. Ein Beispiel für die windowsbasierende Nutzung zeigt die Abbildung 5-6.

```
Public Class Class3
    Dim Parameter, Text1, Text2

    Private Sub test(ByVal c_Class As Class1)
        Dim m_Object As IExtInterface.MyInterface
        m_Object = c_Class
        m_Object.Init(Parameter)
        With m_Object
            Text1.Text = .wert2
            Text2.Text = .wert2
        End With
    End Sub

End Class
```

Abbildung 5-6 Interface-Einbindung

Die erste Programmzeile in der Prozedur deklariert die Objektvariable, danach wird dem Objekt die instanziierte Klasse der gekapselten Daten zugewiesen. Anschließend erfolgt der Aufruf der Methode „Init" entsprechend der Parameterliste und füllt das Objekt mit Daten. Die Eigenschaftswerte des Objektes stehen jetzt den Elementen zur Verfügung. Über die einfache Handhabung hinaus bestehen die folgenden weiteren Vorteile:

- Dieses Baukastensystem wertet die Komponentenentwicklung auf.
- Besseres Design der Assemblies.
- Verwirklichung der Polymorphie, d.h. unterschiedliche Klassen können die gleiche Schnittstelle implementieren.

- Kompatibilitätsprobleme werden reduziert, da für Schnittstellen verbesserte Implementierungen möglich sind.
- Der vorhandene Code ist bei Modifikationen nicht gefährdet.
- Jederzeit können neue Features hinzugefügt werden, indem weitere Schnittstellen und Implementierungen entwickelt werden.

Die Summe aus Klasse, Interface und Benutzerschnittstelle bildet somit das Assembly. Der schematische Aufbau eines Assembly ist in der Abbildung 5-7 zusammengefasst dargestellt. Auf die Details der Kommunikationsebene wird im nächsten Abschnitt eingegangen.

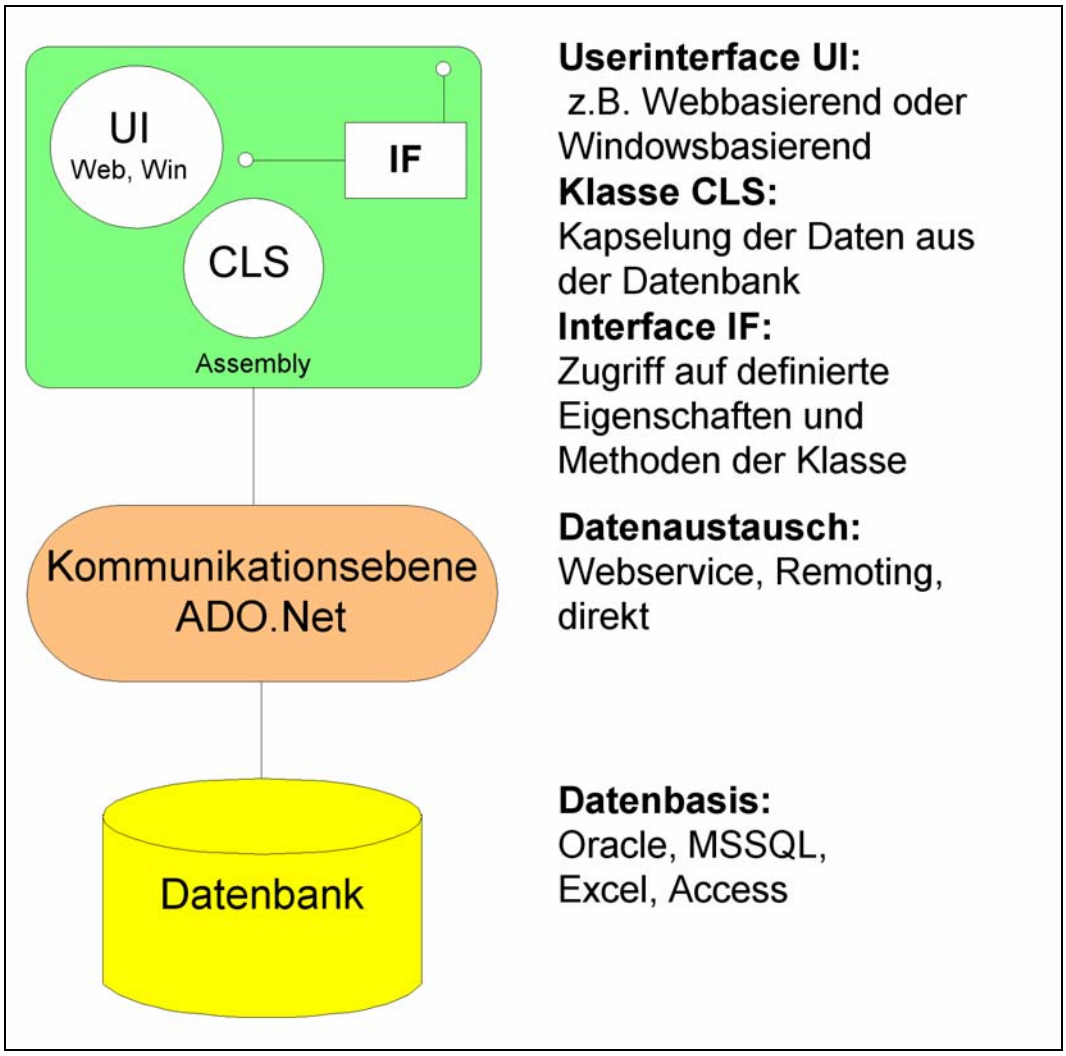

Abbildung 5-7 Assembly schematisch

Zusammenfassend sollen hier noch einmal die wichtigsten Merkmale zur komponentenorientierten Entwicklung wiedergegeben werden:

- Eigenschaften, Funktionen und Methoden die logisch zusammenhängen, werden in Assemblies (Verbund) verpackt.
- Assemblies existieren in Form einer EXE oder einer DLL.
- Assemblies sind wieder verwendbare Softwarekomponenten.
- Assemblies können von anderen Assemblies genutzt werden.
- EXE-Assemblies lassen sich als unabhängige Anwendung starten.
- EXE-Assemblies können Dienste bereitstellen.
- Welche Dateien zum Verbund gehören, wird durch ein Manifest bestimmt. Ein Manifest ist ein Teil der DLL oder EXE und stellt die Deklaration des Assembly dar.
- Mindestens eine Datei gehört zu einem Assembly.
- Anwendungen, die aus Assemblies bestehen, sind übersichtlicher und können leichter gewartet werden.

5.4.2 Gestaltung der Kommunikationsebene

Die Gestaltung der Kommunikationsebene hängt im Wesentlichen von dem geplanten Einsatzgebiet der Datenbankentwicklung ab. Die hier vorgesehene Lösung wird im Intranet (Firmennetzwerk) wegen ihrer Zuverlässigkeit und Schnelligkeit als direkte Datenbankverbindung genutzt. Wird später, aus welchen Gründen auch immer, ein Web- oder Remotedienst gefordert, muss lediglich diese Kommunikationsebene umgestaltet werden. Die Kommunikationsebene besteht nur aus zwei Klassen mit relativ wenig Code. Dieser Code beinhaltet die Parameteraufbereitung, setzt die Verbindungseigenschaften, füllt die Datasets und stellt bei Bedarf die Rückgabeparameter zur Verfügung. Die Struktur dieser Klassen ist identisch mit dem allgemeinen Aufbau einer Klasse, beinhaltet also Eigenschaften und Methoden.

Bei einem Einsatz von Web- oder Remotediensten bleibt die Funktionalität erhalten, muss aber in das entsprechende Layout des Dienstes integriert werden. Die Architekturen für diese Dienste wurden schon bei der Vorstellung des Frameworks kurz erläutert. In diesem Konzept soll die Konzentration auf der direkten Datenverbindung liegen. Wie schon erwähnt, besteht die Kommunikationsschicht aus zwei Klassen, der Datenbank- und der Abstrahierungsklasse. Die wesentliche Arbeit erledigt die Datenbankklasse mit der Datenzugriffsfunktion und dem Setzen der Verbindungseigenschaften. Die Abstrahierungsklasse hingegen dient nur zur vereinfachten Interpretation der Datenbankergebnisse. Dort werden statt der Datasets, die komplette Strukturen aufweisen, nur simple Auflistungen mit Hilfe eines Dictionary im Speicher gehalten. Es enthält Objektauflistungen mit einem Key, um auf diese zugreifen zu können. Der Einsatz des Dictionary ist eher selten

und nicht entscheidend für das Konzept. Im Folgenden ist eine vereinfachte Darstellung der Kommunikationsebene abgebildet

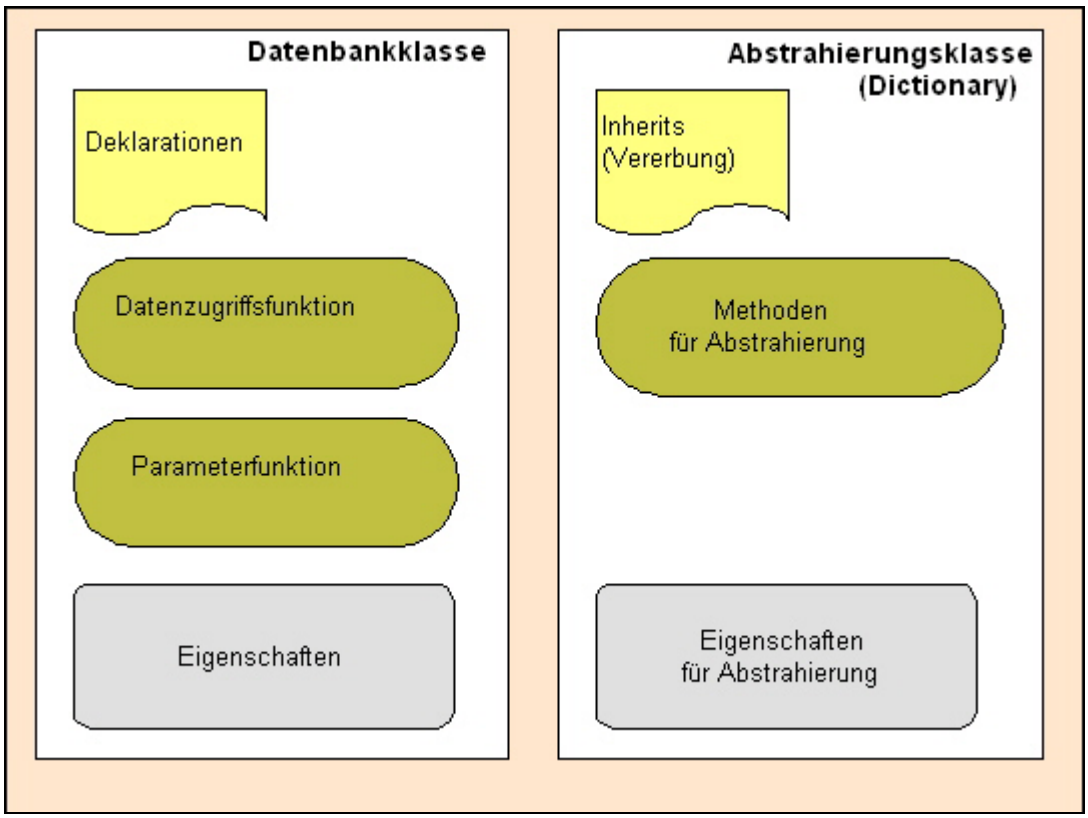

Abbildung 5-8 Kommunikationsebene schematisch

Wie das beschriebene Architekturmodell am konkreten Beispiel aussieht, schildert der Realisierungsabschnitt dieser Arbeit. Stellvertretend für alle Komponenten bzw. Assemblies wird die Umsetzung eines Assembly in dem Realisierungsabschnitt beschrieben. Analog zu diesem kann bei anderen Komponenten gleich vorgegangen werden. Ausnahmen bilden dabei die komponententypischen Features.

5.4.3 Das Programmierparadigma

Das Programmierparadigma begleitet das Konzept durchweg. Nur die rigorose Einhaltung der aufgestellten Richtlinien ermöglicht die erfolgreiche Konzeptumsetzung und die Nutzung der Vorteile des Konzepts. Die Konstellationen von Methoden und Standpunkten definieren dieses Paradigma.

- Programmierung erfolgt ausschließlich in den Framework-Sprachen
- System-Architektur baut nur auf Frameworkbibliotheken auf (Nutzung der Vererbung)
- Strikte Fehlerbehandlung in allen Bereichen

- Datasets sind primäre Daten-Container
- Keine gebundenen Steuerelemente
- Baukastensystem der Komponenten
- Komponenten werden in Assemblies verpackt
- Keine Komponenten von Dritten (wie OCX)
- Konfiguration nur über XML oder Datenbank
- Kein SQL-Code im Programmcode
- Keine realen Prozedurnamen im Quellcode
- Keine Benutzung der Registrierung
- Nur XCopy-Deployment
- Funktionale Trennung von Prozeduren/Funktionen im Programmcode
- Reports/Datenausgabe nur über XML und Browser
- Betriebssystemkonforme Programmierung
- Einheitliche Formulare über alle Komponenten hinweg
- Jeder Datensatz muss über die Attribute Objekt-ID und Klassen-ID verfügen und darüber zu identifizieren sein
- Keine übertriebene, sondern zweckmäßige Normalisierung der Datenbanktabellen
- Datenmanipulation nur über Prozeduren oder Trigger
- Zentrale Instanziierung der Assemblies (cFactory)
- Grundsätzliches Arbeiten mit SourceSafe (Quellcodeverwaltung)
- Konsequente Nutzung der Sicherheitsfeatures des Frameworks (Verschlüsselung)
- Typenverträglichkeit im Programmcode sicherstellen (String nicht ungeprüft in ein Integer konvertieren)
- Separate wieder verwendbare Funktionen für Datenbankkommunikation nutzen (siehe Anhang „Gestaltung der Prozedur zur Datenübergabe an die Kommunikationsebene“)
- Anwendung der Polymorphie

5.5 Das Datenbankdesign als zentraler Punkt der Geschäftslogik

Das Kernstück dieses Konzeptes bildet das Datenbankdesign. Die Strukturen der Tabellen, die Inhalte der Funktionen, Prozeduren und Trigger bilden die Geschäftslogik der Anwendung ab. Datenbanken sind für diese Dienste prädestiniert und bewältigen die gestellten Aufgaben zuverlässig und hochperformant. Ein Großteil der hier eingesetzten Software folgt keiner strikten und zentralen Logik, d.h. Funktionen, Formeln und Bedingungen werden sowohl im Programmcode als auch in der Datenbank hinterlegt. Werden logische Fehler während des Programmeinsatzes festgestellt, können diese nur mit hohem Aufwand behoben werden. Fehler sind zwar nicht vollständig vermeidbar, aber die Fehlersuche und deren Beseitigung kann einfacher und zentraler gestaltet werden. Die

Vorgehensweise im Rahmen dieses Konzeptes trägt dazu bei, logische Fehler an einer Stelle zentral zu behandeln bzw. zu eliminieren.

Ein äußerst wichtiger Aspekt ist der Aufbau der Datenbanktabellen. Üblicherweise besitzt jede Tabelle einen primären Schlüssel, mit dem ein Datensatz in dieser Tabelle eindeutig identifiziert wird. An diesem Merkmal ist zu erkennen, dass beim Abruf von Daten die Tabelle, in der sich der Datensatz befindet, vom Namen her bekannt sein muss. Um dem Anspruch des wieder verwendbaren Codes wirklich gerecht zu werden, genügt demzufolge dieser Primärschlüssel nicht. Um Funktionen, die Ergebnisse aus Datenbankabfragen zurückgeben, mehrfach wieder verwenden zu können, ist ein erweitertes Schlüsselsystem nötig. Hinzu kommt, dass bereits in der untersten Ebene der objekt- und komponentenorientierten Softwarearchitektur, der relationalen Datenablage, die Bildung der Objekte berücksichtigt werden muss. Die Definition eines Objektes beinhaltet, dass die Identifizierung über einen eindeutigen Schlüssel erfolgt, also muss der Primärschlüssel um ein Merkmal zur Identifizierung erweitert werden. Ein Objekt aus der Sicht der Komponente entspricht einem Datensatz in der Datenbank. Das fordert einen Tabellenaufbau, der diesem Anspruch genügt.

5.5.1 Objektstrukturierte Datenbanktabellen

Der Ausgangspunkt für eine komponentenorientierte Datenbankanwendung ist die Datenbank selbst mit ihren Tabellen und Prozeduren. Durch einen speziellen Aufbau der Tabellen und Prozeduren werden die Daten für die Objektbildung vorbereitet. Der Aufbau einer Tabelle erfolgt so, dass nicht nur ein Primärschlüssel, die Objekt-ID, sondern auch ein Tabellenschlüssel, die Klassen-ID, existieren. So wird realisiert, dass mit Hilfe dieser Identifikationen in einer Datenbank ein beliebiges Objekt (Datensatz) identifiziert werden kann. Eine separate Tabelle regelt die Zuordnung der Klassen-ID. Diese separate Tabelle ist nicht nur für die Vergabe der Klassen-ID verantwortlich, es werden auch zusätzliche Attribute vergeben, die einen beschreibenden Charakter besitzen, die Tabelle trägt den Namen SUPERKLASSEN. Die zusätzlichen Attribute definieren den Rechtekey, den Listenindex, den Listennamen, den Listentyp, das geerbte Recht, die Modulgruppe sowie diverse Bezeichnungen. Die nachstehende Abbildung zeigt den Aufbau dieser Tabelle schematisch.

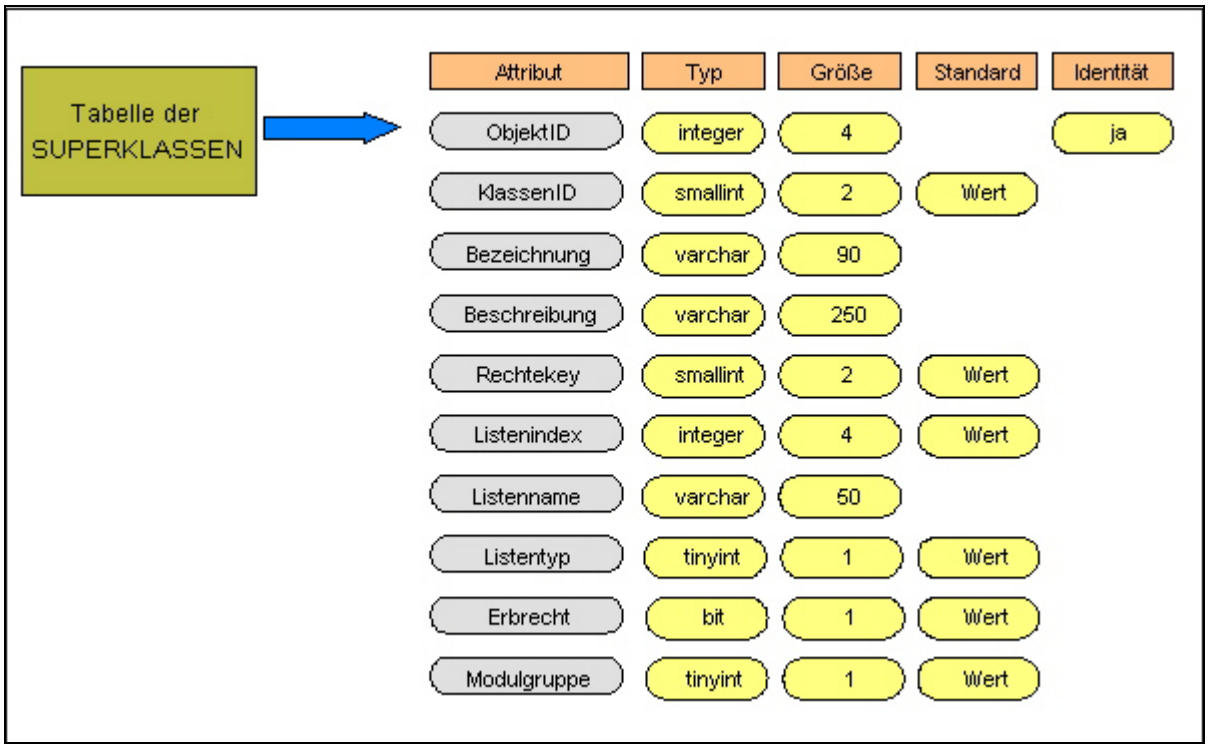

Abbildung 5-9 Struktur Superklassen

Die Attribute bilden in Verbindung mit benutzerdefinierten Systemtabellen und Abfragen diverse Kriterien und Einstellungen für die Anwendung des aktuellen Benutzers beim Laden des Programms. Das sind klassenabhängige und benutzerabhängige Rechte, Formate für Listen sowie Anordnung, Hierarchie und Beschreibungen der Module. Die SUPERKLASSEN-Tabelle listet alle Tabellen auf, die in der Datenbank erstellt werden und für die Anwendung notwendig sind. Diese Enumeration ist parallel in der Anwendung implementiert und vereinfacht die Unterscheidung und den Zugriff der klassenabhängigen Objekte. Einen exemplarischen Auszug der SUPERKLASSEN-Tabelle, siehe Abbildung 5-10 , veranschaulicht die beschriebene Funktionalität.

	ObjektID	KlassenID	Bezeichnung	Klassenbeschreibur	Rechtekey	Listenindex	Listenname	Type	Erbrecht	ModulGroup
▶	64	4	ARCHIV_TB_AUFTRAG	Auftrag Archiv Moc	24	1087	Aufträge Archiv	3	1	0
	51	4	ARCHIV_TB_FERTIGUNGSAUFTRA(	Fertigungsauftrag ,	25	1086	Fertigungsaufträge	3	1	0
	67	4	GLOBALQUERY	Für Globalsuche	34	2014	Globalsuche	1	1	100
	66	4	REL_OBJEKT_ANALOGIE	<NULL>	<NULL>	0	<NULL>	0	0	0
	122	4	REL_OBJEKT_NOTES	<NULL>	<NULL>	0	<NULL>	0	0	0
	155	4	REL_OBJEKT_OBJEKT	<NULL>	<NULL>	0	<NULL>	0	0	0
	116	4	SUB_TB_ABSCHREIBUNGEN	<NULL>	<NULL>	0	<NULL>	0	0	0
	34	4	SUB_TB_ANREDE	Klasse: Anreden	31	1064	Anreden	2	0	0
	144	4	SUB_TB_ARBEITSGANG_DETAIL	<NULL>	<NULL>	0	<NULL>	0	0	0
	59	4	SUB_TB_ZUKAUFKLASSEN	<NULL>	<NULL>	0	<NULL>	0	0	0
	93	4	SUB_TB_ZWISCHENLAGER	Klasse: Hier steht e	19	2006	Lager/Produktion	3	1	0
	4	4	SUPERKLASSEN	Klassenzusammenf.	34	1067	Superklassen	2	0	4
	77	4	SYS_APPLICATION	<NULL>	<NULL>	0	<NULL>	0	0	0
	127	4	SYS_TB_LISTEN	<NULL>	<NULL>	0	<NULL>	0	0	0
	89	4	SYS_TB_LOCKED_OBJECTS	<NULL>	<NULL>	0	<NULL>	0	0	0
	147	4	SYS_TB_LOGIN_ERROR	Klasse: Hier steht e	37	2012	Login Errors	3	1	0
	148	4	SYS_TB_LOGIN_LIST	Klasse: Hier steht e	37	2011	Login Protokolle	3	1	0
	156	4	SYS_TB_MODULGROUPS	<NULL>	<NULL>	0	<NULL>	0	0	0
	50	4	SYS_TB_PRINT_AUSGABE	Klasse: Ausgaben (	0	2024	Printprotokolle	3	1	0
	128	4	SYS_TB_PROCEDUR	<NULL>	<NULL>	0	<NULL>	0	0	0
	124	4	TB_ANGEBOTE	Angebot Modul: Hie	14	1048	Angebote	1	0	3
	142	4	TB_ANGEBOTPOSITIONEN	Klasse: Angebotpo:	14	2028	Angebotpositionen	3	1	0
	135	4	TB_APL	Arbeitsplan Modul:	11	1015	Arbeitsplan	1	0	4
	63	4	TB_AUFTRAG	Aufträge Modul: Hi	24	1060	Aufträge	1	0	3
	32	4	TB_BAUGRUPPEN	Baugruppen Modul:	6	1007	Baugruppe	1	0	2
	3	4	TB_BENUTZER	Benutzer Modul: Hi	1	1001	Benutzer	1	0	3
	96	4	TB_BESTELLUNG	Einkauf Modul: Hier	22	1044	Einkauf	1	0	3

Abbildung 5-10 Superklassen-Tabelle

Wie aus der Abbildung 5-9 ersichtlich, gibt es mehrere unterschiedliche Tabellentypen. Bereits die Namen der Tabellen lassen ihr jeweiliges Charakteristikum bereits erkennen:

- Archiv-Tabellen: beginnend mit ARCHIV_
- Relationstabellen: beginnend mit REL_
- Untergeordnete Tabellen: beginnend mit SUB_
- Systemtabellen: beginnend mit SYS_
- Haupttabellen: beginnend mit TB_
- Temporäre Tabellen: beginnend mit TEMP_
- Importtabellen: beginnend mit X_

Diese Nomenklatur vereinfacht den Umgang mit den Tabellen und klassifiziert sie visuell und funktionell. Alle Tabellen, inklusive der SUPERKLASSEN-Tabelle, besitzen den gleichen strukturellen Aufbau. Die Objekt-ID und die Klassen-ID sind immer und in jeder Tabelle Attribute, d.h. eine Tabelle muss mindestens diese zwei Felder besitzen, um als Datenbanktabelle dem Konzept bzw. der Softwarearchitektur zu entsprechen und damit den Umgang mit ihr zu ermöglichen. Jede Anfrage an die Datenbank ist immer mit der Objekt-ID und der Klassen-ID verbunden. Der Aufbau der anderen Tabellen der Datenbank kann in der Abbildung 5-11 nachvollzogen werden.

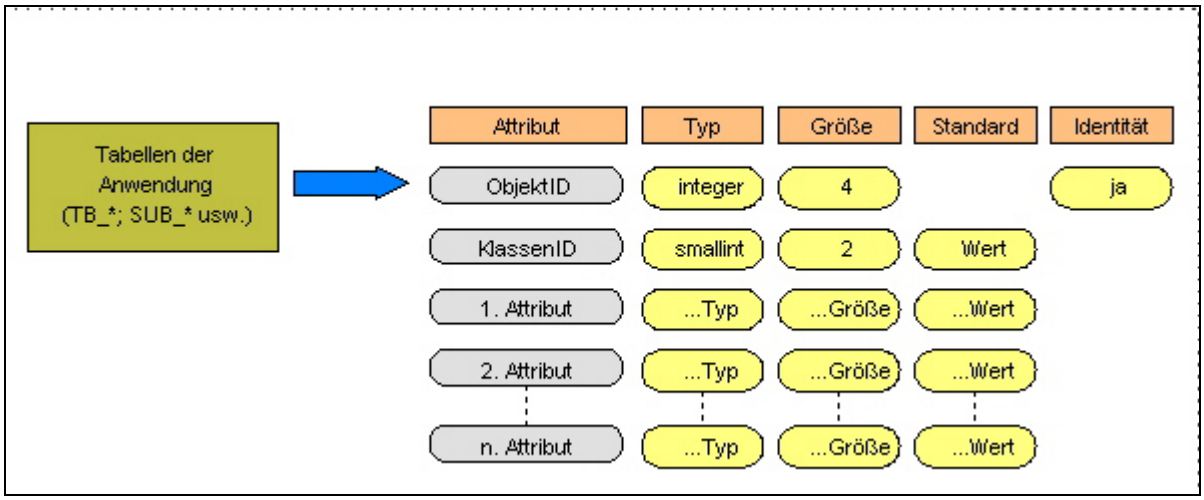

Abbildung 5-11 Tabellenstruktur

Die Tabellentypen entstehen durch die funktionelle Klassifizierung:

- Die *Haupttabellen* beinhalten die Primärdaten der entsprechenden Komponente.
- *Untergeordnete Tabellen* und *Relationstabellen* haben hauptsächlich eine Normalisierungsfunktion (1:n bzw. n:n Beziehungen), zusätzlich dienen sie in bestimmten Fällen der hierarchischen Datenablage.
- Die *Archivierungstabellen* enthalten Daten, die dauerhaft für die Historie abgelegt werden. Gleichzeitig werden die aktuellen Tabellen nicht zusätzlich belastet und eine gleich bleibend hohe Performance garantiert.
- *Systemtabellen* speichern Daten, welche die Mainanwendung benötigt. Diese Tabellen sind von der Komponente unabhängig.
- *Temporäre Tabellen* sind zeitabhängig, werden deshalb meistens nach deren Benutzung entfernt.
- Die *Importtabellen* stellen importierte Daten aus Fremddatenbanken größtenteils statisch dar, z.B. Tabellen, die aus alten Anwendungen stammen, auf die entweder nicht verzichtet werden kann und/oder diese besitzen einen rein informativen Charakter.

Notwendige Verknüpfungen der Haupttabellen zu anderen Tabellen werden mit Hilfe der Relationstabellen erreicht. So wird beispielsweise die Auflistung der Parent-Child-Beziehungen einzelner Aufträge (Kinder/Eltern-Kommissionen) oder Artikel (hierarchische Baugruppen/Stücklisten) umgesetzt. Eine Komponente selbst benötigt mitunter mehrere Tabellen, um die erforderlichen Daten (Menge und dazugehörige Einheit) vollständig darstellen zu können. In Kapitel 6 werden einzelne Situationen exemplarisch aufgezeigt.

5.5.2 Prozeduren, Funktionen und Trigger

Datenbankobjekte, wie Prozeduren, Funktionen und Trigger, manipulieren Daten oder stellen sie als Ergebnistabelle dar. Das Softwaredesign sieht vor, dass eine Datenmanipulation ausschließlich nur von der Datenbank selbst durchgeführt wird. Das ist ein wichtiger Punkt im Konzept, der bedingungslos umgesetzt werden muss. Der Prozeduraufbau ist in der Abbildung 5-12 links beschrieben.

Die Klasse der Komponente, in der das Objekt initialisiert wird, hat unter anderem die Aufgabe, die Datenmanipulation für die gespeicherten Datenbankprozeduren vorzubereiten. So wird eine syntaktische (korrekte Regelanwendung) und eine semantische (korrekte Ausdrucksform) Validierung der Daten auf der Anwendungsebene und eine logische (folgerichtige) Validierung auf der Datenbankebene durchgeführt. Die Prüfung auf logische Fehler findet demzufolge in den Prozeduren statt. Trigger hingegen sind ereignisbasierende Prozeduren, vergleiche hierzu die Abbildung 5-12 rechts. Bei jedem Datenänderungsereignis (INSERT, UPDATE oder DELETE) werden sie automatisch ausgeführt. Inhaltlich können sie wie gespeicherte Prozeduren konstruiert werden, wobei die Parameterliste durch die Tabellenattribute ersetzt wird.

Prozeduren eignen sich nicht nur zur Datenmanipulation, sondern auch hervorragend zur Datenvorbereitung oder Konfiguration der Anwendung. Beim Start der Software werden zunächst einige Prozeduren ausgeführt, die Ergebnisse zurückliefern. Sie enthalten z.B. die Rechte für den Benutzer der Anwendung, Formatierungen für Listen oder Klassen, in denen die Anwendung strukturiert und die Inhalte von Auflistungen erstellt werden. Der Funktionsumfang, den diese Prozeduren abdecken, wird bei einer Demonstration deutlich. Dass Datenmanipulationen oder Auflistungen mit gespeicherten Prozeduren realisiert werden, ist bekannt und wird häufig realisiert. Wie sich eine dynamische Konfiguration gestaltet, soll der folgende Abschnitt demonstrieren.

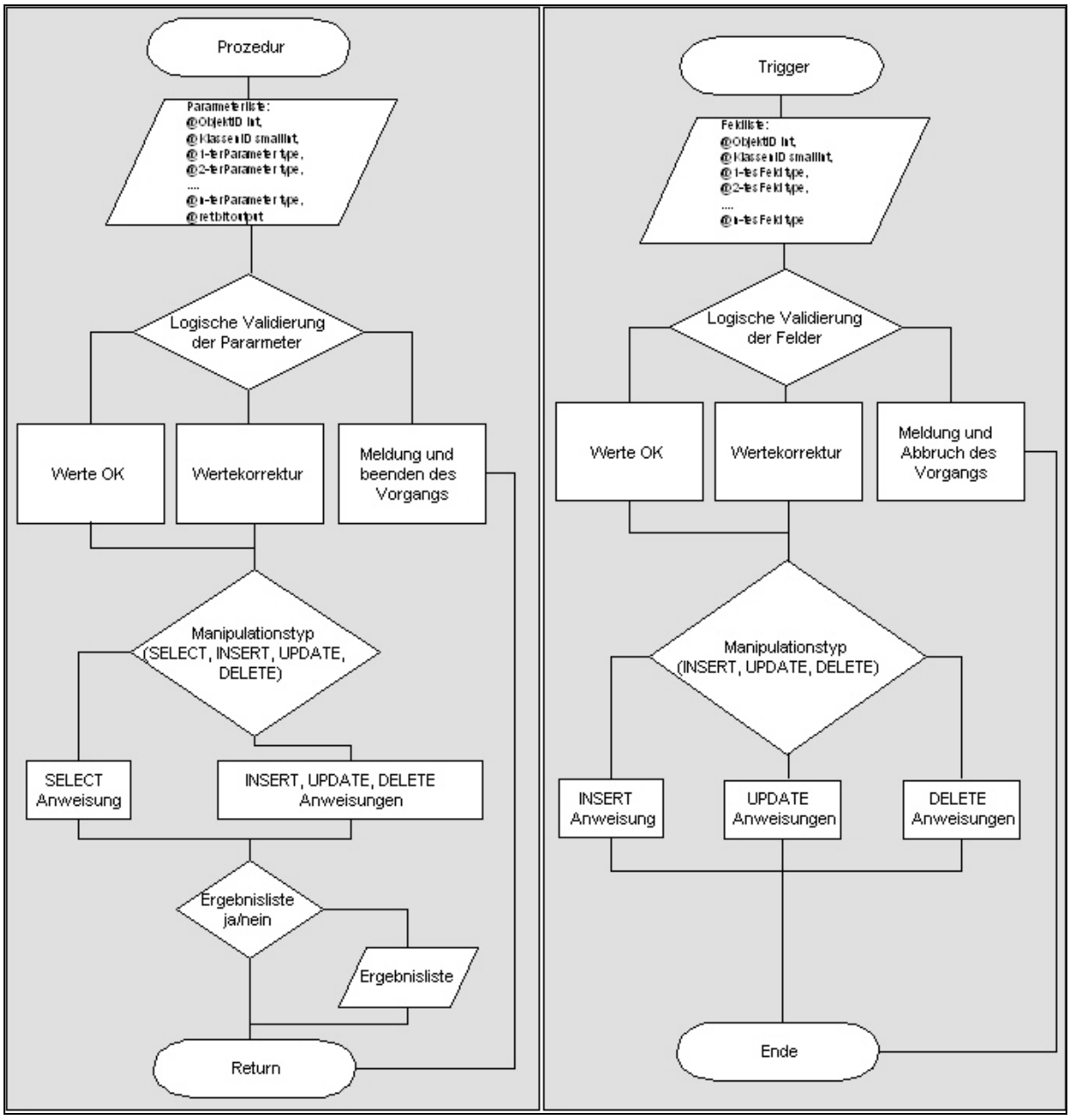

Abbildung 5-12 Trigger- und Prozeduraufbau

5.5.3 Schema zur Nutzung der Datenbankobjekte

Die verschiedenen Module/Komponenten der Systemlösung werden in einer Baumstruktur gemäß Abbildung 5-13 dargestellt. So wird für den Anwender eine visuelle Unterstützung geschaffen, um schneller und übersichtlicher in der Anwendung navigieren zu können.

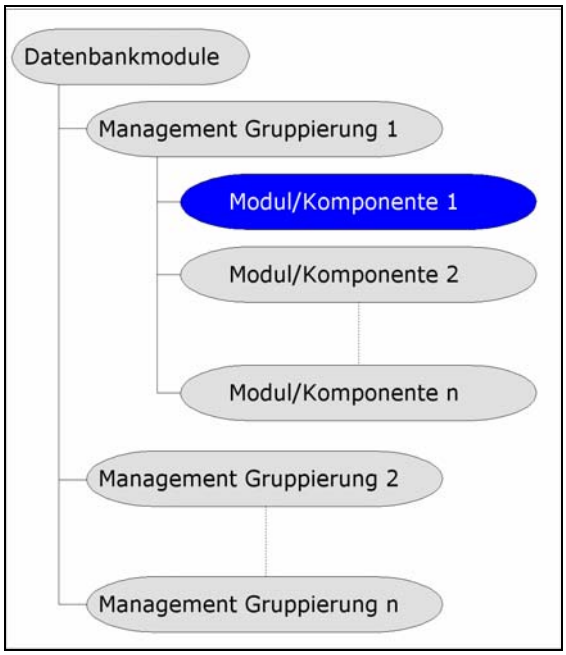

Abbildung 5-13 Modulstrukturierung

Diese Grafik (Abbildung 5-13) verdeutlicht die Strukturierung der Module. Jede Komponente kann einer bestimmten funktionalen Klasse zugeordnet werden. So sind beispielsweise Rechnungen und Lieferscheine, also alle Vertriebsdokumente, dem Officemanagement zugeordnet; Zeichnungen, Programme oder gescannte Dokumente gehören dem Dokumentenmanagement an. Solch eine Unterteilung ist auch hinsichtlich der betrieblichen Bereiche möglich, demnach nicht fest definiert und zur Laufzeit veränderbar.

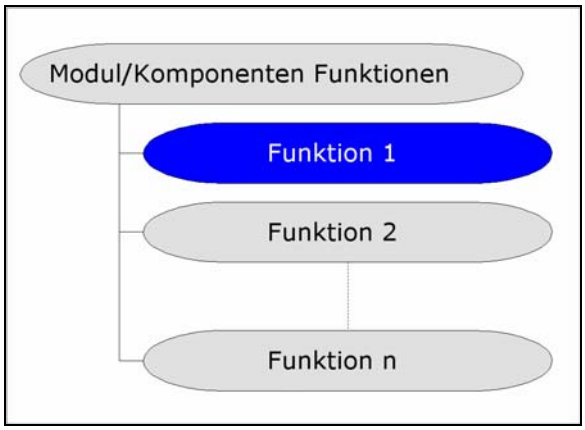

Abbildung 5-14 Modulfunktionen

Die in Abbildung 5-13 hervorgehobene Komponente 1 soll die momentan selektierte Komponente in der Anwendung darstellen. Die Komponenten besitzen in der Regel unterschiedliche Funktionen und Eigenschaften. Allerdings wiederholen sich bestimmte Charaktere. Deshalb muss eine Auswahl der unterschiedlichen Eigenschaften pro Komponen-

te getroffen und auch dargestellt werden, siehe Abbildung 5-14. Je nach gewählter Komponente werden die dazugehörigen Funktionen aufgelistet. Die Zuordnung der Funktionen pro Komponente geschieht während der Laufzeit der Anwendung. Die Modulstrukturierung und Funktionsauflistung in der Anwendung geschieht im Explorerstil und ist daher dem Anwender vertraut. Die dazu notwendigen Daten werden durch Prozeduren aus der Datenbank geholt. Der hierarchische Aufbau der Modulstrukturierung wird mit rekursiven Funktionen und durch erweiterte Steuerelemente in der Anwendung konstruiert.

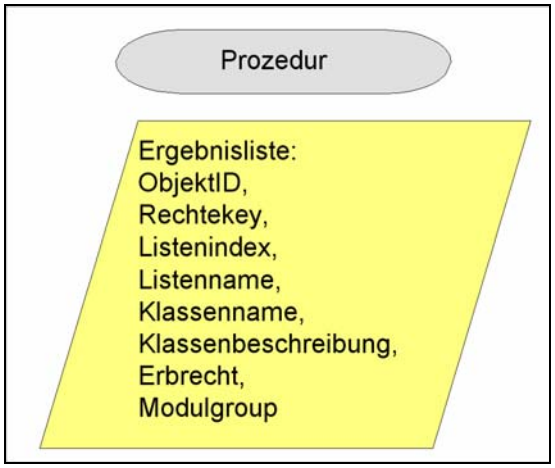

Abbildung 5-15 Prozedur Schema

Die Ergebnisse aus der Abfrage, wie in die Abbildung 5-15 zeigt, werden nun der Anwendung übergeben, diese generiert mit Hilfe der erwähnten Funktionen die Struktur der Module. In der nachfolgenden Grafik gemäß Abbildung 5-16 ist die rekursive Funktion schematisch dargestellt. Sie gestaltet auch den Modul- bzw. Komponentenbaum in der Anwendung.

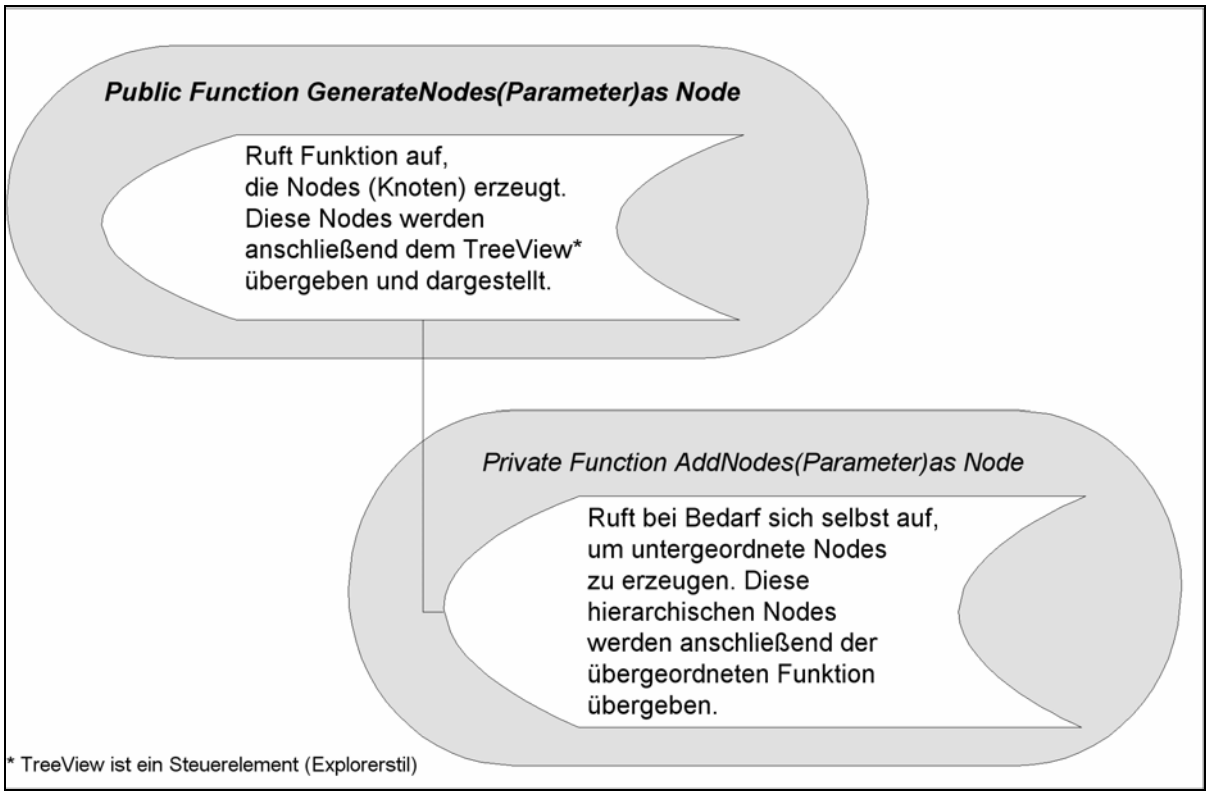

Abbildung 5-16 Rekursive Erstellung einer Hierarchie

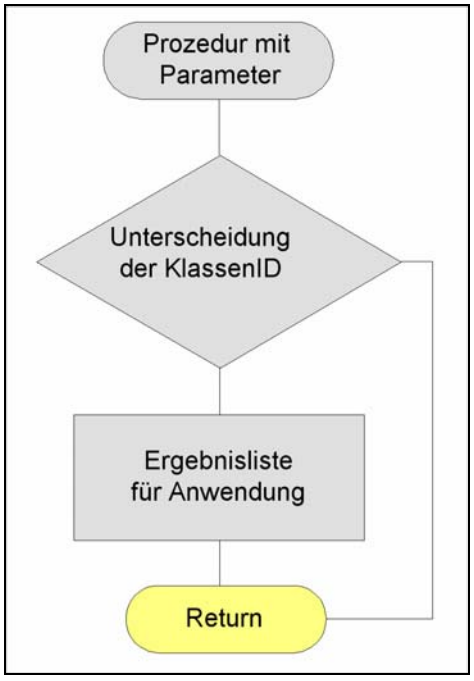

Abbildung 5-17 Schematische Prozedur

Der Aufruf dieser Prozedur mit den entsprechenden Parametern gibt die Auflistung der Funktionen zurück. Beispielsweise sind für die Auftragskomponente die Funktionen: Eigenschaften anzeigen, Relationen zu Kunden, Artikel oder Arbeitspläne auflisten, Aufruf

von Reports/Listen oder die Generierung der Fertigungsaufträge relevant. Ist die Funktionsübersicht aufgebaut, steht diese nach der Funktionsselektion zur Verfügung. Die Berechtigung zur Funktionsausführung ist natürlich abhängig vom jeweiligen Recht des angemeldeten Benutzers, welches bei der Auswahl des Moduls bzw. der Komponente gesetzt wurde. Die Ergebnisliste aus der Prozedur (Abbildung 5-17), also die Modul/Komponentenfunktionen (vgl. Abbildung 5-14), beinhaltet den Namen der Funktion und einen Key, der zur Auswertung und Differenzierung im Programmcode herangezogen wird. Der Vorteil dieser Gestaltung des Zusammenspiels der Datenbank mit der Anwendung liegt klar auf der Hand. Wenn eine Komponente im Zusammenhang mit anderen Modulen implementiert ist und diese jetzt in einer der anderen Komponenten benötigt wird, kann sie ohne Aufwand implementiert werden. Lediglich ein zusätzlicher Eintrag in die Modulfunktionsliste stellt die bereits existierende Funktionalität auch in dieser Komponente zur Verfügung. Besitzt z.B. die Artikelkomponente die Funktionen, Eigenschaften, Relationen zu Dokumenten und Notizen, aber der Auftragskomponente fehlt die Notizen-Funktion, dann wird in die Ergebnisliste der Aufträge die Funktion "Notizen" mit dem dazugehörigen Key eingetragen. Im Augenblick des Speicherns der veränderten Ergebnisliste steht dem Anwender die neue Funktion zur Verfügung. Auch hinsichtlich der "Online-Entwicklung" ist dies eine optimale Lösung. Die Funktionen können zuerst erstellt und getestet werden. Steht einer Veröffentlichung nichts mehr im Weg, wird der Neueintrag in die Ergebnisliste aufgenommen und die neue Funktion steht damit den berechtigten Anwendern zur Verfügung.

Um das Problem der Bekanntgabe von Prozedurnamen im Quellcode des Programms zu entschärfen, existiert eine separate Tabelle, in der dem realen Prozedurnamen ein Aliasname zugeordnet wird. Beim Laden der Anwendung wird dem Programm diese Auflistung übergeben. Ab diesem Zeitpunkt stehen alle Prozeduren über Aliasnamen zur Verfügung. Sind sensible Änderungen in Prozeduren notwendig, können im Debug-Modus der Programmierumgebung die modifizierten Prozeduren manuell angegeben werden. Der Anwender arbeitet solange ungestört mit der herkömmlichen Prozedurversion. Zur Veröffentlichung der Prozedur muss der reale Prozedurname in der Tabelle der Aliasnamen abgeglichen werden. Ab dem Neustart benutzt dann die Anwendung die aktualisierten Prozeduren. Genau so läuft auch das Verfahren bei der Gestaltung neuer Prozeduren ab.

5.6 KONZEPTBEZOGENES PROJEKTMANAGEMENT

Das hier zum Einsatz kommende Projektmanagement beinhaltet die Organisation, die Planung, die Kontrolle und die Steuerung des gesamten Arbeitsvorhabens. Für die Durchführung des Konzeptes wurde das traditionelle Projektmanagement nach [21] gemäß Abbildung 5-18 abgeändert, da einige Punkte des bekannten Projektmanagement hier

nicht relevant sind oder zusammengefasst werden konnten. In der folgenden Abbildung ist das hier benutzte konzeptbezogene Projektmanagement schematisch dargestellt.

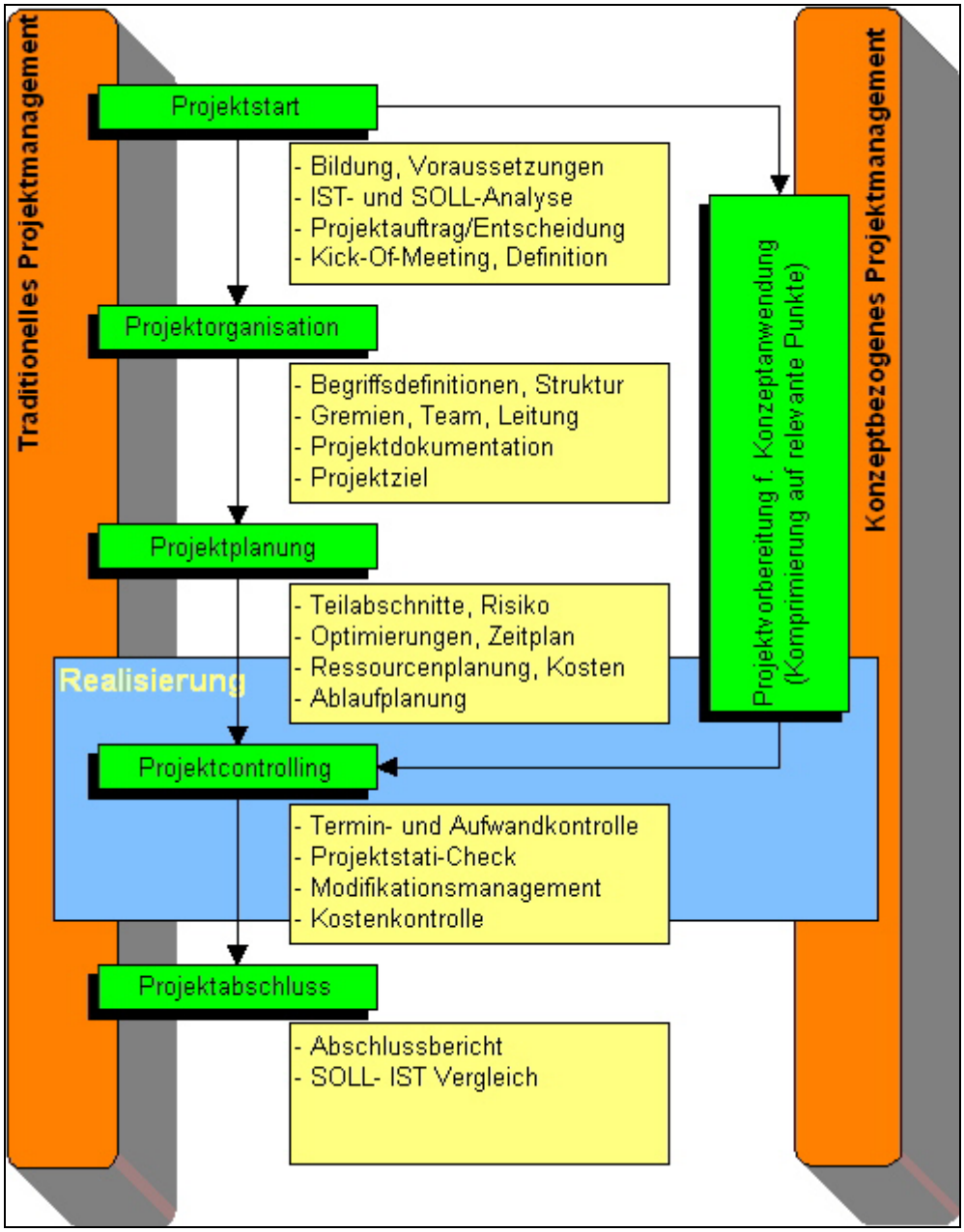

Abbildung 5-18 Projektmanagement schematisch

Im linken Teil stellt sich das traditionelle Projektmanagement mit den einzelnen Phasen der Projektabwicklung dar. Im rechten Teil der Abbildung 5-18 kann nachvollzogen werden, welche Punkte des traditionellen Projektmanagements im gestrafften, d.h. im konzeptbezogenen Projektmanagement, Beachtung finden. Ferner wird die Zusammenfassung der einzelnen Phasen aus dem traditionellen Projektmanagement im konzeptbezogene Projektmanagement aufgezeigt.

Um die Übersichtlichkeit und Transparenz des Projektes zu gewährleisten, empfiehlt es sich, die Projekte auf die Komponenten der Systemlösung zu beziehen. So entstehen mehrere kleine Projekte, die einen handlichen Umgang versprechen. Ebenso erleichtert diese Methode das Projekt-Controlling. Durch die Einbettung der Komponentenentwicklung in den Rahmen des Projektmanagements werden eindeutige Aufgabenstellungen und klare Verantwortungen definiert. Wichtig für den korrekten Projektablauf ist ein gut organisiertes Controlling, welches auch als Frühwarnsystem bezeichnet werden kann. So sind unrealistische Vorgaben erkennbar und unvorhersehbare Änderungen, Prioritätsverschiebungen und Vorgehensalternativen gut in den Griff zu bekommen. Wie die Einbettung der einzelnen Komponenten in die Projektmanagementmethode geschieht, wird in Kapitel 6 ausführlich beschrieben.

5.7 Untersuchung und Klassifizierung der Unternehmensorganisation

Die allgemeine Definition des Begriffes Organisation in einem Lexikon lässt die nicht triviale Aufgabe dieses Konzeptpunktes erkennen, sie lautet: "Organisation ist allgemein die Art und Weise, wie die Teile eines Ganzen untereinander und zu diesem Ganzen orientiert sind und zusammenwirken." Diese Aussage spiegelt aus unternehmerischer Sicht die Gliederung, die internen Abläufe, Instrumente, Verknüpfungen und Hierarchien einer Unternehmung wider.

Durch den verschärften Wettbewerb und die darauf basierende Notwendigkeit der kundenorientierten Ausrichtung der Unternehmensorganisation besteht immer der Bedarf einer Geschäftsprozessoptimierung. Ziel ist es, durchgehende Geschäftsprozesse von der Kundenanfrage bis zur Auslieferung zu gestalten. Die oberste Regel besagt: „Jedes Unternehmen sollte optimal entlang der Wertschöpfungskette organisiert sein und Schnittstellen minimiert werden. Sie bedeuten Stillstand, Fehlerquellen und organisatorische Zuständigkeitsprobleme“. [22]

Wie kann eine Untersuchung und Einschätzung unter Berücksichtigung der genannten Grundregeln vorgenommen werden?

Für jeden Kerngeschäftsprozess werden Personen ausgewählt, bestehend aus Mitarbeitern und Führungskräften, die den jeweiligen Geschäftsprozess sehr gut kennen. Die Arbeitsgruppen haben die folgenden auf softwaretechnische Berührungspunkte ausgerichtete Aufgaben: Untersuchungen, die nicht in Verbindung mit der eigentlichen Softwareentwicklung stehen, haben keine Relevanz für dieses Konzept. Die Planung, Steuerung und Überwachung wird vom Projektmanagement übernommen, die Durchführung der Untersuchung selbst erfolgt durch die fachlichen Mitarbeiter. Zum Aufgabengebiet des Projektmanagements zählen die Aufgabenstellung, der Ablaufplan, die Überwachung der Quali-

tät und der Termine, das Anleiten und Koordinieren der beteiligten Mitarbeiter und die anschließende Auswertung.

Aus diesem Sachverhalt wird die folgende Vorgehensweise definiert:

- Ist-Prozesse beschreiben
- Schwachstellen analysieren
- Soll-Prozess konzipieren
- Implementierungsplan des Soll-Prozesses entwerfen

Der Implementierungsplan wird den Verantwortlichen präsentiert und mit der Softwarearchitektur abgeglichen. Bei Komplikationen in der Umsetzung, d.h. Unverträglichkeiten mit der Softwarearchitektur, muss Rücksprache mit den Untersuchenden getroffen und die Alternativen besprochen werden. Da alle Geschäftsprozesse im Ansatz einem Standard entsprechen, sind Unverträglichkeiten selten in der Grundstruktur zu finden, eher bei ausgebauten oder modifizierten Prozessen (firmeneigene Prozesse). Um genau diese Anpassungen der ausgebauten oder modifizierten Prozesse zwischen Unternehmensorganisation und Softwarearchitektur geht es bei der Untersuchung und Klassifizierung. Sind Komplikationen schon in der Grundstruktur zu finden, muss eine Überprüfung der Softwarearchitektur in diesem Segment durchgeführt werden.

Betrachtung und Einschätzung der Strukturen und Prozesse im Unternehmen sind Voraussetzung für einen reibungslosen Projektanlauf. Nur eine funktionierende Infrastruktur bietet eine stabile Basis für die Systemlösung. Die Infrastruktur eines Unternehmens wird durch das Firmennetzwerk abgebildet. Performanceprobleme bei Transaktionen dürfen beispielsweise nicht auftreten. Ein hochwertiges LAN (Local Area Network) engt die Fehlersuche ein und gewährleistet stabile Verbindungen. Die Prozesse, die in einem Unternehmen ablaufen, müssen zuvor aufgenommen und bewertet werden. So können bereits im Vorfeld Schwachstellen erkannt und bei der Auswertung Lösungen vorgeschlagen werden. Im folgenden Abschnitt wird repräsentativ ein ausgewählter Geschäftsprozess näher betrachtet.

Die in einem Unternehmen stattfindenden Geschäftsprozesse sind Abläufe, die den Arbeitsfluss definieren. So sind z.B. typische Prozesse die Erstellung eines Auftrages nach Auftragseingang, die Neuanlage eines Artikels, die Erstellung einer Rechnung oder die Generierung eines Fertigungsauftrages. Ziel der Prozessanalyse ist eine optimale Gestaltung des Ablaufes in Sachen Kosten und Ablaufzeit. Dadurch ist ein schnelleres Erkennen von Potentialen für Optimierungen und eine Fehlerdiagnose möglich. Die Standardisierung von Prozessen erhöht die Transparenz und vereinfacht die innerbetriebliche Informationsbeschaffung, die auch zur Preisgestaltung herangezogen werden kann.

In jedem Unternehmen existieren derartige Vorgänge in den verschiedensten Varianten. Diese Prozesse sind geprägt von der meist vorhandenen Software oder der angewandten innerbetrieblichen Routine. Selbst wenn nur mit Hilfe von Officeprogrammen Dateien erstellt und verwaltet werden, wird dieser Vorgang als Prozess gewertet. Eine Prozessbewertung kann auch durchgeführt werden, wenn betriebliche Software im betreffenden Unternehmen bisher nicht eingesetzt wurde, d.h. beispielsweise bei einem komplett manuellen Auftragswesen. Die Zusammenfassung aller Kerngeschäftsprozesse zu einem Gesamtgeschäftprozess gemäß Abbildung 5-19 beschreibt die Wertschöpfungskette.

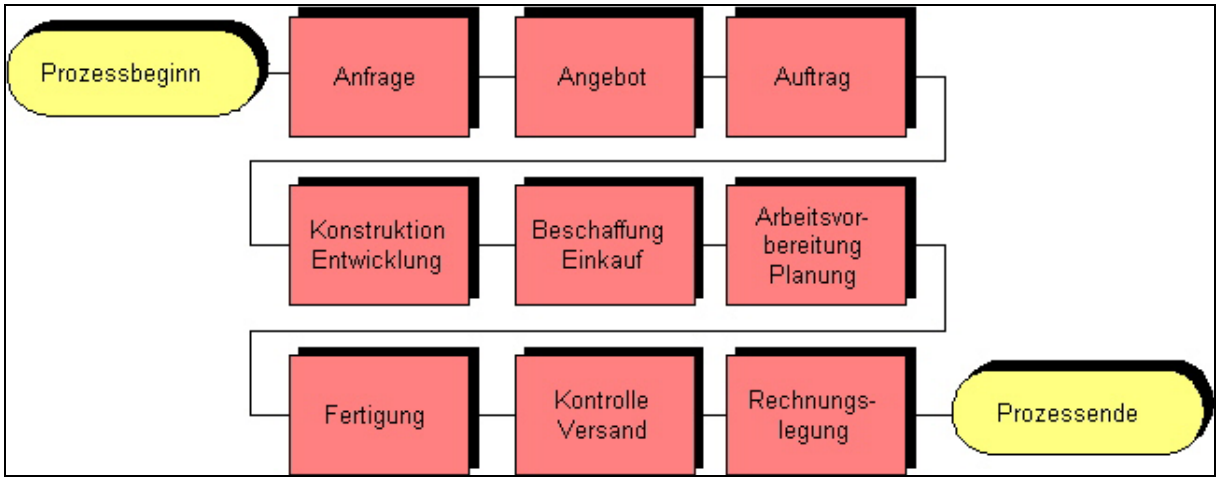

Abbildung 5-19 Gesamtgeschäftsprozess

Zu beachten ist, dass jeder der Kernprozesse wiederum Teilprozesse beinhaltet. So beinhaltet der Kernprozess der Anfrage unterschiedliche Teilprozesse gemäß Abbildung 5-20.

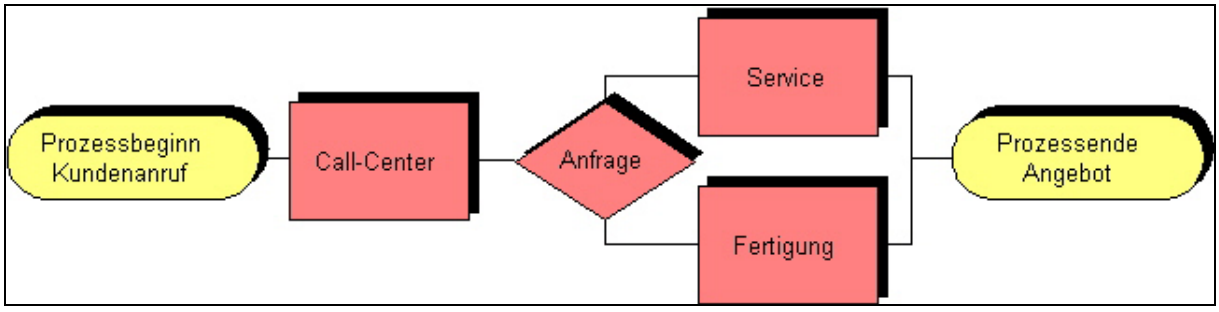

Abbildung 5-20 Teilprozess Anfrage

Andere Prozesse sind bedeutend komplexer, z.B. die Fertigung, sie beinhaltet mehrere Teilprozesse. Hier können die Feinplanung und die Betriebsdatenerfassung exemplarisch herangezogen werden. Dass sich diese Prozesshierarchie nicht nur auf eine Ebene beschränkt, ist verständlich. Die Reihenfolge der Prozessanalyse richtet sich nach der vorher festgelegten Folge der Komponentenentwicklung.

5.8 VORGEHENSWEISE BEI DER KOMPONENTEN-ENTWICKLUNG

Die grundsätzliche Vorgehensweise ist für alle Komponenten gleich. Sicherlich werden für einige Komponenten nicht alle hier aufgeführten Detailschritte notwendig sein, deshalb wird für die Vorstellung ein Geschäftsprozess ausgewählt, der als so genannte Obermenge alle wesentlichen Aspekte der Komponenten enthält. Als beispielhafter Geschäftsprozess wurde die umfangreiche **Auftragsbearbeitung** ausgewählt. Zunächst ist für diesen Geschäftsprozess unter Einbeziehung des Anwenders ein Lastenheft zu erstellen, aus dem ein Pflichtenheft für den Programmierer abgeleitet wird, welches dann als Grundlage für die Entwicklung der Komponente gilt.

5.8.1 Erstellen des Lastenheftes

Das vollständige und widerspruchsfreie Lastenheft gemäß der Richtlinie nach VDI/VDE 3694 beschreibt das Projekt aus der Sicht der Anforderungen des Anwenders mit dem wichtigen Tenor:

„Was soll die Software leisten und welche Probleme soll sie lösen?"

Im Rahmen des Lastenheftes erfolgt jedoch keinerlei Vorgriff auf die konkrete Realisierung der Anforderungen. Im Wesentlichen beinhaltet das Lastenheft die folgenden Punkte:

- Einführung in das Projekt (Kosten, Personal, Termine usw.)
- Beschreibung der Ausgangssituation (IST)
- Aufgabenstellung (SOLL)
- Schnittstellen
- Anforderung an die Systemtechnik
- Anforderung für die Inbetriebnahme und den Einsatz
- Anforderung an die Qualität
- Anforderung an die Projektabwicklung

Das Lastenheft bezieht sich, wie beim Projektmanagement, immer auf die einzelne Komponente aus der gesamten Entwicklung. So entsteht die gleiche Struktur, wie bei der zu entwickelnden Software, denn eine gute Transparenz für Projekte dieses Ausmaßes ist Voraussetzung für eine erfolgreiche Abwicklung. Für das Lastenheft für die Komponente Auftragsbearbeitung gilt folgende Zielsetzung:

- Automatisierung des gewählten Geschäftsprozesses
- Entwicklung der dazu notwendigen Komponente
- Spätere Erweiterungen müssen möglich sein

- Leicht verständliche Bedienung
- Analogie der Konventionen der bereits eingesetzten Komponenten

Durch die Automatisierung des Geschäftsprozesses werden ein verringerter Aufwand bei der Datenpflege des jeweiligen Prozesses und ein durchgängiger Informationsfluss im Zusammenhang mit den existierenden Komponenten erreicht.

Voraussetzungen/Richtlinien:

Beim Entwurf einer Komponente müssen verschiedene Richtlinien und Standards berücksichtigt werden, z.B. die Richtlinie zur Rechnungserstellung oder der Inhalt von Aufträgen. Durch die Einhaltung von Standards vereinfacht sich die Datenpflege, da sie in den Grundzügen bekannt sind. Weiterhin ist das Design der Formulare so zu gestalten, dass sie mit dem Betriebssystem konform sind. Spezielle Voraussetzungen, die komponentenspezifisch sind, müssen berücksichtigt werden, wie z.B. fertigungsbedingte Änderungen in den Arbeitsfolgen. So beinhaltet beispielsweise die Richtlinie für die Rechnungserstellung folgende Pflichtdaten:

- die Kundennummer
- eigene Steuernummer
- die Auftragsnummer
- die Kundenanschrift bzw. Rechnungsanschrift
- die Positionen
- den Gesamtpreis
- das Zahlungsziel

Beschreibung des Ist-Zustandes als Ergebnis der Analyse:

Diese Erkenntnisse werden aus dem Punkt „Untersuchung und Klassifizierung der Unternehmensorganisation" dieses Konzeptes gewonnen. Sie beinhalten unter anderem den bisherigen Aufbau des gewählten Geschäftsprozesses und dessen Merkmale. Der derzeitige Betriebsablauf des Geschäftsprozesses wird kurz erläutert. Eine Ereignisstabelle hilft dabei, den Ablauf plausibel darzustellen, z.B.:

- Kunde stellt Anfrage
- Kunde erhält Angebot
- Kunde erteilt telefonisch Auftrag
- Auftragsbestätigung wird zugesandt und Fertigungsauftrag erstellt
- Fertigung wird veranlasst
- …

Schnittstellen:

Als Schnittstelle wird die Art und Weise bezeichnet, wie die Daten gewonnen bzw. eingegeben werden. Dies kann beispielsweise mündlich, wie bei der Auftragsannahme, bzw. schriftlich, wie bei der Anfrage erfolgen. Eine andere Möglichkeit ist die elektronische bzw. die empirische Datenerfassung, wie z.B. bei der Erstellung eines Protokolls.

Forderungen:

Über die Komponente sollen bestimmte Aufgaben rechnergestützt realisiert werden. Hier folgt die Aufzählung der Anforderungen für die jeweilige Komponente, z.B. Pflege der Auftragsdaten oder Erstellung des Angebotes. Weitere Forderungen wären denkbar, sie sind komponentenabhängig und lassen sich wie folgt gruppiert übersichtlich darstellen, beispielsweise als organisatorische, technische oder spezielle Forderungen.

Organisatorische Forderungen:

Der Prozess ist so zu organisieren, dass der Ablauf der Kundenbedienung und der Stammdatenpflege kunden- und benutzerfreundlich durchgeführt werden kann. Das bedeutet, dass Wartezeiten durch schnellere Informationsflüsse gesenkt werden und Folgeinformationen schnell zu beschaffen sind.

Technische Forderungen:

Die rechentechnische Realisierung der Komponente erfolgt an einem PC und die entwickelte Software soll unter beispielsweise Windows XP laufen. Die Funktion der Komponente wird parallel zur Entwicklung direkt im betroffenen Bereich von vorher bestimmten Benutzern getestet.

Forderungen in Ausnahmesituationen:

Da es immer wieder vorkommt, dass Hardware ausfällt, muss sichergestellt werden, dass ein Datenverlust zu keiner Zeit eintreten kann. Der Komplettausfall der Rechentechnik ist innerhalb kürzester Zeit zu beheben und die Wiederherstellung der Funktionstüchtigkeit der Komponente/Anwendung vorzunehmen.

Allgemeine Forderungen:

Für die Entwicklung, Architektur und Konzeption ist ein Framework mit interoperablen, standardisierten Sprachen zu nutzen. Die Praxisnähe der Anwendung muss durch ein entsprechendes Konzept gewährleistet sein und mit den Verantwortlichen abgestimmt werden.

Qualität:

Das Software-Produkt soll modular und wartbar sein. Die Qualitätsanforderungen lauten:

- hohe Effizienz
- Änderbarkeit
- Hardwareunabhängigkeit
- hohe Zuverlässigkeit

5.8.2 Erstellen eines Pflichtenheftes

Ein Pflichtenheft beschreibt wie und womit die Anforderungen zu realisieren sind und regelt die verbindliche Vereinbarung bezüglich der Realisierung und Abwicklung des Projektes (Richtlinie nach VDI/VDE 3694). Es beschreibt die technischen Festlegungen, die aufgrund der im Lastenheft genannten Ziele und Vorgaben benötigt werden, angefangen von der Festlegung auf Betriebssystem(e) und Programmiersprache(n), über die Auslegung der Server und der Organisation der Software, bis hin zum detaillierten Aufbau von Bildschirmdialogen, Datenbanktabellen und den benötigten Schnittstellen. Hierbei sind nicht nur die offensichtlichen Anforderungen wie beispielsweise "Angebote erstellen" und "Rechnungen schreiben" zu berücksichtigen, es müssen vielmehr alle Arbeitsabläufe der durch die Softwareentwicklung betroffenen Stellen untersucht werden. Hierbei müssen auch die Meinungen und Vorschläge der beteiligten Mitarbeiter berücksichtigt werden. Die Gestaltung des Pflichtenheftes für eine Komponente beruht auf der folgenden Struktur:

Auftraggeber: Anschrift und der Kontakt des Kunden und der Verantwortlichen.

Auftragnehmer: Anschrift und der Kontakt des Dienstleisters und der Verantwortlichen.

Version: Stand und Datum des Pflichtenheftes.

Projektbeschreibung und -ziel: An dieser Stelle erfolgt eine genaue Beschreibung der zu entwickelnden Komponente, auch im Hinblick auf das Zusammenspiel mit anderen bereits laufenden oder geplanten Komponenten. Weiterhin muss in diesem Abschnitt das Ziel benannt werden, welches mit der Erstellung der Komponente erreicht werden soll.

Forderungen und Kriterien: Die hier aufgeführten Punkte müssen vom Dienstleister unbedingt erfüllt werden. Sie sind Grundlage zur erfolgreichen Abschlussbetrachtung und Auftragserfüllung.

Zusätzliche Features: Besonderheiten oder diverse Funktionen die vom Auftraggeber gewünscht, aber nicht als Pflichtteil definiert wurden, sind hier aufzuführen. Möglich ist,

dass erst während der Entwicklung ersichtlich wird, ob diese Forderungen und Kriterien erfüllt werden können.

Ausgeschlossene Kriterien: Hier wird, falls notwendig, abgegrenzt welche Kriterien nicht erfüllt werden sollen. Es erfolgt ein Ausschluss von Funktionen oder Merkmalen der Komponente.

Anwendungsbereiche: Die Definition des Einsatzortes der Komponente, d.h. externe oder interne Einsatzumgebung, wie z.B. Büroräume, Abteilungen, Außengelände oder Lagerhalle.

Hard- und Softwareumgebung der Komponente: Festsetzung der Voraussetzungen für die Lauffähigkeit der Komponente, wie Betriebssystem, Infrastruktur und Datenbank (Bsp. Windows XP, TCP/IP-Protokoll, MS-SQL-Datenbank). Weiterhin ist die zu verwendende Hardware zu definieren, d.h. Mindestanforderungen für Arbeitsplatz und Server (Bsp. Pentium III 500 MHz, 256 MB RAM, Display 1024x768 Auflösung für Arbeitsstation, Netzwerk mit mind. 100MBit).

Benutzeroberfläche: Regeln zur Gestaltung der Benutzeroberfläche und der Konformität der Anwendung.

Ergänzungen: Als Anhang zu bezeichnende Merkmale und Funktionen, wie Exportformate, Protokoll- oder Reportaufbau, Logos auf Dokumenten oder Einhaltung externer Richtlinien (z.B. Buchhaltungslisten).

Datenstruktur der Komponente: Schema, wie die Daten in den jeweiligen Datenbanktabellen abgelegt werden, d.h. Tabellennamen mit Feldern, deren Typ, Länge und Beschreibung. Struktureller Aufbau von Prozeduren und Triggern.

Komponentendesign und Datenzugriff: Eine grobe Beschreibung des Aufbaus der Komponente, die den Datenfluss schematisch wiedergibt.

Qualität: Festlegungen zur Unterstützung der Einhaltung der Qualitätsanforderungen, z.B. die Verwendung des .NET Framework und eine renommierte relationale Datenbank zur Sicherstellung der Qualitätsziele.

5.8.2.1 Schnittstellen zu vorhandener oder künftiger Software

In der Regel sind in einem Unternehmen bereits verschiedene Softwaresysteme im Einsatz, so haben z.B. Anwendungen wie Office, CAD-Programme, CAM-Programme oder Bildbearbeitungssoftware einen hohen Verbreitungsgrad. Gerade im Bereich der CAD- und CAM-Software ist eine Anbindung an die Systemlösung wichtig und oft Bestandteil

des Lastenheftes der entsprechenden Komponente. Diese Schnittstellen sind besonders sensibel, da sie den Datenbestand der bereits laufenden Lösung mit beeinflussen. Das Design der Schnittstelle des Systems muss gut durchdacht sein und mit den verschiedensten Fremdanwendungen kommunizieren können. Bei älteren Anwendungen, die integriert werden müssen, stehen die Entwickler vor komplizierten Aufgaben. Sogar Programme, die keine konventionellen Schnittstellen zur Verfügung stellen (COM, API), müssen eingebunden werden, wenn auch nur im begrenzten Umfang.

Durch ein rationelles Schnittstellendesign und spezielle Programmiertechniken ist es möglich, diverse Anwendungen erfolgreich einzubinden und einen Datenaustausch zu gewährleisten. In Verbindung mit gespeicherten Datenbankprozeduren und besonderen Funktionen im System kann eine Implementierung vorgenommen werden. Die Schwerpunkte liegen dabei auf den gespeicherten Prozeduren der Datenbank und den dazugehörigen Funktionen im System, die in Komponenten gekapselt werden. In den gespeicherten Prozeduren werden die Parameter für die Funktionen aufbereitet und übergeben. Für jeden Integrationstyp existiert eine Komponente. Die Kapselung ist deshalb nötig, da sie direkt den Schnittpunkt mit der Fremdanwendung beinhaltet und so keine Abhängigkeiten in der Hauptanwendung existieren. Die Integrationskomponenten sind so konstruiert, dass sie mit geringem Aufwand bei Versionsupdates der Fremdsoftware aktualisiert werden können.

5.8.2.2 Richtlinien zur Einhaltung der Standards

Während der gesamten Entwicklung der Komponente muss darauf geachtet werden, dass die vorgegebenen Richtlinien und Standards, die das Konzept selbst definiert, exakt eingehalten werden. Schon geringe Abweichungen können später Probleme hervorrufen, die nicht mit vernünftigen Mitteln behebbar sind. Ein kritischer Rundblick in die heutige Softwareentwicklung zeigt, wie oft Behelfe konstruiert und programmiert werden, die zwar lauffähig erscheinen, aber desolat in der Wartung sind. Ein Grundsatz in diesem Konzept ist die Vermeidung solcher Hilfsmittel und der Einsatz von Komponenten, die nicht zum Standardsortiment des Frameworks gehören. Ist das reichliche Komponentenangebot des Frameworks nicht genügend, wird wiederum nur aus dem Standardsortiment die fehlende Komponente erstellt. Nur so wird gewährleistet, dass bei einem späteren Update der Anwendung eventuell auftretende Fehler minimiert werden und sich nur auf das eigene Produkt beziehen.

Aus diesem Grund wird z.B. für einen Report aus der Datenbank der Browser genutzt und keine Drittsoftware eingesetzt. Diese Philosophie erlaubt durch ein reines "XCopy-Deployment" das Verteilen der kompletten Software ohne Registrierungseinträge und Installationsroutinen. Wird das Programmverzeichnis entfernt, hinterlässt die Anwendung

keine Spuren auf dem Rechner. Das Ansprechen der Hardwareschnittstellen, wie COM1, COM2, LPT1 oder USB (Barcodescanner, Flachbettscanner) geschieht ohne OCX-Steuerelemente, die eine Registrierung notwendig machen. Diese Fakten werden durch das Framework unterstützt, welches ein Teil des Betriebssystems ist. Somit wird zusätzlich eine simple Verteilung und Aktualisierung der Anwendung garantiert.

Ein weiterer Aspekt, der zum Punkt Standardisierung gehört, ist die einheitliche und strukturierte Fehlerbehandlung über Komponentengrenzen hinweg. Schon das Programmierparadigma diktiert die Vorgehensweise und Behandlung bei Programmfehlern. Auch wenn Fehler in Funktionen oder Prozeduren als unwahrscheinlich angesehen werden, so muss ungeachtet dessen trotzdem eine Behandlung bei auftretenden Ausnahmesituationen erfolgen.

5.8.2.3 Grund- und Standardsoftware

Wie aus den Anforderungen abzuleiten ist, genügt eine saubere Betriebssysteminstallation mit einer stabilen Datenbankverbindung zur Inbetriebnahme der Anwendung. Dies setzt natürlich eine laufende Datenbank als Desktopengine für geringe Benutzerzahlen oder einen Datenbankserver für höhere Ansprüche voraus. Als Mindestvoraussetzung für das Betriebssystem gilt Windows XP inklusive .NET Framework und für den Server ebenfalls Windows 200x mit einer SQL-Datenbank, die von einem Framework unterstützt wird (stetig wachsende Anzahl der Datenbankprovider). Um die Qualität der Arbeit im anwendenden Unternehmen zu erhöhen, muss eine Officeanwendung und eine Bildbearbeitungssoftware vorhanden sein. Meistens sind diese Programme beim Anwender ohnehin installiert oder oft sogar schon beim PC-Kauf inbegriffen und stellen deshalb keine ungewöhnliche Anforderung dar. Bereits diese elementaren Voraussetzungen ermöglichen einen einwandfreien Einsatz der hier entwickelten Anwendung.

5.8.2.4 Terminals, sonstige und spezielle Geräte

Einige Komponenten der Systemsoftware benötigen zur Arbeitsoptimierung zusätzliche Hardwarekomponenten. So sind Scanner, Terminals oder mobile elektronische Datenerfassung verschiedenen Eingabemöglichkeiten, welche die Dateneingabe enorm beschleunigen. Die Barcodescanner kommunizieren beispielsweise über die serielle Schnittstelle und Flachbettscanner über den USB-Port. Terminals hingegen können als Arbeitsstation betrachtet werden, bei denen die Eingabe über Touch Screen und/oder über Kartenleser abläuft. Diese Hardware muss zuverlässig und robust konstruiert sein, um den jeweiligen Umwelteinflüssen im Einsatzbereich standzuhalten.

5.8.3 Projektüberwachung und Controlling

Die "Online-Entwicklung" begünstigt viele Faktoren außerordentlich, besonders die Überwachung und das Controlling. Die Überwachung der einzelnen Projekte ist einfach, da sie durch ihre Komponentenzuordnung transparent konstruiert und die Entwickler ständig vor Ort sind. Auch wenn Projekte parallel laufen, d.h. Komponenten gleichzeitig entwickelt werden, ist durch das angewandte Projektmanagementmodell die Transparenz gewährleistet. Hier sind die Verantwortlichkeiten und die Arbeitsaufteilung geregelt. In bestimmten zeitlichen Intervallen, die erst bei der Komponenteneinführung festgelegt werden, finden Besprechungen der Projektbeteiligten statt. Es werden Eindrücke und Verbesserungen diskutiert, die bisher durch Visiten oder Kontrollen nicht berücksichtigt wurden. Besonders nach der Einführung oder Aktualisierung einer Komponente stehen Visiten in diesen Bereichen auf der Tagesordnung der Entwicklungsleiter. Sie nehmen Stimmungen und Reaktionen der Anwender auf und leiten notwendige Schritte ein. Außerdem gehören zielgerichtete Gespräche zwischen Anwendern und Entwicklern über die Komponente zur Agenda. Der ständige Kontakt der Parteien wertet das Controlling deutlich auf, die schriftliche Fixierung der Besprechungsthemen und Visiten runden es ab. Eine durchdachte und auf das Notwendige konzentrierte Dokumentation begleitet das gesamte Projekt, denn aufwendige und überdimensionierte Dokumentationen verzögern und behindern das wesentliche Projektziel und sind schwer zu beurteilen.

5.8.4 Bewertung des Projektverlaufes

Zur Beurteilung des Projektes werden Ressourcen herangezogen, die bereits von vielen anderen zur Bewertung von Softwareprojekten genutzt werden. Professor Dr. Siegfried Seibert überarbeitete eine Bewertungsmethode von Steven C. McConnell, die frei zugänglich ist. Dieser Projekttest ist eine Übersetzung und wurde ursprünglich von Steven C. McConnell erstellt. Die Publikation dieses Bewertungsschemas steht auf der Survival Guide Website (Link siehe Literaturverweis) zur freien Verfügung. Die kostenfreie Erlaubnis zur Nutzung, Anpassung und Verteilung des Projekttestes wird vom Autor erteilt, so lange die vorliegenden Hinweise in allen Materialien aufgenommen werden und der Test nicht verkauft, lizenziert oder anderweitig zu kommerziellen Zwecken verwendet wird. [23]

Der nachfolgende Projekttest nach Professor Dr. Siegfried Seibert muss in bestimmten Intervallen erfolgen, um Korrekturen einleiten zu können. Nachfolgend werden Auszüge aus einem Projekttest wiedergegeben, welcher während der Komponentenentwicklung begleitend durchgeführt worden ist. Eventuelle Veränderungen, Anpassungen oder Ergänzungen werden gesondert gekennzeichnet.

Beschreibung: Der folgende Test dient zur Bewertung der Qualität des Managements von Softwareprojekten. Damit können die Chancen für einen erfolgreichen Projektabschluss ermittelt werden. Wenn der Test anzeigt, dass das Projektmanagement unzureichend ist, können gezielte Maßnahmen zur Verbesserung der Situation getroffen und Korrekturen eingeleitet werden.

Anleitung:

- Der Test besteht aus 33 Prüffragen aus den Bereichen Projektziele, Projektplanung, Projektüberwachung, Projektsteuerung, Risikomanagement und Projektteam.
- Geben Sie 3 Punkte für jede Frage, die Sie mit einem klaren "Ja" beantworten können, 0 Punkte für jedes eindeutige "Nein". Geben Sie anteilige Punkte, wenn eine Frage nicht eindeutig mit "Ja" oder "Nein" beantwortet werden kann, beispielsweise 2 Punkte für "wahrscheinlich" und 1 Punkt für "ein wenig ".
- Beantworten Sie die Fragen in frühen Projektphasen auf der Basis der vorliegenden Projektpläne. Später beantworten Sie die Fragen entsprechend der tatsächlichen Handhabung im Projekt.
- Die Punktesumme für alle Fragen wird mit einem Multiplikator für die Teamgröße angepasst. Der Multiplikator beträgt 1,5 bei bis zu 3 Vollzeitkräften, 1,25 bei 4 bis 6 Kräften und 1,0 bei 7 und mehr Kräften.
- Vergleichen Sie den erzielten Punktwert mit den Werten benachbarter Projektteams. Wiederholen Sie den Test in regelmäßigen Abständen (z. B. am Ende jeder Projektphase) und verfolgen Sie die erzielten Punktwerte im Projektverlauf.

Teamgröße: Anzahl der Teammitglieder und zwar umgerechnet auf Vollzeitkräfte, einschließlich Entwickler, Qualitätssicherungspersonal und Projektleiter.

Lfd. Nr.	**Punkte**	**Bewertungsfrage**
1		Hat das Projekt eine klare und eindeutige Zielsetzung oder Projektvision?
2		Glauben alle Teammitglieder, dass die Projektvision realistisch ist?
3		Gibt es für das Projekt eine Wirtschaftlichkeitsbetrachtung, die den geschäftlichen Nutzen des Projekts belegt und aufzeigt, wie dieser Nutzen gemessen wird?
4		Gibt es einen Prototyp der Benutzerschnittstelle, der die Funktionalität des zu realisierenden Systems erkennbar demonstriert?

5		Hat das Projekt eine schriftliche Spezifikation (Lastenheft/Pflichtenheft), was das System genau leisten soll?
6		Hat das Projektteam Endbenutzer des Systems bereits in frühen Projektphasen interviewt und sie im weiteren Projektverlauf kontinuierlich in das Projekt einbezogen?

Abbildung 5-21 Projekt-Bewertungsfragen 1

7		Hat das Projekt einen detaillierten, schriftlichen Software-Projektplan?
8		Enthält der Projektplan die Erstellung eines Installationsprogramms, die Datenübernahme aus früheren Systemversionen, die Integration von/in Fremdsoftware, Treffen mit Kunden und andere, häufig übersehene "kleinere" Aufgaben?
9		Wurden die Termin-, Aufwands- und Kostenschätzungen am Ende der jüngsten Projektphase aktualisiert und die Aktualisierung gegengeprüft?
10		Hat das Projekt eine detaillierte, schriftliche Dokumentation des Systemdesigns und der technischen Systemarchitektur?
11		Hat das Projekt einen detaillierten, schriftlichen Qualitätssicherungsplan, der neben Softwaretests auch Entwurfs- und Code-Inspektionen vorsieht?
12		Hat das Projekt einen detaillierten Fertigstellungsplan, der beschreibt, in welchen Versionen welche Funktionalitäten implementiert werden?
13		Sind im Projektplan Urlaubstage, Feiertage, Krankheitstage, Schulungen und andere Abwesenheiten berücksichtigt und sind die verfügbaren Kapazitäten/Ressourcen zu weniger als 100% verplant?
14		Wurde der Projektplan, einschließlich der Terminierung, von allen für die Durchführung der Arbeiten verantwortlichen Personen (Entwickler, Designer, Qualitätssicherer, Dokumentationsteam etc.) gesehen und gebilligt?

Abbildung 5-22 Projekt-Bewertungsfragen 2

15		Hat das Projekt genau definierte Detail-Meilensteine ("Mini-Meilensteine"), die entweder als 100% erledigt oder als 100 % unerledigt betrachtet werden?
16		Kann ein Projektinteressent ("Stakeholder") leicht herausfinden, welche dieser Mini-Meilensteine fertig gestellt sind?
17		Gibt es im Projekt eine Feedback-Möglichkeit, über die Teammitglieder Probleme anonym an das übergeordnete Management melden können?
18		Sind aktuelle Projektstatusinformationen (einschließlich Aufwands- und Terminschätzungen, Aufgabenverteilung, Projektfortschrittsberichte und Soll-Ist-Vergleiche) für jedes Teammitglied verfügbar?

Abbildung 5-23 Projekt-Bewertungsfragen 3

19		Wurde für das Projekt ein verantwortlicher, übergeordneter Manager mit genügend Entscheidungskompetenz benannt und hat das Projekt dessen aktive Unterstützung?
20		Erlaubt es die Arbeitsbelastung des übergeordneten Managers, dem Projekt einen angemessenen Zeitanteil zu widmen?
21		Hat das Projekt eine schriftliche Verfahrensrichtlinie, wie Änderungen an der gültigen "Baseline" zu prüfen, genehmigen und umzusetzen sind?
22		Hat das Projekt ein Änderungskontrollgremium mit der abschließenden Autorität, Änderungen zu genehmigen oder zu verwerfen?
23		Unterliegt der Quellcode einer vollständigen, automatisierten Versionskontrolle?
24		Sind die für eine erfolgreiche Projektbearbeitung erforderlichen Tools verfügbar (einschließlich Fehlerverfolgung, Versionskontrolle und Projektmanagement)?

Abbildung 5-24 Projekt-Bewertungsfragen 4

25		Enthält der Projektplan eine priorisierte Liste der Projektrisiken? Wird die Liste regelmäßig überprüft und aktualisiert?

26		Wurde im Projektteam ein Risikoverantwortlicher benannt, der sich um das Erkennen, Beurteilen und Verfolgen von Risiken kümmert?
27		Wenn im Projekt Unteraufträge fremdvergeben werden, gibt es einen Plan und eine verantwortliche Person zum Management jedes Unterauftragnehmers? (Volle Punktzahl, wenn keine Unteraufträge vergeben werden)

Abbildung 5-25 Projekt-Bewertungsfragen 5

28		Besitzt das Projektteam alle technischen Fachkenntnisse, die für Bearbeitung des Projekts bis zu dessen vollständigem Abschluss erforderlich sind?
29		Hat das Projektteam Fachkenntnisse im Anwendungsbereich, in dem das System eingesetzt wird?
30		Hat das Projekt einen vom Team akzeptierten Leiter, der in der Lage ist, die Projektaktivitäten erfolgreich zu koordinieren?
31		Gibt es genügend Personen, um alle erforderlichen Arbeiten durchzuführen?
32		Stimmt die Kommunikation im Team? Arbeiten alle Projektbeteiligten gut zusammen?
33		Setzt sich jede beteiligte Person voll für das Projekt ein?

Abbildung 5-26 Projekt-Bewertungsfragen 6

Bewertungsergebnis:

- Vorläufiger Punktwert (Summe der Einzelpunkte)
- Endgültiger Punktwert (Vorläufiger Wert, der aufgrund der Teamgröße angepasst wurde)

Punkte	**Erfolgschance**	**Beurteilung**
0 - 39	Schlecht	Ein Projekt mit dieser Bewertung hat deutliche Schwächen in Bezug auf Projektziele, Projektplanung, Projektverfolgung, Risikomanagement und Projektteam. Die Hauptfrage in dieser Kategorie ist, ob ein solches Projekt überhaupt zu Ende geführt werden kann und soll.

Ab 40	Befriedigend	Die Bewertung ist durchschnittlich. Ein solches Projekt muss hohe Belastungen und eine wechselhafte Teamdynamik verkraften. Das Endergebnis hat wahrscheinlich eine geringere Funktionalität und Qualität und wird mehr kosten und später ausgeliefert werden als erwartet. Projekte in dieser Kategorie können von einem professionelleren Projektmanagement am meisten profitieren.
Ab 60	Gut	Eine Bewertung in dieser Kategorie liegt bereits über dem Durchschnitt. Ein solches Projekt hat gute Chancen, seine Produktziele entweder in der festgelegten Zeit- oder in der festgelegten Kostenbandbreite zu erreichen, aber wahrscheinlich nicht alles gleichzeitig.
Ab 80	Sehr gut	Ein Projekt auf dieser Stufe liegt deutlich über dem Industriedurchschnitt. Ein solches Projekt hat eine hohe Wahrscheinlichkeit, sowohl seine Produkt- als auch seine Termin- und Kostenziele annähernd zu erreichen.
Ab 90	Hervorragend	Bei einem Projekt mit dieser Bewertung ist der Erfolg praktisch garantiert. Das Projekt wird ohne wesentliche Einschränkungen professionell gemanagt. Seine Produkt-, Termin-, Kosten- und sonstigen Ziele werden mit höchster Wahrscheinlichkeit voll erreicht.

Abbildung 5-27 Ergebnisübersicht

Ergebnis-Interpretation:

In der Praxis schneiden die meisten Projekte (noch) befriedigend ab. Viele erreichen noch nicht einmal 50 Punkte. Die folgende Tabelle zeigt, wie die Ergebnisse zu interpretieren sind.

- Wenn 60 oder mehr Punkte erreicht werden, befindet sich das Projekt in einer guten Verfassung.
- Wenn weniger als 60 Punkte erreicht werden und Verbesserungen beschlossen werden, die auch durchgeführt werden, ist die Situation ebenfalls noch akzeptabel.
- Bedenklich ist die Situation, wenn weniger als 60 Punkte erreicht werden, und keine Verbesserungsmaßnahmen eingeleitet werden oder das Projektteam den festgelegten Verbesserungsmaßnahmen nicht folgt.

6 UMSETZUNG DES KONZEPTES IN DER PRAXIS

Das in Kapitel 5 vorgestellte Konzept ist komponentenunabhängig und dient als Vorlage für die Entwicklung beliebiger Komponenten. Die Realisierung der allgemeingültigen Grundlagen wie Basisarchitektur, Datenbank, Logik und Projektplan gemäß Abbildung 5-2 ist in Kapitel 5.2 hinreichend beschrieben, so dass diese als existent vorausgesetzt werden und deshalb für die Beschreibung der Umsetzung unberücksichtigt bleiben. Die hier zu definierende Umsetzung bezieht sich gemäß Abbildung 6-1 auf die Entwicklung der einzelnen Komponenten.

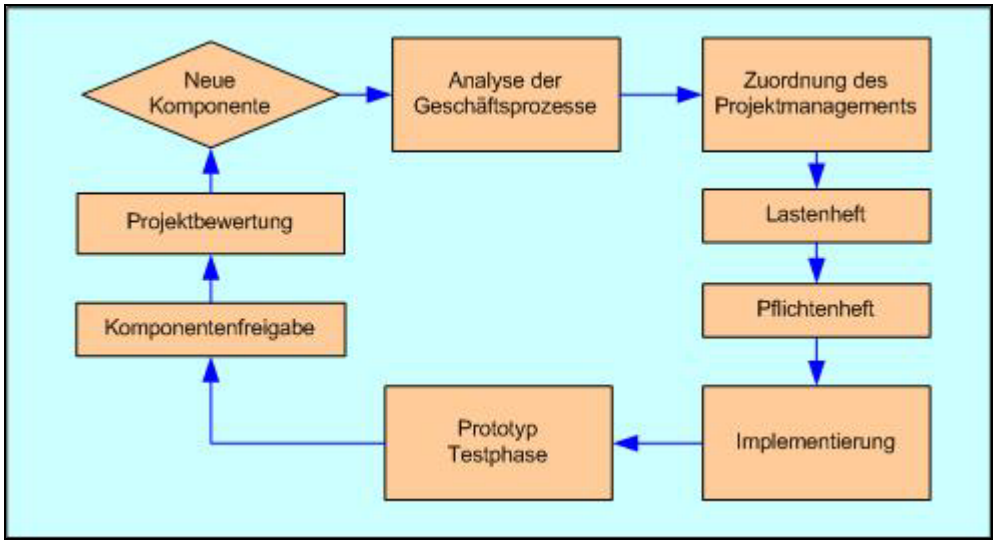

Abbildung 6-1 Entwicklungsprozess einer Komponente

Die Umsetzung des Konzeptes auf alle Komponenten eines Unternehmens würde den Umfang einer Dissertation sprengen. Deshalb wird das Konzept beispielhaft an einer umfangreichen Komponente demonstriert, die nahezu alle Möglichkeiten der Komponentenentwicklung beinhaltet. Bezüglich der Abläufe, Strukturen und Prozeduren spielt die Wahl der Komponente eine untergeordnete Rolle, da diese in jeder Komponente, abgesehen von einigen Besonderheiten oder Erweiterungen, gleich sind. Im Rahmen der Beschreibung für die Umsetzung wird auf die Nennung eventuell notwendiger Hard- und /oder Software für die Realisierung verzichtet, da der Schwerpunkt des Konzeptes die Komponenten-Entwicklung beinhaltet.

6.1 AUSWAHL EINES KLEINEN MITTELSTÄNDISCHEN UNTERNEHMEN

Als Hersteller von Maschinenbauteilen bietet der hier ausgewählte Anwender (vgl. Abbildung 2-1, Kandidat A) seinen Kunden alle Arten von spanenden Bearbeitungen an,

wie Drehen, Fräsen, Schleifen, Nuten und Bohren und die Herstellung von Schweißkonstruktionen. Das Unternehmen führt alle relevanten Tätigkeiten der Produktherstellung von der Konstruktion bis zur Endmontage durch einschließlich der Erstellung aller Kundenzeichnungen. Die Kunden stammen aus den unterschiedlichsten Bereichen, dazu gehören Sondermaschinen-, Anlagen- und Schiffsbau, Schienen-Transportwesen und Gleisbau sowie Zulieferer der Automobilindustrie.

Der Autor hat mit diesem mittelständischen Unternehmen in Zusammenarbeit mit dem Institut für Ingenieurinformatik an der Universität Essen seit Januar 1998 das Konzept gemäß Kapitel 5 entwickelt und auch erfolgreich umgesetzt. Die in Zusammenarbeit mit der Universität entstandene individuelle Systemsoftware basiert auf aktuellen technologischen Erkenntnissen und ist seit Anfang 2002 komplett im Einsatz. Weit über 50 Anwender unterschiedlicher Grundvoraussetzungen nutzen diese Software. Im Lager, in der Fertigung, Konstruktion, Arbeitsvorbereitung und im Officebereich findet die Anwendung ihren Einsatz. Eine konsequente Umsetzung in allen Bereichen ist nur mit einer Unternehmensführung zu bewerkstelligen, die innovativ und fortschrittlich denkt und handelt und somit einen Grundstein für ihre künftige Wettbewerbsfähigkeit legt.

Wie das erarbeitete Konzept angewendet und umgesetzt wird, zeigen schrittweise die folgenden Abschnitte der Realisierung.

6.2 Prüfung der Randbedingungen für die „Online-Entwicklung"

Die Auftragsbearbeitung ist einer der sensibelsten Bereiche der Geschäftsprozesse und für die Umsetzung des Konzeptes sehr gut geeignet. Hier bestehen Berührungspunkte mit der Fertigung, dem Kunden und den Lieferanten. Es ist sehr sorgfältig zu prüfen, wo verstärkt Komplikationen entstehen können. An Hand des Prozessablaufplanes ist erkennbar, wo diese empfindlichen Punkte zu erwarten sind. Eine Auflistung erfolgt im nächsten Abschnitt. Darüber hinaus muss hier ein altes System abgelöst werden, welches eine Auftragsbearbeitung besitzt und deshalb gezwungener Maßen zeitweise parallel gearbeitet werden muss. Aufträge, die im alten System eingepflegt sind, konnten nur oberflächlich importiert werden. Diese importierten Daten haben nur informativen Charakter. Der Import ist deshalb uninteressant, da der vorhandene Datenaufbau in keiner Weise kompatibel zu heutigen Strukturen ist.

Demzufolge werden zu einem Stichtag alle neuen Aufträge in das neue System aufgenommen und der vorhandene Datenbestand im Altsystem bis zum Beenden des jeweiligen Vorgangs weiter gepflegt. So ist ein sauberer Datenabschluss im Altsystem gewährleistet und einer vollständigen Informationsbeschaffung über vergangene Vorgänge steht nichts im Weg. Wenn die kompletten Abläufe eines Auftrages im System abgeschlossen

werden, d.h. Rechnungen, Lieferungen, Buchungen usw., ist auch die Quelle für die Auskunft eindeutig und die Daten liegen nicht verstreut in verschiedenen Datenbanken. Zunächst müssen die sensiblen Schnittstellen im Geschäftsprozess der Auftragsbearbeitung aufgelistet werden:

Datenaufnahme: Kein Einfluss auf den Geschäftsprozess.

Dateneingabe: Geringfügiger Einfluss auf den Geschäftsprozess, durch die Datenvalidierung bei der Dateneingabe. Liefertermin, Kunde, Artikel, Menge sind Pflichtfelder, wobei der Liefertermin ein korrektes Format fordert (KW oder Datum).

Arbeitsplan wählen/verknüpfen: Starken Einfluss auf den Geschäftsprozess, da sich der Arbeitsplan als Basis für alle Fertigungsdokumente der Kommission darstellt. Hier besteht ein Fehlerrisiko durch inkorrekte Arbeitspläne, die den Prozessfluss der Auftragsbearbeitung beeinträchtigen, d.h. Arbeitspläne, die Fehler aufweisen, müssen dann durch die zuständigen Mitarbeiter erst korrigiert werden. Diese Unterbrechungen sind anhaltend und nachhaltig.

Daten speichern: Kein Einfluss auf den Geschäftsprozess.

Fertigungspapiere ausgeben: Starker Einfluss auf den Geschäftsprozess, da die vollständigen Papiere für die Beschaffung und die Fertigung benötigt werden. Treten Komplikationen auf, sind Verzögerungen oder Unklarheiten in der Fertigung nicht ausgeschlossen.

Fertigung: Kein direkter Einfluss auf den Geschäftprozess, aber es nicht ausgeschlossen, dass in der Fertigung Störungen durch die "Online-Entwicklung" auftreten können, die sich auch auf die Auftragsbearbeitung negativ auswirken.

Lieferschein, Frachtbrief, Rechnung usw. schreiben: Teilweise starker Einfluss auf den Geschäftsprozess, da z.B. Speditionen bzw. Abholer auf die Ware warten und Frachtpapiere und Lieferscheine notwendig sind. Zu bestimmten Zeiten ist auch ein pünktlicher Rechnungsausgang unumgänglich.

Plausibilitätsprüfung: Mäßiger, aber nachhaltiger Einfluss auf den Geschäftsprozess. Falsche oder unvollständige Lieferscheine oder Rechnungen beeinträchtigen oder verfälschen Buchungs- und Abrechnungsverfahren, der Auftrag kann nicht sauber zum Abschluss gebracht werden.

6.3 Analyse der unternehmensspezifischen Prozesse und Strukturen

Die gewählte Komponente *Auftragsbearbeitung* dient hier als Prototyp für die Umsetzung. Eine beispielhafte Anwendung des Konzeptes mit einer Stammdatenkomponente, wie Ar-

tikel, Zeichnungen, Kunden oder DIN-Teile ist zu einfach und gibt nur wenig Auskunft über die Vorzüge des Konzeptes. Die Untersuchung der Infrastruktur ist zwar Teil des Konzeptes, aber nicht Inhalt dieser Arbeit. Die Abbildung 6-2 zeigt die bisherige Vorgehensweise in der Auftragsbearbeitung. Anhand dieses Flussdiagramms wird die IST-Situation analysiert. Die Überschneidung der innerbetrieblichen Prozesse wird hier besonders in Verbindung mit dem Fertigungsprozess deutlich.

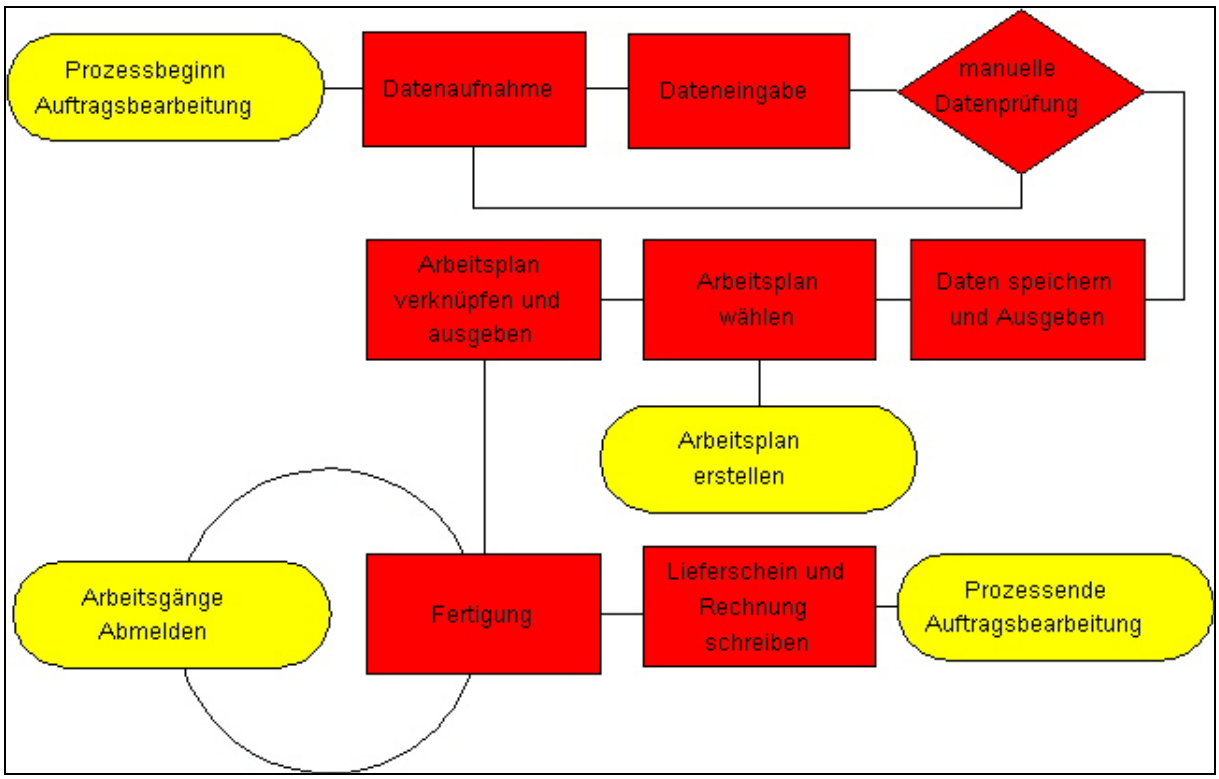

Abbildung 6-2 Schematische Darstellung der Auftragsbearbeitung

Vor Beginn der gesamten Entwicklung existierten bei dem Anwender bereits Richtlinien und Anordnungen im Umgang mit bestimmten Geschäftsprozessen. Jeder Mitarbeiter wurde eingewiesen und mit dem jeweiligen Dokument vertraut gemacht und er musste diese Einweisung quittieren. Regeln, wie Lieferschein und Rechnung schreiben, Arbeitspläne übernehmen, Altaufträge aktivieren oder Kommissionsbuch drucken, waren schriftlich ausgearbeitet und fixiert. So auch für die Auftragserfassung und Auftragsbearbeitung, die hier nachfolgend aufgeführt ist und als Referenz und zur Analyse dient. Nachfolgend ist der bisherige Prozessablauf für die Auftragsbearbeitung gemäß Abbildung 6-2 stichpunktartig wiedergegeben, beginnend mit dem Auftragseingang:

- Eingabe der bekannten Daten, mündlich oder schriftlich ohne Plausibilitäts-, Redundanz und Konsistenzprüfung
- Auftragsdaten speichern und Auftragskarte für die Werkstatt ausdrucken

- Vorhandenen Arbeitsplan an Hand der Zeichnungsnummer übernehmen; falls nicht existierend, neu erstellen lassen
- Arbeitsplan ausdrucken (Etiketten) und an der Auftragskarte fixieren
- Arbeitsgänge laut Arbeitsplan ausführen
- Kommissionslisten, die an den Maschinen liegen, bei erledigtem Arbeitsgang ausfüllen (Stückzahl, Zeit usw.)
- Zum Wochenende die Daten aus den Listen in die Datenbank eintragen
- Nach Abschluss der Fertigung Lieferschein- und Rechnungserstellung
- Auftragsstatus auf erledigt setzen

Zur Vervollständigung des IST-Zustandes der Auftragsbearbeitung werden nun ergänzend die Besonderheiten zu den einzelnen Teilprozessen erläutert:

Datenaufnahme: Die Datenaufnahme kann telefonisch, per Fax oder Brief, aber auch persönlich, also auf jede erdenkliche Art und Weise erfolgen (Hinweis: diese Möglichkeit ist sogar bei neuen Systemen keine Selbstverständlichkeit).

Dateneingabe: Die Mindestanforderungen für die Speicherung eines Auftrages sind die Kommissionsnummer, die Kundennummer und der Liefertermin, die ohne Datenprüfung durch das Programm oder die Datenbank akzeptiert werden, lediglich leere oder nicht numerische Felder werden nicht akzeptiert. Beim Liefertermin sind alle Eingaben möglich. Die Kommissionsnummer muss manuell vergeben werden.

Manuelle Datenprüfung: Dies bedeutet, die Verantwortung liegt in der Hand des Anwenders. Es werden keine Plausibilitäts-, Redundanz und Konsistenzprüfungen vom Programm oder der Datenbank vorgenommen.

Auftragsdaten speichern und ausgeben: Die Speicherung des Auftrages erfolgt nach manueller Datenprüfung, die Felder der Kundennummer und des Liefertermins dürfen nicht leer sein. Anschließend kann die Auftragskarte für die Fertigung ausgedruckt werden. Die Ausgabe kann nicht überprüft oder nachvollzogen werden, d.h. ungewollte Mehrfachausdrucke sind möglich.

Arbeitsplan wählen: Zur Vervollständigung der Fertigungspapiere gehört auch ein kommissionsabhängiger Arbeitsplan. Mit Hilfe der im Auftrag manuell eingegebenen Zeichnungsnummer wird nach Arbeitsplänen im System gesucht. Bei Syntaxunterschieden in der Zeichnungsnummer (z.B. Leerzeichen) verläuft die Suche erfolglos; hierin liegt eine enorme Gefahr der doppelten Erstellung von Arbeitsplänen. Weiterhin können auch keine ähnlichen Arbeitspläne erkannt werden, da keine Möglichkeit der Klassifizierung existiert.

Arbeitsplan verknüpfen und ausgeben: Ist ein geeigneter Arbeitsplan im System verfügbar, wird dieser mit dem aktuellen Auftrag verknüpft, dabei wird die vorhandene Verknüpfung, trotz Informationsverlust, überschrieben. Existiert kein passender Plan, muss er von den zuständigen Personen erstellt werden. Der Inhalt des Arbeitsplanes muss immer auf die jeweilige Kommission abgestimmt werden, da die Daten der vorangegangenen Kommission entsprechen. Historien gehen so verloren und eine enorme Fehlerquelle besteht darin, dass alte Informationen von abgeschlossenen Aufträgen automatisch übernommen und versehentlich nicht angepasst werden. Die Ausgabe des Arbeitsplanes erfolgt über große Klebeetiketten, die nach dem Druck auf die Arbeitskarte geklebt werden. Probleme bei längeren Arbeitsplänen oder häufigen Arbeitsplanänderungen machen die Unterlagen unübersichtlich und fehlerbehaftet, die Folge ist eine fehlende Transparenz. Zudem fehlen auf allen Fertigungspapieren die Informationen über den Druckvorgang.

Fertigung: Die ausgestellten Fertigungspapiere gehen bei Produktionsbeginn in die Fertigung und begleiten das Werkstück kontinuierlich zu den Kostenstellen, verschlissene und unleserlich gewordene Papiere sind dabei nicht auszuschließen. Zudem ist der Aufenthaltsort der Papiere stets unbekannt. Diese Tatsachen sind von allgemeiner Bedeutung und nicht EDV-unabhängig, Ideen für ein neues Verfahrensmodell sind somit gefragt. An jeder Maschine liegt eine Kommissionsliste mit den zu erwartenden Aufträgen, passiert die Kommission die Maschine und der jeweilige Arbeitsgang ist abgeschlossen, wird das in dieser Liste vermerkt. Die eingetragenen Daten der Liste werden zum Wochenende mit der Datenbank abgeglichen. Wurden alle Kostenstellen durchlaufen, ist die Fertigung beendet.

Lieferschein und Rechnung schreiben: Die Dokumente können unabhängig vom Auftrag geschrieben werden, es wird keine Plausibilität geprüft. Mehrfachausführungen der Dokumente sind ohne Warnung möglich. Wurden Lieferschein und Rechnung erzeugt, änderte sich der Auftragsstatus (in erledigt), falls die Restmenge Null ist. Der Auftragsstatus kann zusätzlich manuell geändert werden.

Die Skizzierung des Geschäftsprozesses vor der Komponentenentwicklung genügt an dieser Stelle. Die Einschätzung und Optimierung des Geschäftsprozesses ist somit durchführbar.

6.3.1 Einschätzung und Optimierung der Unternehmensorganisation

Unter Zuhilfenahme der Analyse des Geschäftsprozesses Auftragsbearbeitung wird nun eine Beurteilung dieses Prozesses durchgeführt. Bei dieser Bewertung wird gleichzeitig der Optimierungsbedarf mit einbezogen. In den Grundzügen entspricht die Auftragsbear-

beitung des Anwenders den branchenüblichen Richtlinien. Das Auftragswesen ist dem des Lohn- und Einzelfertigers angelehnt. Vereinzelte Abläufe sind jedoch verbesserungsbedürftig. Im Ganzen betrachtet muss eine höhere Transparenz erzielt werden, da Teilbereiche den Informationsgehalt beeinträchtigen. Positiv hervorzuheben ist die flexible Datenaufnahme für Aufträge. Ein optimierter Prozessablauf der Auftragsbearbeitung beginnend mit dem Auftragseingang, sieht gemäß Abbildung 6-3 folgendermaßen aus:

- Eingabe der bekannten Daten mit Plausibilitäts-, Redundanz und Konsistenzprüfung sowie Konfliktmeldung
- Bei der Selektion des Artikels wird die entsprechende mengenabhängige Variante des Arbeitsplanes übernommen
- Auftragsdaten speichern
- Terminierung und Generierung des Fertigungsauftrages
- Fertigungspapiere ausgeben (Beschaffungsliste, Produktbegleitschein, Fertigungsauftrag, Auftragskarte, Zeichnungen und sonstige Dokumente)
- Arbeitsgänge laut Fertigungsauftrag ausführen
- Stati der Arbeitsgänge melden, Personalnummer, Stückzahl, Meldetyp (Aktivieren, Unterbrechen, Wiederaufnehmen, Abmelden)
- Nach Abschluss der Fertigung Erstellung von Lieferschein und Rechnung, beim Druck der Rechnung wird nach Prüfung mit Meldung der Auftragsstatus auf beendet gesetzt.

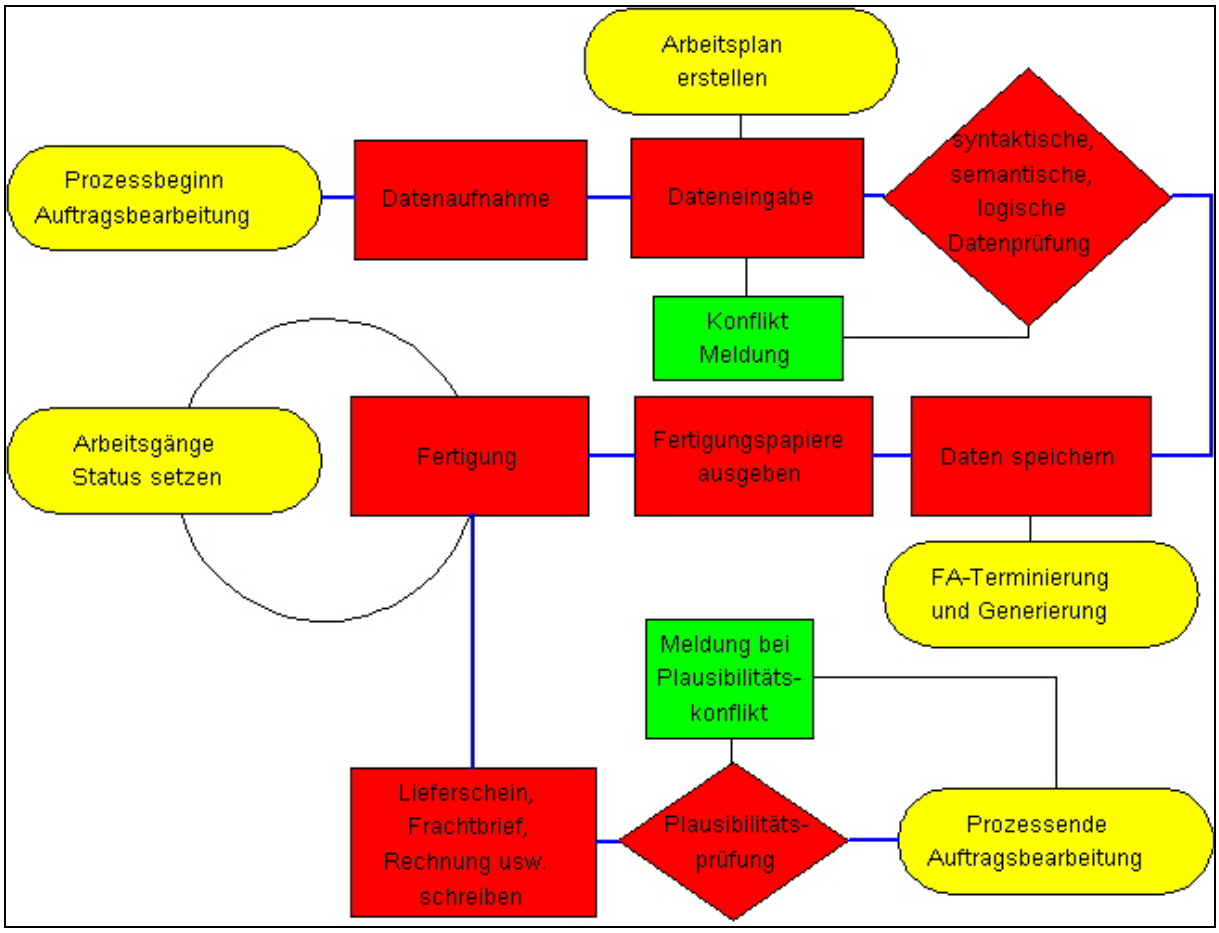

Abbildung 6-3 Optimierter Auftragsprozess

Optimierungen, welche die Schnittstellen zu anderen Prozessen betreffen, z.B. der Fertigung, werden der Vollständigkeit halber und des durchgängigen Verständnisses wegen mit erwähnt. An dieser Situation ist schon zu erkennen, dass die Verzahnung der meisten Geschäftsprozesse intensiv ausfällt.

Einige Prozesse wurden umorganisiert bzw. erweitert, um eine transparentere Informationspolitik zu betreiben. Das Verfahren im Umgang mit den Fertigungspapieren wurde vollständig überarbeitet und umgestellt. Abgeschlossene Aufträge und Fertigungsaufträge werden archiviert und somit eine zügige Informationsbeschaffung der Altdaten garantiert. Die an wichtigen Stellen eingefügten Prüfungen setzen den Benutzer über Widersprüche oder Fehleingaben in Kenntnis. Bei der Dateneingabe werden bereits wichtige Felder geprüft. Ein Mindestmaß an bekannten Daten ist jedoch Bedingung, da sonst keine ordnungsgemäße Funktionalität gewährleistet und der Informationsfluss nachhaltig beeinträchtigt ist. Die von den Auftragsdaten abhängigen Prozesse, die während der Auftragsbearbeitung manuell oder automatisch angestoßen werden, benötigen zudem eine ordnungsgemäße Datenbasis als Ausgangspunkt. Im Folgenden werden die einzelnen Teilprozesse mit den Optimierungen hervorgehoben.

Datenaufnahme: Die Datenaufnahme hat sich im Vergleich zum vorhergehenden Prozess nicht geändert. Die Annahme erfolgt weiterhin in beliebiger Form.

Optimierungen: nicht notwendig

Dateneingabe: Die Voraussetzung für die Speicherung eines Auftrages sind die korrekte Kommissionsnummer, die Kundennummer, der Liefertermin, die Auftragmenge und der Artikel. Die Kommissionsnummer wird dabei automatisch von der Datenbank vergeben und ist eindeutig. Mit einer Datenvalidierung durch die Anwendung bzw. durch die Datenbank werden Fehler nicht akzeptiert. Der Liefertermin muss im Format KW oder Datum eingegeben werden, da sonst eine Terminierung nicht möglich ist. Eine weitere Voraussetzung für die Terminierung ist ein korrekter Arbeitsplan und die Struktur der Kostenstellen und der Arbeitsgänge müssen ordnungsgemäß und schlüssig aufgebaut sein.

Optimierungen: Automatische Kommissionsnummernvorgabe; Arbeitsplanprüfung; Datenaufbereitung für die Transaktion in die Datenbank

Syntaktische, semantische und logische Datenprüfung: Bevor die Daten in der Datenbank abgelegt werden, findet eine Kontrolle im Programmcode bezüglich Format und Inhalt statt. Nach der erfolgreichen Überprüfung geschieht die Kontrolle der Folgerichtigkeit, Widerspruchsfreiheit und vorhandener Existenz in der Datenbank. Bei schwerwiegenden Vergehen wird das Speichern verweigert, im anderen Fall erfolgt eine Meldung an den Benutzer. In gesonderten Situationen übernimmt eine Autokorrektur in der Datenbank die Berichtigung (Bsp.: Die Eingabe eines vergangenen Datums für den Liefertermin ergibt keinen Sinn, die Datenbank ersetzt es durch das gegenwärtige Datum).

Optimierungen: Der komplette Validierungsablauf wurde von manuell auf datentechnisch umgesetzt

Daten speichern: Nur nach erfolgreicher Datenprüfung werden die Informationen in der Datenbank abgelegt. Der Prozess Fertigungsauftrag terminieren und generieren kann gestartet werden, um den weiteren Prozessfluss sicherzustellen.

Optimierungen: Fehlerminimierung durch neue Validierung erreicht

Fertigungspapiere ausgeben: Die Unterlagen für die Produktion stehen nun im System zur Verfügung. Papiere können zentral und zeitunabhängig inklusive Druckinformation (z.B. wer und wann gedruckt) ausgegeben werden. Ein Mehrfachausdruck ist möglich, wird aber registriert und es wird darauf hingewiesen. Die Standarddokumente sind Produktbegleitschein, Auftragskarte, Fertigungsauftrag (FA-Arbeitsplan) und Beschaffungsliste, sie werden in eine wetterfeste Klarsichtmappe gesteckt. Lediglich der Produktbegleitschein befindet sich ständig beim Werkstück, alle anderen Dokumente verbleiben in

der Schutzmappe und werden an zentralen Punkten (z.B. Terminals) in der Fertigung aufbewahrt.

Optimierungen: Umstrukturierung des Verfahrens mit Fertigungsunterlagen für schonenderen Dokumentenumgang und zusätzliche Druckinfo auf den Papieren mit Datum, Uhrzeit und Benutzer (neues Verfahrensmodell); übersichtlicherer Dokumentenaufbau; keine besonderen Papiersorten wie Etiketten, d.h. Kosten sparend.

Fertigung: Die prägnantesten Neuerungen der Fertigung haben zwar nicht direkt mit dem Prozess der Auftragsbearbeitung zu tun, beeinflussen ihn aber entscheidend und müssen deshalb erwähnt werden. Neu ist, dass sich nur der Produktbegleitschein ständig am Werkstück befindet und dass der Status der Arbeitsgänge sofort gemeldet werden kann (Terminal, elektronische Datenerfassung oder Anwendung). Das hat zur Folge, dass bei abgeschlossener Fertigung sofort reagiert werden kann und die Auftragsbearbeitung fließend fortgeführt wird.

Optimierungen: Änderung der Arbeitsgangstati ohne Zeitversatz; Fertigungsunterlagen zentral gelagert (neues Verfahrensmodell); Übersicht über Arbeitsgangstati zu jeder Zeit und an jeder Stelle; transparentere Fertigung; flüssigerer Auftragsbearbeitungsprozess; die Prozesse Auftragsbearbeitung und Fertigung sind strikter getrennt, besitzen aber saubere Schnittstellen

Lieferschein, Frachtbrief, Rechnung usw. schreiben: Diese Dokumente sollten kommissionsabhängig geschrieben werden, können aber auch unabhängig erstellt werden. Die Mehrfachausgabe ist weiterhin möglich, der Benutzer wird aber darauf hingewiesen und der Druck sensibler Dokumente (z.B. Rechnung und Lieferschein) wird in der Datenbank registriert.

Optimierungen: Schnelle Rechnungs- und Lieferscheinerstellung; beliebige Ausgabeformate, Druckregistrierung

Plausibilitätsprüfung: Zusätzlich muss in die Auftragsbearbeitung eine Prüfung eingefügt werden, die den Status des Auftrages, Liefermengen, Rechnungssummen und Buchungsdetails betrifft.

Optimierungen: Die komplette Plausibilitätsprüfung trägt zum abgerundeten Informationsgehalt bei und vereinfacht den Folgeprozess Buchungen

6.4 VORBEREITUNG DER KOMPONENTENENTWICKLUNG

In Anlehnung an Kapitel 5.8 wird die Komponenten-Entwicklung für die Auftragsbearbeitung vorbereitet. Zunächst wird ein Projektteam definiert, welches gemeinsam das Lasten- und Pflichtenheft erstellt, wodurch auch gleichzeitig später eventuell auftretende Kommunikationsprobleme verhindert werden können.

6.4.1 Erstellung des Lastenheftes

Die Notwendigkeit eines Lastenheftes sind in Kapitel 5.8.1 hinreichend erklärt. Im Einzelnen müssen die folgenden Kriterien aufgeführt werden:

Einführung in das Projekt (Kosten, Personal, Termine usw.): Die Komponente Auftragsbearbeitung soll in das bereits laufende System implementiert werden. Zurzeit befindet sich der Artikel-, Kunden- und Lieferantenstamm im Einsatz. Die Arbeitspläne wurden bereits erprobt und werden größtenteils mit dem neuen System erstellt. Änderungen dieser Komponente haben nur indirekt Einfluss auf die Auftragsbearbeitung und können parallel vorgenommen werden. Demzufolge sind die Voraussetzungen für die Einführung der Auftragsbearbeitung gegeben. Auswertungen oder Reports können nach Bedarf schnell zur Verfügung gestellt werden, da sie nicht mit der Softwarekomponente selbst, sondern nur über die Datenbank generiert werden.

Personeller Aufwand: Es werden ca. 200 Mannstunden angesetzt.

Zeitlicher Rahmen: Die Entwicklung erstreckt sich über ca. 4 Wochen.

Finanzieller Aufwand: Für die Kosten wird zunächst nur das Personal herangezogen, da die notwendigen Hilfsmittel ohnehin zur Verfügung stehen.

Ausgangssituation (IST): Momentan werden alle Aufträge im Altsystem erstellt und abgewickelt, die Auftragsbestätigungen erfolgen datenbankunabhängig.

Aufgabenstellung (SOLL): Die Komponente Kundenaufträge muss im neuen System erstellt werden, die Datenbasis Kunden und Artikel steht zur Verfügung. Die Art und Weise der Programmierung soll dem aktuellen Stil entsprechen. Die vorgegebenen Attribute und Eigenschaften müssen Bestandteil des Auftrages sein.

Vorgaben: Für die zukünftige Auftragsbearbeitung sind mindestens die nachfolgenden Attribute vorzusehen:

- Kommissionsnummer
- Ihr Zeichen

- Auftragsmenge
- Bestellnummer
- Bestelldatum
- Kriterien für Listensortierung
- Einheit (Stück, Paar usw.)
- Artikel
- Kunde
- Preis
- Zuschläge
- Liefertermin
- Prioritätshinweis
- Informationen für die Arbeitskarte
- Anmerkungen
- Benutzerinformationen (erstellt, geändert)

Schnittstellen: Der Auftrag wird ohne besondere Formblätter oder Vordrucke entgegengenommen und in die Anwendung eingegeben. Der Auftrag wird schriftlich bestätigt. Dabei sind Standardtexte laut Auftragsabwicklung zu verwenden.

Anforderung an die Systemtechnik: Hier ist die Handhabung der zukünftigen Software etwas genauer zu definieren. Die nachfolgenden Punkte sind verbindlich:

- Bedienung der Kunden und Pflege der Stammdaten
- Annahme des Auftrages
- Bearbeitung/Änderung des Auftrages
- Stornierung des Auftrages
- Erstellung einer Auftragsstatistik
- Automatische Auftragsbestätigung, manueller Druck

Anforderung für die Inbetriebnahme und den Einsatz: Der Ablauf ist so zu organisieren, dass der bisherige Ablauf der Kundenbedienung und der Auftragsbearbeitung kundenfreundlich durchgeführt werden kann. Die Bedienerführung muss verständlich und übersichtlich gestaltet werden. Ein schneller Informationsfluss sowie Folgeinformationen müssen präsent sein.

Anforderung an die Qualität: Die Komponente soll effizient, änderbar und wartbar sein, eine hohe Effizienz und Anpassungsfähigkeit bei Hardwareaustausch sowie Zuverlässigkeit besitzen. Die Qualitätsanforderung muss den bereits eingesetzten Komponenten entsprechen. Die Hardwareunabhängigkeit muss sich auf windowsbasierende Hardware beziehen.

Anforderung an die Projektabwicklung: Für die Entwicklung ist das .NET Framework zu nutzen. In den frühen Entwicklungsphasen soll ein Prototyp generiert werden, damit die Anwender rechtzeitig Korrekturen und Fehler melden können. Der Systementwurf soll der bewährten Architektur schon laufender Komponenten entsprechen. Die Programmiersprache muss eine .NET Framework-Sprache sein und als Datenablage die vorhandene relationale Datenbank nutzen.

6.4.2 Erstellung des Pflichtenheftes

Aus dem Lastenheft lässt sich nunmehr der Inhalt des Pflichtenheftes für die Komponente Auftragsbearbeitung ableiten. Technische Beschreibungen werden detailliert vorgenommen, zuvor werden jedoch die einleitenden, allgemein beschreibenden Merkmale erläutert:

Projektbeschreibung und -ziel: Die Auftragsbearbeitung ermöglicht einem kleinen oder mittleren Betrieb die Erfassung und Bearbeitung von Aufträgen. Trotz ihrer Komplexität ist die Auftragsbearbeitung eine komfortable und einfach zu bedienende Komponente. Die Auftragsbearbeitung ist optimiert für kleine und mittlere Unternehmen, die als Schwerpunkt die Lohn- und Einzelfertigung aufweisen. In Verbindung mit der Fertigungskomponente wird eine Terminierung und Überwachung komfortabel möglich. Archiv-Funktionen, Excel-Export und Statistiken sind selbstverständliche Features der Komponente. Die Auftragsbearbeitung besitzt eine intuitive windowskonforme Benutzeroberfläche und kann somit einfach und auch ohne aufwendige Schulung bedient werden. Die wichtigsten Funktionen sind sofort zugänglich und übersichtlich strukturiert. Sämtliche relevanten Informationen stehen direkt zur Verfügung. Intelligente und flexible Such- und Filterfunktionen ermöglichen einen bequemen Umgang. Die abgestimmten Übergänge zu den anderen Modulen verhelfen zu einer leichten, intuitiven Navigation, die das Merken von Nummern oder Zeichen überflüssig macht.

Forderungen und Kriterien: Die Mindestforderungen für die Komponente setzen sich aus den Merkmalen und Funktionen des Altsystems und Ansprüchen, die aus Erfahrungen resultieren, zusammen. Bekannt ist, dass die meisten neuen Ansprüche nur in Verbindung mit den notwendigen Nachbarkomponenten erfüllt werden können. Bedingung für die Erfüllung dieser Forderungen ist der Einsatz der Arbeitspläne, der Fertigung, des Einkaufs und des Lagers. Nur so kann eine durchgängige Auftragsabwicklung garantiert werden.

Forderungskatalog an die Komponente:

- Strenge Datenüberprüfung bei Eingabe
- Windowskonforme Navigation in der Eingabemaske
- Verständliche Feldbeschreibungen

- Hervorhebung wichtiger Attribute (Liefertermin, Preis)
- Reichhaltige Information über auftragsbeeinflussende Daten (Lagerbestand, Parallelkommissionen, abhängige Artikel, analoge Artikel, Varianten, Preise, Historien)
- Übersichtlich strukturierte Auftragsdokumente
- Automatische Datenübergabe an abhängige Module (Zahlungs- und Lieferbedingungen, Kontenklasse usw.)
- Meldung bei widersprüchlichen Vorfällen
- Oberflächendesign muss sich an vorhandene Komponenten anlehnen
- Auftragsbestätigungen einzeln und bestellabhängig
- Simple Such- und Filtermöglichkeiten
- Automatische Trennung von erledigten und aktiven Aufträgen (Archivierungsfunktion mit Recovery)
- Bidirektionaler Datenabgleich zwischen Auftrag – Beschaffung, Auftrag – Fertigung
- Prioritätenvergabe von Aufträgen
- Benutzerabhängige Rechte im Modul (Lesen, Ändern, Erstellen, Löschen)

Zusätzliche Features:

- Exportmöglichkeit nach Excel
- Listengenerierung
- Kunden- und Artikeldatenpflege ohne Modulwechsel
- Restmengenangaben bei Teillieferungen
- Druckinformationen (wann und von wem gedruckt)
- Hierarchische Kommissionen (Unter- bzw. Kinderkommissionen)
- Verschiedene Auftragsstatistiken und Auswertungen
- Ortsungebundene Verwendung (z.B. via VPN, DFÜ)

Ausgeschlossene Kriterien:

- Änderung oder Löschen der Verknüpfung des Artikels zum Auftrag, wenn der Fertigungsauftrag aktiv ist, Artikeleigenschaften können modifiziert werden
- Korrekturen, welche die Struktur des Arbeitsplans betreffen, sind bei aktiven und abhängigen Fertigungsaufträgen nicht zulässig

Anwendungsbereiche:

Diese Komponente wird hauptsächlich im Büro verwendet, jedoch ist eine Bearbeitung auch unabhängig vom Firmenstandort möglich, beispielsweise mit einem Notebook über eine Mobilfunk- oder Internetverbindung (DFÜ, VPN via UMTS, GPRS usw.).

Benutzeroberfläche:

- Windowskonform
- Den bereits installierten Komponenten entsprechend
- Allgemeine Regeln der UI-Gestaltung einhaltend

Komponentendesign und Datenzugriff: Die Ausarbeitung dieses Punktes beschreibt den technischen Kern des Konzeptes. Aus diesem Grund wird diesem Thema ein eigenständiger Abschnitt zugeordnet. Obwohl dieser Punkt explizit zum Pflichtenheft gehört, dient eine Trennung der Übersichtlichkeit. Er wird daher im kommenden Kapitel, 6.5 Design und Architektur am Beispiel einer Komponente, detailliert erläutert.

Schnittstellen: Die Einhaltung der Standards, Regeln und Richtlinien, wie sie im Programmierparadigma (Kapitel 5.4.3) fixiert sind, macht eine explizite Beschreibung der Schnittstellen überflüssig. Allein die strikte Einhaltung des Paradigma und die Nutzung des Frameworks garantieren eine problemlose Schnittstellenimplementierung.

Einsatzort: Die Komponente wird als Modul in der Hauptanwendung unter Windows XP und dem .NET Framework als Rahmenbedingung eingesetzt. Die Netzwerkvoraussetzungen sind TCP/IP als Protokoll mit entsprechender Datenübertragung.

6.5 Design und Architektur am Beispiel einer Komponente

Neben dem Projektmanagement bildet das Komponentendesign den nächsten Kernpunkt des Konzeptes. Genau wie die strikte Einhaltung des Dogmas hinsichtlich der Projektabwicklung muss das Design des Assembly der Vorlage entsprechen. Zwar sind dem Programmierer alle Freiheiten gegeben, programmiertechnisch bessere Ansätze zu finden, jedoch die Struktur des Assembly steht fest. Es muss mindestens aus den erforderlichen Datenbankobjekten, der Klasse, dem Interface und einem Userinterface (Web UI oder Windows UI) bestehen. In der folgenden Abbildung sind die Bestandteile und die Struktur der Auftragskomponente zu erkennen. Sie stellt das Projekt „IAuftrag“ in der Visual Studio® .NET™ Programmierumgebung dar.

Abbildung 6-4 Assemblyprojekt

Die Klasse Assemblyinfo zeigt allgemeine Informationen über das Assembly und wird von der Programmierumgebung selbst erzeugt, die Attributwerte können verändert werden. Zusätzliche Verweise zu Bibliotheken, die nicht aus dem Framework stammen, werden nicht benötigt und entsprechen auch nicht der Idee des Konzepts. Jede Komponente, die nach diesem Modell erstellt wird, beinhaltet nur die Frameworkklassen System, System.Data, System.XML, System.Drawing und System.Windows.Forms beim Windows UI (Userinterface) bzw. alternativ System.Web beim Web UI. In gewissen Assemblies ist selbstverständlich auch die Frameworkklasse System.Security enthalten oder diverse Klassen, die den Zugriff auf Systemressourcen zulassen. Nachfolgend ist ein Codeauszug der Assemblyinfo abgebildet.

```
<Assembly: AssemblyTitle("SmallBusinessEDM.IAuftrag")>
<Assembly: AssemblyDescription("Interface: IAuftrag")>
<Assembly: AssemblyCompany("x-dms")>
<Assembly: AssemblyProduct("SmallBusinessEDM")>
<Assembly: AssemblyCopyright("Dipl.-Ing. J. Holl")>
<Assembly: AssemblyTrademark("")>
<Assembly: CLSCompliant(True)>
```

Abbildung 6-5 Assemblyinfo

6.5.1 Layout der Datenbankobjekte

Die Entwürfe der erforderlichen Tabellen, Trigger und Prozeduren sind auf das Komponentendesign abgestimmt. Die primäre Auftragstabelle, die Archivierungstabelle für Aufträge und die Relationstabelle für hierarchische Aufträge sowie diverse Systemtabellen sind die Grundlage für die Funktionalität der Komponente. Der Update-Trigger der Auftragstabelle aktualisiert die Zusammenhänge der relevanten Tabellen bzw. stimmt diese ab, wie etwa die Tabelle Beschaffung oder Fertigungsauftrag. Beim Setzen des Auftragsstatus auf „*Beendet*“ erfolgt die selbständige Archivierung. In der Auftragstabelle aktualisiert sich dadurch der Objektstatus, der Wert wird auf 3 gesetzt, d.h. der Datensatz wird als gelöscht markiert. Der Aufbau der Archivierungstabelle ist identisch mit der primären Auftragstabelle. Die folgenden Anweisungen (Abbildung 6-6 und Abbildung 6-7) sind Inhalt des Update-Triggers der Auftragstabelle und vollziehen die Archivierung. Die Archivtabelle besitzt keine Trigger.

Bestimmung der Objektstati (allgemeingültig):

1 = Vorhanden bzw. Lesen des Datensatzes

2 = Aktualisieren des Datensatzes

3 = Löschen (zum Löschen gekennzeichneter Datensatz)

```
INSERT   INTO     ARCHIV_TB_AUFTRAG
SELECT   * FROM TB_AUFTRAG
WHERE    TB_AUFTRAG.ObjektID = @mObjektID
AND      TB_AUFTRAG.KlassenID = @mKlassenID
```

Abbildung 6-6 Archivierungsanweisung

```
UPDATE   TB_AUFTRAG
SET      TB_AUFTRAG.State = 3,
         TB_AUFTRAG.GeaendertVon = @mBenutzerID,
         TB_AUFTRAG.GeaendertAm = GETDATE()
WHERE    TB_AUFTRAG.ObjektID = @mObjektID
AND      TB_AUFTRAG.KlassenID = @mKlassenID
```

Abbildung 6-7 Objektstatus

Diese Anweisungen werden in einer Schleife (lokaler Datenbank-Cursor) ausgeführt, da hierarchische Aufträge, d.h. Aufträge mit untergeordneten Kommissionen, automatisch mit abgemeldet werden. Eine separate Abmeldung der Unterkommissionen ist selbstverständlich möglich. Die Archivierung kann manuell oder bei festgelegten Ereignissen, z.B. Rechnungsausgabe, durchgeführt werden. Der vollständige Code des Triggers der Auftragstabelle ist im Anhang abgebildet (Abbildung 8-2 bis Abbildung 8-7). Bei der Ausgabe der Rechnung mit der vollständigen Auftragsmenge werden der Fertigungsauftrag und der Auftrag direkt abgemeldet bzw. archiviert, dieser Vorgang ist komplett reversibel. Bei jeder Reaktivierung eines Auftrages bzw. Fertigungsauftrages wird zuvor in der Archivierungstabelle überprüft, ob bereits eine Archivierung erfolgte, gegebenenfalls werden diese Einträge vor der neuen Archivierung in der Tabelle entfernt. Da in der primären Auftragstabelle der Datensatz lediglich zum Löschen markiert und der Auftragsstatus auf beendet gesetzt wurde, genügt die Änderung des Objekt- und Auftragsstatus, um den Auftrag wieder zu aktivieren. Parallel zur Archivierung werden Anweisungen ausgeführt, die das Auftragsvolumen aus historischen Gründen in entsprechende Tabellen einfügen (Varianten der Artikel). Bevor allgemein Änderungen der Auftragsdaten stattfinden, wird dieser Auftrag einer Statusprüfung unterzogen, z.B. Änderungen der Auftragsmengen, des Liefer- oder Korrekturtermins oder des Klienten (Kunden).

```
Alter Procedure AuftragSave_100
                                @ObjektID int output,
                                @KlassenID smallint,
                                @AuftragsNR int output,
                                .
                                .
                                .
                                @OhneLTAusgabe bit,
                                @BenutzerID int,
                                @dbAction tinyint,
                                @ret bit output

AS
DECLARE @mDatenOK bit

--logische Datenprüfung
--Code
--SELECT    @mDatenOK = 1 oder @mDatenOK = 0

IF @mDatenOK = 1
    BEGIN
        IF @dbAction = 0        --DB_READ
            BEGIN
                --Code
            END
        ELSE IF @dbAction = 1   --DB_INSERT
            BEGIN
                --Code
            END
        ELSE IF @dbAction = 2   --DB_UPDATE
            BEGIN
                --Code
            END
        ELSE IF @dbAction = 3   --DB_DELETE
            BEGIN
                --Code
            END
    END


SELECT  @ret = 1
RETURN
```

Abbildung 6-8 Prozedur zur Datenmanipulation

Das Hinzufügen, Löschen und die Manipulation der Auftragsdaten erfolgt ausschließlich über die vorgesehene Prozedur (vollständiger Prozedurcode im Anhang, Abbildung 8-8 bis Abbildung 8-12). Dieser Prozedur werden die entsprechenden Parameter durch die Komponentenklasse über die Kommunikationsebene übergeben. Die Abbildung 6-8 ist ein Code-Auszug aus der im Anhang abgebildeten vollständigen Anweisung zum Speichern der Auftragsdaten.

Der Zugriff auf die Prozeduren geschieht über Aliasnamen, genau so wie es auch im gesamten System organisiert wird. D.h., es existiert eine Tabelle mit den Attributen ObjektID, KlassenID, Klasse, Keyname (Aliasname) und Prozedurname. Beim Starten der Anwendung liest eine Funktion diese Auflistung ein und stellt sie während der gesamten Laufzeit zur Verfügung.

	ObjektID	KlassenID	Klasse	Keyname	Procedurname
▶	622	128	63	NetAuftrag_FA_InfoListe	Auftrag_FA_InfoListe_100
	625	128	63	NetAuftrag_GetChildren	Auftrag_GetChildren_100
	623	128	63	NetAuftrag_GetParent	Auftrag_GetParent_100
	502	128	63	NetAuftragBestaetigung	Auftrag_Bestaetigung_100
	503	128	63	NetAuftragBestaetigungListe	Auftrag_BestaetigungListe_100
	562	128	63	NetAuftragBilanz	AuftragBilanz_100
	458	128	63	NetAuftragCreateFA	Auftrag_CreateFA_100
	463	128	63	NetAuftragGesamtpreis	Auftrag_ReadGesamtpreis_100
	462	128	63	NetAuftragGlobalInfo	Auftrag_GlobalInfo_100
	558	128	63	NetAuftragHohePrioritaeten	Auftrag_HohePrioritaeten_100
	559	128	63	NetAuftragHohePrioritaetenListe	Auftrag_HohePrioritaetenListe_100
	461	128	63	NetAuftragObjektInfo	Auftrag_ObjektInfo_100
	604	128	63	NetAuftragPositionspreise	Auftrag_ReadPositionspreise_100
	460	128	63	NetAuftragRead	AuftragRead_100
	459	128	63	NetAuftragSave	AuftragSave_100
	575	128	63	NetAuftragStatistik	Auftrag_Statistik_100
	593	128	63	NetAuftragStatistikInnerListe	Auftrag_StatistikInnerListe_100
	576	128	63	NetAuftragStatistikListe	Auftrag_StatistikListe_100

Abbildung 6-9 System-Prozedurentabelle

Der Ausschnitt aus der Prozedurentabelle (Abbildung 6-9) gibt die Alias- und Prozedurnamen der Auftragskomponente wieder. Jetzt wird deutlich, wie neue oder geänderte Prozeduren der Anwendung ohne Versionsänderung und Kompilierung bekannt gegeben werden; eine einfache, aber effektive und wirkungsvolle Lösung.

Der Aufbau der Auftragstabelle, d.h. ihre Attribute mit Name, Typ, Größe, Standardwert und Beschreibung, ist auszugsweise in der Abbildung 6-10 dargestellt. Die komplette Auftragstabelle findet sich im Anhang in Abbildung 8-13. Im Anschluss erfolgt die Darstellung der Relationstabelle (Abbildung 6-11), die für Verknüpfungen der Kommissionen untereinander (Parent/Child) zuständig ist.

Attributname	**Typ**	**Größe**	**Standard**	**Beschreibung**
ObjektID	int	4		Primärschlüssel
KlassenID	smallint	2	63	KlassenID aus der Superklassentabelle
ArchivCL	smallint	2	64	KlassenID der Archivtabelle
KommissionsNR	int	4		Laufende Auftragsnummer, wird automatisch von der Datenbank vergeben
Menge	int	4		Auftragsmenge
Einheit	int	4		1:n Beziehung zur Tabelle der Einheiten, Zugriff über den entsprechenden Schlüssel (ObjektID)

ClientID	int	4		1:n Beziehung zur Tabellen der Klienten, Zugriff über die ObjektID
ClientCL	smallint	2		1:n Beziehung zu Tabellen der Klienten, dient zur Unterscheidung des Klienttyps
ChildID	int	4		1:n Beziehung zu Tabellen des Objekts, welches beauftragt wird
ChildCL	smallint	2		1:n Beziehung zu Tabellen der Objekte, dient zur Unterscheidung des Objekttyps (Artikel, Zukauf, DIN oder Arbeitsplan)
BestellNr	varchar	50		Bestellnummer des Klient
Bestelldatum	datetime	8		Datum des Auftragseingangs
IhrZeichen	varchar	50		Kontaktname der Bestellung
Positionspreis	money	8		Preis der Auftragsposition
Zuschlagpreis	money	8		Preiszuschlag pro Stück
PreisPlusPlus	money	8		Preiszuschlag pro Auftrag
Gesamtpreis	money			Auftragsvolumen
Liefertermin	datetime	8		Wunschtermin der Lieferung

Abbildung 6-10 Auftragstabelle (Auszug)

Attributname	**Typ**	**Größe**	**Standard**	**Beschreibung**
ObjektID	int	4		Primärschlüssel
KlassenID	smallint	2	22	KlassenID aus der Superklassentabelle
ParentID	int	4		n:n Beziehung zur Tabelle der Aufträge, Zugriff über die ObjektID
ParentCL	smallint	2		n:n Beziehung zur Tabelle der Aufträge, dient zur Unterscheidung der Klassen
ChildID	int	4		n:n Beziehung zur Tabelle der Aufträge, Zugriff über die ObjektID

ChildCL	smallint	2		n:n Beziehung zur Tabelle der Aufträge, dient zur Unterscheidung der Klassen
BearbeitetAm	datetime	8		Bearbeitungsdatum
BearbeitetVon	int	4		1:n Beziehung zur Tabelle der Benutzer

Abbildung 6-11 Relationstabelle

Die Relationen der Aufträge untereinander werden in einer eigenständigen Tabelle abgelegt. Die Parent-Attribute verkörpern die übergeordnete Kommission (Auftrag) und die Child-Attribute stellen entsprechend die Unterkommissionen dar. Die Eingrenzung auf nur eine Strukturtiefe der Relationen ist für die Auftragsbearbeitung völlig ausreichend. Hierarchieabbildungen mit beliebig tiefen Strukturen sind vorwiegend in Baugruppenstücklisten notwendig. Die Prüfungen auf Logik und Konsistenz finden vor dem Einfügen der Daten in die Tabelle statt. Zur Speicherung der Relation existiert eine eigenständige Prozedur, sie beinhaltet alle Drag&Drop-Speicherungen für Relationen, da ein großer Teil der Verknüpfungen mit dieser Methode erstellt wird. Die nachstehende Abbildung zeigt lediglich den Teil der Prozedur, der für die Auftragsrelationen bestimmt ist.

```
Alter Procedure ObjektRelationSave_100

                                        @dragID int,
                                        @dragCL smallint,
                                        @dropID int,
                                        @dropCL smallint,
                                        @dropKey varchar(5),
                                        @BenutzerID int,
                                        @dbAction smallint,
                                        @ret bit output

AS
IF  ((@dragCL = 63 AND @dropCL = 63) OR @dragCL = 22) AND @dragID <> @dropID AND @dropKey = 'O12'
    BEGIN
        IF @dbAction = 1        --INSERT in REL_AUFTRAG_AUFTRAG
            BEGIN
                --Logik- und Konsistenzprüfung
                --Code
            END
        ELSE IF @DBAction = 3   -- DELETE in REL_AUFTRAG_AUFTRAG

            BEGIN
                --Code
            END
    END
RETURN
```

Abbildung 6-12 Prozedur zur Relationsspeicherung

Die Funktionen, die der Komponente zur Verfügung stehen, werden ebenfalls über Datenbankprozeduren definiert. Sinnvolle und wichtige Features sind z.B. Links zu abhängigen Tabellen, Statistiken und Eigenschaften. Welche Funktion für die Auftragsklasse aktiv

ist, zeigt die folgende Anweisung (Abbildung 6-13). Über die Klassen-ID werden die Funktionen mit einem Key bereitgestellt. Dieser Key dient zur Identifizierung der Funktion im Programmcode.

```
ELSE IF @KlassenID = 63 --Auftrag
    BEGIN
        SELECT  'Eigenschaften' as [001],
                'Links Fertigungsauftrag' as [037],
                'Links Arbeitsplan' as [005],
                'Links Artikel' as [002],
                'Links Kunden' as [040],
                'Links Lieferscheine' as [058],
                'Links Rechnungen' as [059],
                'Links Parents' as [016],
                'Links Children' as [012],
                'Status ändern' as [044],
                'FA generieren' as [035],
                'Transfer Auftrag' as [009],
                'Prioritäten' as [057],
                'Bilanz' as [039],
                'Statistik' as [042],
                'Send Workflow' as [022],
                'XML Report' as [018]

    END
ELSE IF @KlassenID = 94 --Fertigungsauftrag
```

Abbildung 6-13 Funktionen der Auftragsbearbeitung

Wie im Programmcode anschließend der Umgang mit den Komponentenfunktionen erfolgt, erklärt das Kapitel „6.5.8 Die Komponente im Verbund der Hauptanwendung" dieser Arbeit. Um den Rahmen dieser Arbeit nicht zu sprengen, können nur Auszüge aus dem Funktionsumfang dargestellt werden.

6.5.2 Layout der Klasse des Assembly

Zu jedem Assembly gehört eine Klasse, sie beinhaltet die Datenaufbereitung und Validierung für die Transaktionen und stellt Eigenschaften und Methoden der Komponente zur Verfügung. Die Informationen des Datasets, welches von der Kommunikationsebene für die Klasse auf Anforderung bereitgestellt wird und die Ergebnisse der Datenbankanfrage beinhaltet, werden hier weiter verarbeitet. Das Initialisieren und Kapseln der Daten in ein Objekt sowie das Speichern des geänderten Objekts geschehen mit dem Aufruf der Methoden. Die Methoden Init, Save und Delete sowie allgemeingültige Standardeigenschaften kommen in jeder Komponente vor. Eine Übersicht über Standardeigenschaften und –methoden sowie komponentenabhängige Eigenschaften und Methoden ist in der folgenden Tabelle (Abbildung 6-14)enthalten. Ferner können die Typen und, falls notwendig, die dazugehörigen Beschreibungen den Tabellen (Abbildung 8-13 bis Abbildung 8-16) im Anhang entnommen werden.

Methodenname	Typ	Beschreibung
InitClass	Prozedur (Public Sub)	Stellt die Verbindung zur Datenbank über die Kommunikationsebene her und füllt mit dem Rückgabeergebnis die Eigenschaften
Save	Prozedur (Public Sub)	Übergibt die geänderten Objektdaten der Kommunikationsebene zum Speichern
Delete	Prozedur (Public Sub)	Setzt den Flag zum Löschen und ruft die Methode Save auf
CheckData	Prozedur (Private Sub)	Prüft die Objektdaten und gibt sie bei Erfolg zum Speichern frei
OpenInterface	Funktion (Public Function)	Regelt den Zugriff auf das Benutzercontrol und informiert über den Erfolg
IICheckData	Funktion (Private Function)	Prüft, ob die Instanziierung durch OpenInterface erfolgen kann
ClearProperties	Prozedur (Private Sub)	Setzt Objektdaten in den initialisierten Zustand zurück

Abbildung 6-14 Übersicht der Standardmethoden

Mit dieser Aufstellung ist es jetzt möglich, die Assembly-Klasse zu konstruieren. Beginnend mit dem Deklarationsbereich über die Methoden zu den Eigenschaften entsteht die notwendige Klasse. Die folgende Abbildung zeigt den Ansatz zur Klassenerstellung. Natürlich können nur Auszüge aus der Klasse gezeigt werden, da sonst die Übersichtlichkeit verloren geht. Das Augenmerk soll auf dem strukturellen Aufbau der Klasse liegen.

```
'Implementierung
Implements IExtInterface.MyInterface
'################################################
'Deklarierung
Private cIAuftragsNR As Integer
'...
Const DB_INSERT As Integer = 1                    'gilt nur für Objekte in Klassen!!!
'...
Private WithEvents AuftragControl As ctrlAuftrag
'################################################
```

Abbildung 6-15 Deklarationsbereich

Mit der Implementierung des Interfaces und der anschließenden Deklaration der privaten Variablen, Konstanten sowie Objekte, beginnt der Aufbau der Klasse. Stellvertretend für die Standarddeklarationen steht die Konstante zur Unterscheidung des Objektstatus. Die Auftragsnummer ist eine klassische komponentenabhängige Variable. Zum Abschluss der Deklaration steht das Control, also die Benutzerform der Komponente.

```
'Beginn Standard-Methoden
Public Sub InitClass() Implements ClassLibrary6.IExtInterface.MyInterface.InitClass
    'Code
End Sub
'weitere Standard Methoden
'...
'Ende Standard-Methoden
'################################################
'Beginn Komponentenabhängige-Methoden
Public Function ReadPositionspreis() As System.Data.DataSet _
                    Implements ClassLibrary6.IExtInterface.MyInterface.ReadPositionspreis
    'Code
End Function
'################################################
```

Abbildung 6-16 Methodenbereich

Repräsentativ für die Standardmethoden steht die Init-Methode, sie holt über die Kommunikationsebene die Daten für das Objekt aus der Datenbank und weist ihm die Eigenschaften, z.B. die Auftragsnummer, der Klasse zu. Die Anzahl der komponentenabhängigen Methoden kann stark variieren, je nachdem, welche Ansprüche die Komponente aufweist. Im Fall der Auftragsbearbeitung, wie aus der entsprechenden Tabelle ersichtlich, sind lediglich vier Methoden implementiert. Der ausgeführte Programmcode in den Methoden beinhaltet keine besonderen Merkmale, auf die speziell eingegangen werden muss. Es werden lediglich allgemein bekannte Verfahren bzw. Routinen, wie Schleifen, Bedingungen, Umwandlungen usw. ausgeführt.

So werden in der Init-Methode der Datenbankprozedurname aus einer Auflistung gelesen und die notwendigen Parameter erzeugt, dann diese der Kommunikationsebene übergeben, die das Abfrageergebnis in einem Dataset zurückgibt. Nach erfolgreicher Ausführung der Anweisung werden die Resultate den entsprechenden Eigenschaften zugewiesen. Bei der Speicherung der geänderten Objektdaten werden die Eigenschaftswerte in eine Struktur geschrieben und wiederum der Kommunikationsebene übergeben. Diese konvertiert das Array, abhängig vom Datenbankprovider, zu einem datenbankkonformen Layout und übergibt es der Datenbank.

Der letzte Teil der Klasse beschreibt die Eigenschaften der Komponente. Die Eigenschaft "*Vorhanden*" steht standardmäßig in jeder Klasse, die „*Auftragsnummer*" hingegen nur in der Beispielkomponente. Die komponentenabhängigen Eigenschaften sind frei formulierbar, wie es von den Methoden her bekannt ist (vgl. Abbildung 6-17).

```
'Beginn Standard-Eigenschaften
Public Property IVorhanden() As Boolean _
                    Implements ClassLibrary6.IExtInterface.MyInterface.IVorhanden
    Get
    End Get
    Set(ByVal Value As Boolean)
    End Set
End Property
'weitere Standard Eigenschaften
'...
'Ende Standard-Eigenschaften
'################################################
'Beginn Komponentenabhängige Eigenschaften
Public Property IAuftragsNR() As Integer _
                    Implements ClassLibrary6.IExtInterface.MyInterface.IAuftragsNR
    Get
    End Get
    Set(ByVal Value As Integer)
    End Set
End Property
'weitere Komponentenabhängige Eigenschaften
'...
'Ende Komponentenabhängige Eigenschaften
'################################################
```

Abbildung 6-17 Eigenschaftenbereich

6.5.3 Layout des Interfaces des Assembly

Das von der Klasse eingebundene Interface ist einfach zu verstehen und bedarf keiner ausgiebigen Beschreibung. Es stellt die Eigenschaften und Methoden (Abbildung 6-18) der Klasse zur Verfügung, die über das Interface zugänglich gemacht werden sollen.

```
Public Class IExtInterface
    Public Interface MyInterface
        'Methoden Standard
        Function OpenInterface() As Boolean
        Sub InitClass(ByVal Connection As SqlClient.SqlConnection, _
                    ByVal BenutzerID As Integer, ByVal ObjektID As Integer)
        Sub Save()
        Sub Delete()
        '########################
        'Methoden komponentenabhängig
        Sub ReadObjektinfo()
        Sub ReadGlobalinfo()
        Sub ReadGesamtpreis()
        Function ReadPositionspreis() As DataSet
        Function GetClassRecht(ByVal IClass As Short) As Short
        Function GetMyListe() As Object
        '... weitere
        '########################
```

Abbildung 6-18 Interface-Methoden

Der erste Teil des Interfaces enthält die Methoden der Komponente, sie entsprechen der Auflistung in der Tabelle. Bei Bedarf können die komponentenabhängigen Methoden und

Eigenschaften des Interfaces erweitert werden. Eine Veränderung der Standardmerkmale zieht eine Änderung durch die gesamten Assemblies nach sich.

```
'Eigenschaften komponentenabhängig
Property IAuftragsNR() As Integer
Property IChildID() As Integer
Property IChildCL() As Short
Property IClientID() As Integer
Property IClientCL() As Short
Property IMenge() As Integer
Property IBestellNR() As String
Property ILiefertermin() As Date
'... weitere
Property IErstelltAm() As Date
Property IErstelltVon() As String
Property IGeaendertAm() As Date
Property IGeaendertVon() As String
Property IVersion() As String
Property IStatus() As String
'#######################
```

Abbildung 6-19 Komponentenabhängige Eigenschaften

Die Eigenschaften, die von der jeweiligen Komponente abhängig sind, werden in der Regel vom Usercontrol (Windowsforms, Webforms) benutzt. Sie füllen letztendlich Textfelder, Listboxen, Auswahlboxen, Label usw. und damit die Form mit Daten. Die Standardeigenschaften hingegen, wie in der Abbildung 6-20 zu sehen, beinhalten wichtige Informationen, die für jede Komponente relevant sind. Benutzer-ID, Datenbankprovider, Rechteinformationen usw. sind allgemeingültige Eigenschaften.

```
        'Eigenschaften Standard
        ReadOnly Property IDataOK() As Boolean
        Property IVorhanden() As Boolean
        Property IObjektID() As Integer
        Property IKlassenID() As Short
        Property IRechte() As Object
        Property IRechteKey() As Short
        Property IInterfaceDic() As Object
        Property IBenutzerID() As Integer
        Property IConnection() As SqlClient.SqlConnection
        '... weitere
    End Interface
End Class
```

Abbildung 6-20 Standardeigenschaften

6.5.4 Layout des Userinterfaces des Assembly

Wie die Gestaltung des Benutzerinterfaces aussieht, ist dem Entwickler zu überlassen. Wichtig ist nur die Einhaltung der Regeln zur Gestaltung des Userinterfaces, wie sie in Seminaren oder Unterlagen für Entwickler [24] erläutert werden. Wie Controls mit Daten gefüllt und diese wieder ausgelesen werden, hängt vom Programmierer ab. Das Benut-

zercontrol greift nur auf Objektdaten zurück und hat keine Kenntnis über die Datenbank. Der Aufruf des Usercontrols erfolgt über die Assembly-Klasse mit der Methode „Init-Control", in ihr wird das Control instanziiert. Bei der Instanziierung übergibt sich die Klasse selbst als Parameter, somit wird eine falsche Klassenzuweisung des Controls ausgeschlossen.

6.5.5 Zusätzliche Module und Funktionen der Komponente

Bestimmte Komponenten verlangen gesonderte interne Behandlungen, z.B. der Zugriff auf Hardwareschnittstellen, wie COM, USB oder LPT. Der Code, der für diese Fälle benötigt wird, ist separat in Module zu verfassen. Die Auftragsbearbeitung besitzt derzeit keinen in Module ausgelagerten Code. Bei dem Assembly der Dokumentenverwaltung hingegen wäre das Ansprechen des Flachbettscanners als Beispiel zu nennen, dessen Code in einem Modul zu finden ist.

6.5.6 Instanziierung der Komponente im Main

Der Aufruf des Assembly findet in einer so genannten Factory-Klasse statt. Der Vorteil ist durch die zentrale Instanziierung aller Komponenten im Host gegeben. Jede Instanz, die die Anwendung benötigt, wird hier erstellt und bei Bedarf übergeben. In der Abbildung 6-21, Abbildung 6-22 und Abbildung 6-23 sind Ausschnitte aus der Factory-Klasse, die für die Instanziierung der Komponente verantwortlich sind, markiert und dargestellt.

```
    Private cIWerkzeuge As IWerkzeug.IExtInterface.MyInterface
    Private cIWerkzeugausgabe As IWerkzeugausgabe.IExtInterface.MyInterface
    Private cIAuftrag As IAuftrag.IExtInterface.MyInterface
    Private cIEinkauf As IEinkauf.IExtInterface.MyInterface
    Private cIBuchungen As IBuchungen.IExtInterface.MyInterface
    Private cIRechnungen As IRechnungen.IExtInterface.MyInterface
    Private cIInventar As IInventar.IExtInterface.MyInterface
    Private cIFirmeneinsatz As IFirmeneinsatz.IExtInterface.MyInterface
    Private cIAngebote As IAngebot.IExtInterface.MyInterface
    Private cILieferungen As ILieferungen.IExtInterface.MyInterface
    Private cIArbeitsgangtext As IArbeitsgang.IExtInterface.MyInterface
    Private cIKostenstellen As IKostenstellen.IExtInterface.MyInterface
    Private cIMaschinen As IMaschinen.IExtInterface.MyInterface
    Private cIKnowledge As IKnowledge.IExtInterface.MyInterface
    Private cIImports As IImports.IExtInterface.MyInterface

    '################################################################################
```

Abbildung 6-21 Deklarationen in der Factory-Klasse

```
                cIWerkzeugausgabe = New IWerkzeugausgabe.cWerkzeugausgabe()
                mInterface = BuildInterface(cIWerkzeugausgabe)
            Case mMyTables.TB_APL
                cIWithListe = True
                cIArbeitsplan = New IAPL.cAPL()
                mInterface = BuildInterface(cIArbeitsplan)
            Case mMyTables.TB_AUFTRAG
                cIWithListe = True
                cIAuftrag = New IAuftrag.cAuftrag()
                mInterface = BuildInterface(cIAuftrag)
            Case mMyTables.TB_FERTIGUNGSAUFTRAG
                cIFA = New IFertigungsauftrag.cFertigungsauftrag()
                mInterface = BuildInterface(cIFA)
            Case mMyTables.TB_BESTELLUNG
                cIWithListe = True
                cIEinkauf = New IEinkauf.cEinkauf()
```

Abbildung 6-22 Instanziierung in der Factory-Klasse

```
        cILocked = CheckLockedObject(cIObjektID, cIKlassenID, True)

        If IAsForm Then

            If TypeOf cICaller Is frmBasis Then cICaller.SetLinkLabelSpeichernFalse()

            cIMyForm = New frmEdit()
            cIMyForm.Editklasse = cIKlassenname
            cIMyForm.ActiveID = cIObjektID
            cIMyForm.ActiveCL = cIKlassenID
            cIMyForm.LockedObject = cILocked
            cIMyForm.Caller = cICaller
            cIMyForm.OpenForm(mInterface)

            Exit Function

        End If

        If Not mInterface Is Nothing Then
            InterfaceInititialize1 = mInterface
        End If

    Catch e As Exception
```

Abbildung 6-23 Komponentenaufruf in der Factory-Klasse

Der endgültige Aufruf stellt letztlich die Komponente in einer Form oder auf einem Panel (Steuerelement, das zur Komponentenvorschau dient, vgl. Abbildung 6-28) dar.

6.5.7 Aufbau der Kommunikationsebene

Der Aufbau der Kommunikationsebene entspricht einer herkömmlichen Klassenstruktur mit Methoden und Eigenschaften. Im Folgenden wird der programmiertechnische Inhalt konzentriert dargestellt. Beginnend mit der Deklaration der Variablen (Abbildung 6-24) über die Auflistung der Methoden (Abbildung 6-25) und Eigenschaften (Abbildung 6-26) wird im Anschluss ergänzend die Abstrahierung (Abbildung 6-27) der allgemeinen Basisklasse für Dictionaries vorgestellt.

```
Public Class cMyProcedur
    '############################
    'Deklarierung
    Private m_retval As Integer
    Private m_retparms As SqlClient.SqlParameterCollection
    Private pName() As String
    Private pType() As SqlDbType
    Private pDirection() As ParameterDirection
    Private pSize() As Integer
    Private pValue() As Object
    Private cMyDictionary As Object
    Private cTableName As String = "Data"
    Private cTimeInfo As String = ""
    Private cMyDataSet As DataSet = Nothing
    Private cMyDataAdapter As SqlClient.SqlDataAdapter = Nothing
    Private cMsgBoxOptions As MessageBoxOptions = 0
```

Abbildung 6-24 Deklarationsabschnitt der Kommunikationsebene

```
'############################
'Methoden/Funktionen
Public Function getSQLDatabase(ByRef pConnection As SqlClient.SqlConnection, _
                               ByRef pCommandText As String, _
                               Optional ByVal pCount As Integer = 0, _
                               Optional ByRef pWithParameter As Boolean = False, _
                               Optional ByRef pReturnParameter As Boolean = False) As DataSet...

Public Function getReturnParameter() As SqlClient.SqlParameterCollection
    getReturnParameter = m_retparms
End Function
```

Abbildung 6-25 Methoden der Kommunikationsebene

```
    '############################
    'Eigenschaften
    Public WriteOnly Property MyTableName() As String...
    Public ReadOnly Property MyTimeInfo() As String...
    Public WriteOnly Property MyDataSet() As DataSet...
    Public WriteOnly Property ParName() As Object...
    Public WriteOnly Property ParType() As Object...
    Public WriteOnly Property ParDirection() As Object...
    Public WriteOnly Property ParSize() As Object...
    Public WriteOnly Property ParValue() As Object...
    Public WriteOnly Property MsgBoxOptions() As Integer...

End Class
```

Abbildung 6-26 Eigenschaften der Kommunikationsebene

```
' Die Dictionary-Klasse, die auf der abstrakten Basisklasse für
' getypte Dictionaries basiert
Public Class Dictionary
    Inherits System.Collections.DictionaryBase
    ' Die Add-Methode wird neu implementiert, damit ein
    ' Anfügen möglich wird
    Public Sub Add(ByVal Key As Object, ByVal Value As Object)...
    ' Die Remove-Methode sollte implementiert werden,
    ' damit ein Löschen möglich wird
    Public Sub Remove(ByVal Key As Object)...
    ' Die Item-Methode muss implementiert werden, damit der
    ' Zugriff auf die Objekte möglich wird
    Public Function Item(ByVal Key As Object) As Object...
    ' Die Contains-Methode sollte neu implementiert werden,
    ' damit überprüft werden kann, ob ein Objekt mit einem
    ' bestimmten Schlüssel existiert
    Public Function Contains(ByVal Key As Object) As Boolean...
    ' Die Values-Eigenschaft sollte implementiert werden, damit ein
    ' Durchgehen mit For Each möglich wird
    Public ReadOnly Property Values() As ICollection...
End Class
```

Abbildung 6-27 Abstrahierungsklasse

Dieser gesamte Code befindet sich in einer Datei, stellt zwei Klassen dar und bildet zusammen mit der Konfiguration ein Assembly. Zusätzliche Verweise sind, wie bei den anderen Assemblies, nicht notwendig. Alle Datenbankprovider befinden sich innerhalb der standardisierten System.Data-Klasse. Wie die Daten an die Kommunikationsebene übergeben werden, wird im Anhang, Abbildung 8-1 Prozeduraufbau für Datenübergabe, an Hand des Prozedur-Codes demonstriert.

6.5.8 Die Komponente im Verbund der Hauptanwendung

Die Funktionsweise der Komponente lässt sich am besten im Verbund mit dem Main erklären. Der Host der Komponente, das Main, ist modular aufgebaut und bietet für die einzelnen Komponenten (Module) die optimale Umgebung. Bevor auf die Funktionen und Features der mit dem Konzept erstellten Auftragsbearbeitungskomponente eingegangen wird, müssen das Prinzip der Hauptanwendung und ihre Struktur, welche die Komponente instanziiert, erläutert werden. Beim Starten des Programms wird zuerst ein Login verlangt, d.h. der Benutzername und das entsprechende Passwort sind einzugeben. An Hand der Daten stellt die Datenbank die Benutzerrechte für die Anwendung zusammen.

Jedes Modul und jede Klasse der Anwendung besitzt einen eigenen Rechteschlüssel. So ist gewährleistet, dass an beliebiger Stelle in der Anwendung die angeforderten Informationen der einzelnen Klassen und Module auf Berechtigung geprüft werden können. Die Strukturierung des Rechteschlüssels vereint das Lesen, Ändern, Hinzufügen und Entfernen eines Objektes. Pro Modul und Klasse können so differenziert Rechte zugeordnet werden. In Modulen/Komponenten, in denen eine Klassifizierung stattfindet, wie Artikel

(Fertigungsart, Herstellungsart, Artikelgruppe usw.) und DIN-Teile (Lager, Verbindungselemente, Wellen usw.), existieren zusätzliche Rechte die die Klassifizierung des jeweiligen Objektes, gestatten oder verbieten. Erweiterte Rechte existieren ebenfalls im Mitarbeitermodul, da hier die Personaldaten integriert sind. Die Identifizierung eines Objektes in der Anwendung erfolgt prinzipiell über die Objekt-ID und die Klassen-ID. Durch diese eindeutige Identifizierung über die gesamte Datenbank hinweg, und nicht nur über die Tabelle, in der sich der Datensatz befindet, werden die zugeordneten Rechte des jeweiligen Objektes ermittelt. Es folgt eine kurze Erläuterung des Basisfensters (siehe Abbildung 6-28) der Anwendung, welches in verschiedene Bereiche gegliedert ist.

Anmerkung: Der Unterschied zwischen Modul und Klasse besteht darin, dass ein Modul mit einer Komponente gleichzusetzen ist und eine Klasse keine Komponente repräsentiert, wie z.B. Postleitzahlen oder Rohmaterialklassen. Klassen hingegen werden von Komponenten benötigt und sind meistens die Ergebnisse aus der Datenbanknormalisierung.

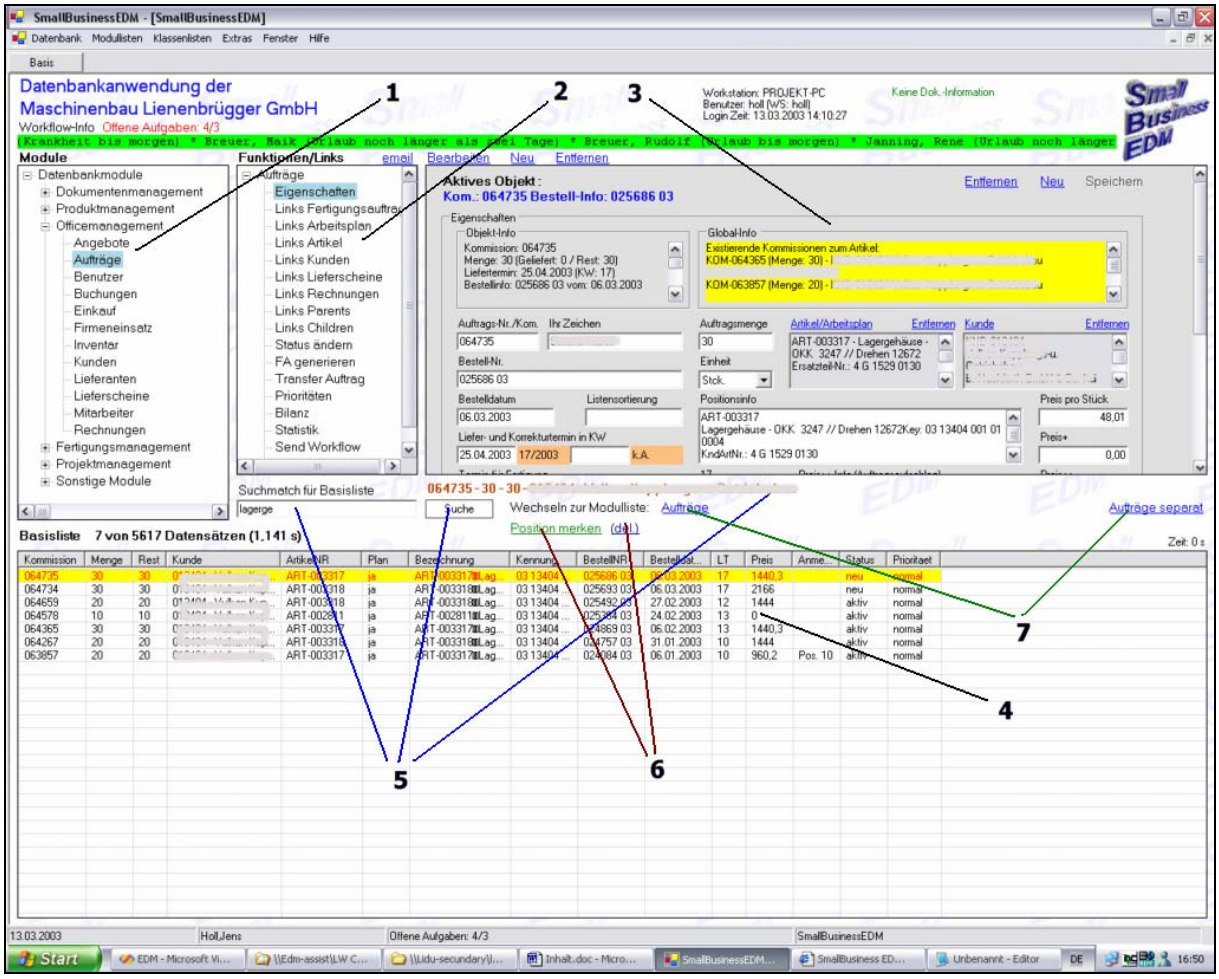

Abbildung 6-28 Basisfenster des Main

Nachfolgend die Erklärung der Bereiche für die Abbildung 6-28:

- Strukturansicht der Systemmodule, gruppiert nach Modultyp (Bereich 1).
- Eigenschaften und Funktionen des gewählten Moduls (Bereich 2).
- Ergebnis der Funktionswahl (Bereich 3), d.h. wird ein Objekt in der Basisliste aktiviert, stehen die Funktionen und Features durch Selektion zur Verfügung. In der Abbildung wird demonstrativ die Eigenschaft des Objektes gewählt. Ist das entsprechende Control, hier die Bearbeitungsform, zu groß bzw. in der Control-Vorschau ungenügend sichtbar, kann es zur Bearbeitung separat und variabel dargestellt werden.
- Basisliste, Listenansicht der Objekte eines Moduls (Bereich 4).
- Suchmatcheingabe für die Basisliste (Bereich 5), schränkt die Ergebnismenge ein und zeigt das aktivierte Objekt des Controls an. Verknüpfungen von Ausdrücken via AND und OR bei der Suche sind möglich.
- Speicherung der Position für schnelle Wiederaufnahme bei Unterbrechungen (Bereich 6).
- Link zu Modulliste (gebunden bzw. ungebunden), d.h. zu jedem Modul existiert eine separate Suchliste mit detaillierter Suchmöglichkeit. Sie dienen der Drag&Drop-Funktionalität, der erweiterten Detailsuche, dem Listendruck und modulabhängigen Listenfunktionen (Bereich 7).

Weiterhin können wichtige Informationen beim ersten Blick auf die Anwendung entnommen werden, beispielsweise Workflowmeldungen, Mitarbeiterticker, Umgebungsvariablen, Objektinfos oder Abfragezeiten. Wie in jeder Windowsanwendung existiert eine Menuleiste, mit der direkt zu den Menupunkten Datenbanken, Modullisten, Klassenlisten, Extras, Fenster oder Hilfe gewechselt werden kann.

Abbildung 6-29 Menu der Anwendung

Beim genauen Betrachten der Menuleiste und dem Lesen der Einträge der Untermenus wird noch mal der Unterschied zwischen Klassen und Modulen deutlich. Im Untermenu „Spezielle Listen" im Menu „Extras" werden gesonderte Abfragen gruppiert, die eine schnelle Ausführung ermöglichen. Ferner sind anwendungsspezifische Funktionen im Menu zu finden.

In der Abbildung 6-30 wird deutlich, dass die einzelnen Module zu Gruppen zusammengefasst werden. Exemplarisch sind das Baugruppen- und Fertigungsmanagement expandiert.

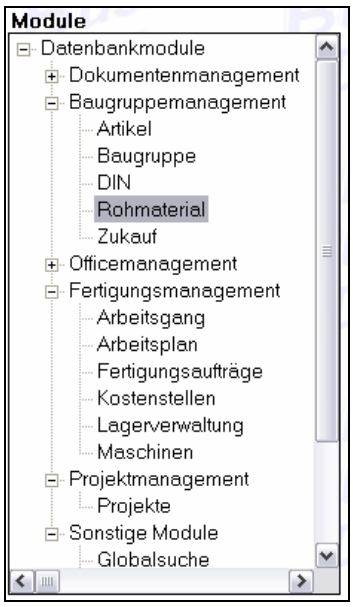

Abbildung 6-30 Module

Mit dieser Methode ist eine visuelle Unterstützung für den Benutzer gegeben, um einfacher im gesamten System zu navigieren. Wie bereits im Konzept erwähnt, ist eine Gruppierung auch nach anderen Kriterien möglich. Bei der vorgestellten Gruppierung sind unter „*Sonstige Module*" Features implementiert, die auf Fremddatenbanken zugreifen und andere spezielle Elemente, wie eine Wissensdatenbank, repräsentieren.

Die Funktionen- und die Links-Auswahl der selektierten Auftragskomponente (Auftragsmodul) in der Abbildung 6-31 geben einen Eindruck vom Umfang der Features dieser Komponente. Nicht nur eine Reihe von Links (Verknüpfungen) zu relevanten Informationen, auch verschiedene Auswertungsvarianten (Prioritäten, Bilanz, Statistik) bieten sich an. Selbstverständlich sind auch die Workflow-Funktionalitäten, die heute allgemein zum Standard zählen, für jedes Datenbankobjekt verfügbar. Das erste Feature in der Auflistung bietet die Bearbeitungsform des selektierten Auftrages in einer Control-Vorschau (vgl. Abbildung 6-28). Bei unzureichender Darstellung (z.B. durch zu geringe Auflösung) kann das Control separat instanziiert werden.

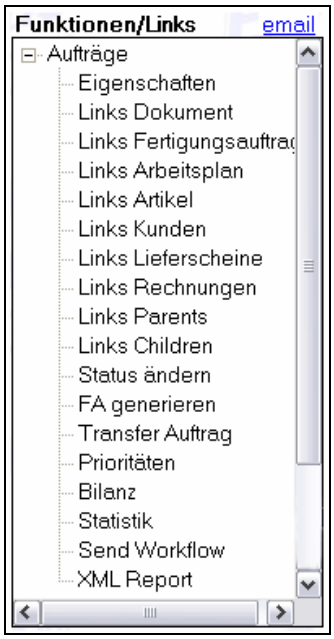

Abbildung 6-31 Funktionen und Links

Die folgenden Abbildungen zeigen, wie einzelne Features, welche die Auftragsbearbeitung betreffen, genutzt werden können. Dem Umfang und der Vielfalt der Funktionen und Links sind technisch keine Grenzen gesetzt, es werden jedoch demonstrativ nur einige dargelegt. Diese Auswahl ist kundenspezifisch definierbar.

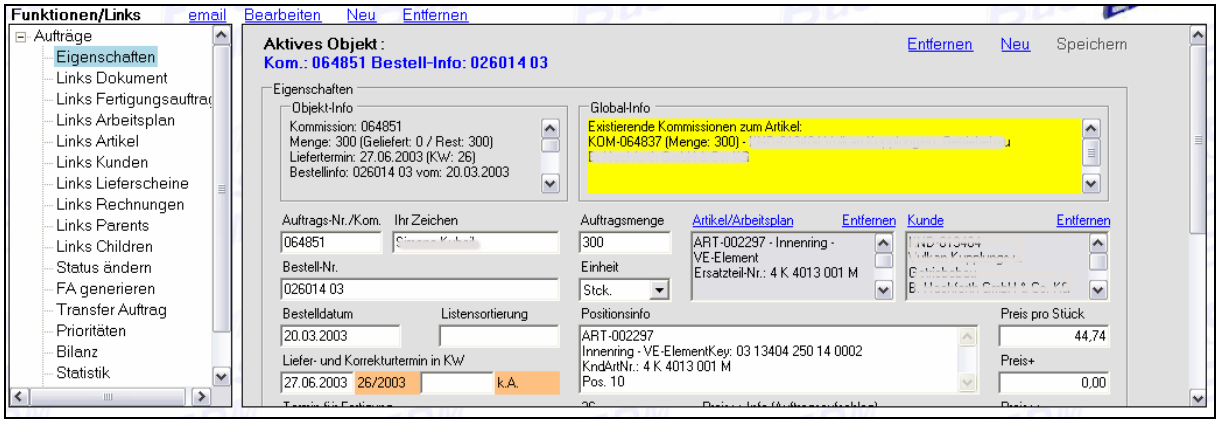

Abbildung 6-32 Auftrag Eigenschaften

Hinter dem Feature Eigenschaften verbirgt sich die Eingabemaske für die Auftragseigenschaften. Wie in der Abbildung 6-32 ersichtlich, ist eine Vorschau aktiviert. Ist die Komplexität der Eingabeform zu hoch, kann sie separat instanziiert werden. Die Eingabe erfolgt dann über eine eigenständige und in der Größe angepasste Darstellung. Entweder können die zu ändernden Eigenschaften bereits in der Vorschau modifiziert und gespeichert werden oder in der separat instanziierten Eingabemaske. Die Steuerelemente oben links über der Control-Vorschau erstellen die neuen Instanzen des Objektes oder löschen

es direkt. Das Steuerelement „email" über der Funktionsauflistung gestattet, auf Grundlage einer irgendwo in der Anwendung markierten E-Mailadresse, die sofortige Erstellung der Mail (Instanziierung des Email-Clients).

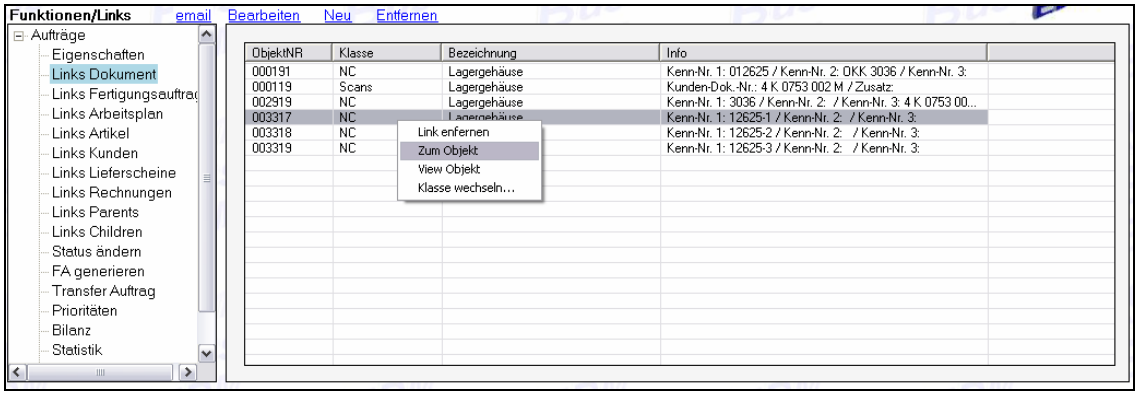

Abbildung 6-33 Auftrag Links Dokumente

Nach der Selektion eines Auftrages in der Basisliste (Bereich 4 aus Abbildung 6-28) kann die Funktion Links Dokument aktiviert werden. Das Resultat ist eine Liste mit den für den Auftrag relevanten Dokumenten. In der Abbildung 6-33 ist zum Beispiel erkennbar, dass das Objekt direkt betrachtet oder sogar visualisiert werden kann. D.h., das Scan-Dokument wird im Viewer betrachtet oder die NC-Programme im Editor geöffnet. Die jeweilige Reaktion hängt von den allgemeinen Windows-Einstellungen ab. Der Menupunkt "*Klasse wechseln*..." bedeutet einen direkten Übergang zur entsprechenden Klasse bezugnehmend auf den vorselektierten Eintrag. Mit dieser Methode lassen sich geschlossene Navigationskreise durch alle relevanten Module bis zurück zum Ausgangspunkt erzeugen, d.h., die Suche nach Informationen zu einem Vorgang gestaltet sich ohne Suchbegriffeingabe, sondern nur durch einen Mausklick. In der Abbildung 6-34 wird dieses Feature nochmals verdeutlicht.

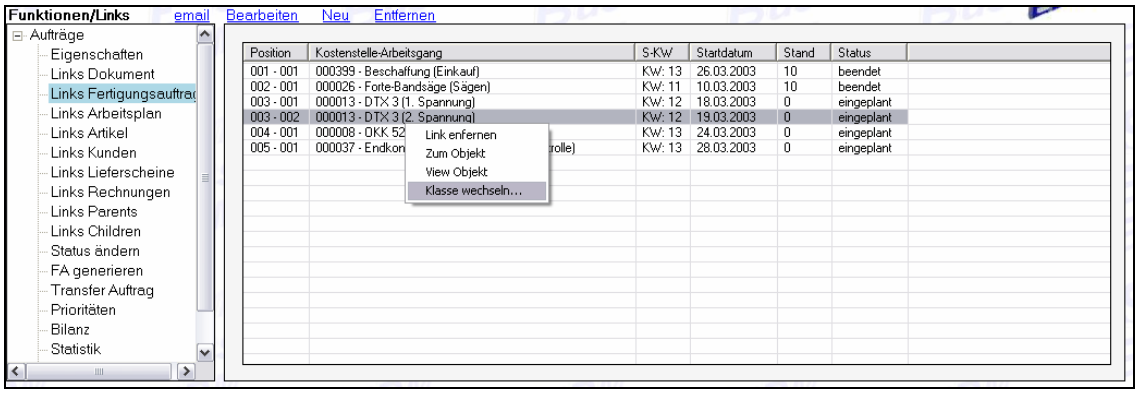

Abbildung 6-34 Relation Links Fertigung

An Hand des Kundenauftrages geschieht der unmittelbare Wechsel zum Fertigungsauftrag oder die augenblickliche Status-Information des Arbeitsganges kann abgelesen werden (vgl. Abbildung 6-34). Selbstverständlich sind ebenso die anderen objektbezogenen Funktionen verfügbar. Die Funktion „*Link entfernen*" ist natürlich nur sinnvoll bei Relationen, die keine intelligenten, bidirektionalen Eigenschaften aufweisen, da diese vom System selbst aufgelöst werden.

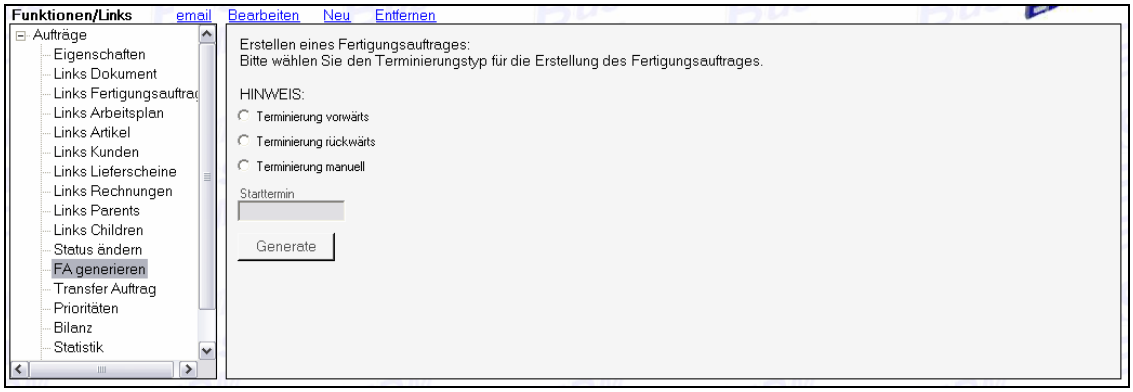

Abbildung 6-35 FA generieren

Die Funktion „FA generieren" in der Abbildung 6-35 erstellt einen terminierten Fertigungsauftrag für die Werkstatt/Produktion. Es werden automatisch Daten für die Beschaffung sowie eine Grobplanung für die Fertigung erzeugt und diverse Informationen (Stati, Flags) in den auftragsabhängigen Komponenten abgelegt.

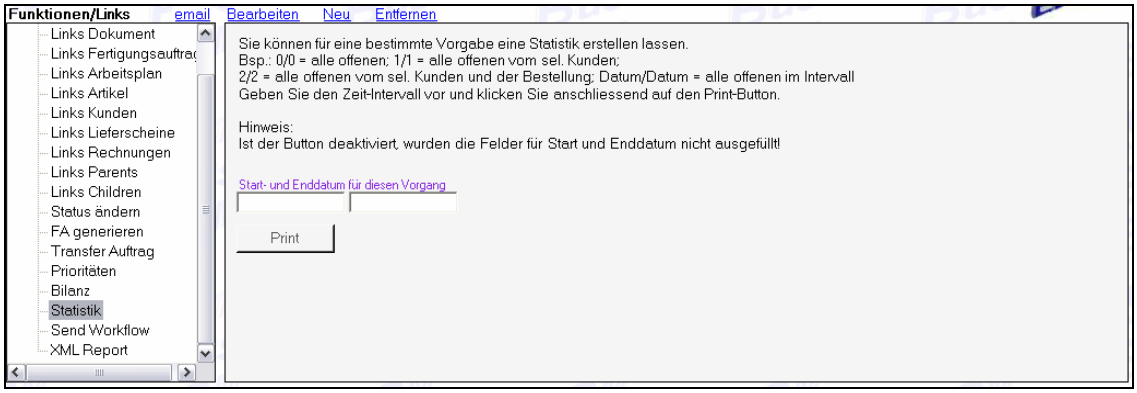

Abbildung 6-36 Auftragsstatistik

Mit der Statistikfunktion, wie in der Abbildung 6-36 ersichtlich, können Listen nach unterschiedlichen Kriterien erstellt werden, die Ausgaben basieren auf XML und werden im Browser dargestellt.

Die Funktionen und Links der Komponenten sind frei definierbar und erweiterbar. Die Systematik im Programmcode der Hostanwendung lässt Erweiterungen ohne Modifikatio-

nen des Quellcodes im Main zu. Somit ist der Ausbau des Funktionsumfanges ohne Einschränkung möglich.

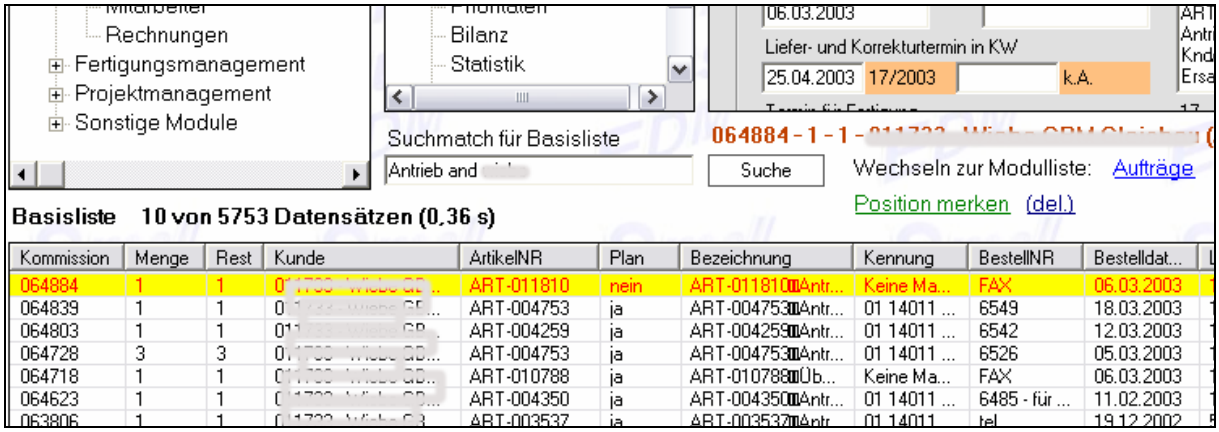

Abbildung 6-37 Allgemeine Suchfunktion

Die direkte und einfachste Suchmöglichkeit bietet sich im Basisfenster der Anwendung (siehe Abbildung 6-37). Die Suche nach dem eingegebenen Begriff, auch mit AND oder OR verknüpft, kann schnell und unkompliziert durchgeführt werden. Die Ergebnisliste beinhaltet nur noch die Einträge, die den Kriterien entsprechen. Zusätzliche nützliche Features, wie z.B. “Position merken”, dienen zum schnellen Auffinden der gemerkten Position bei den täglichen kleinen Arbeitsunterbrechungen.

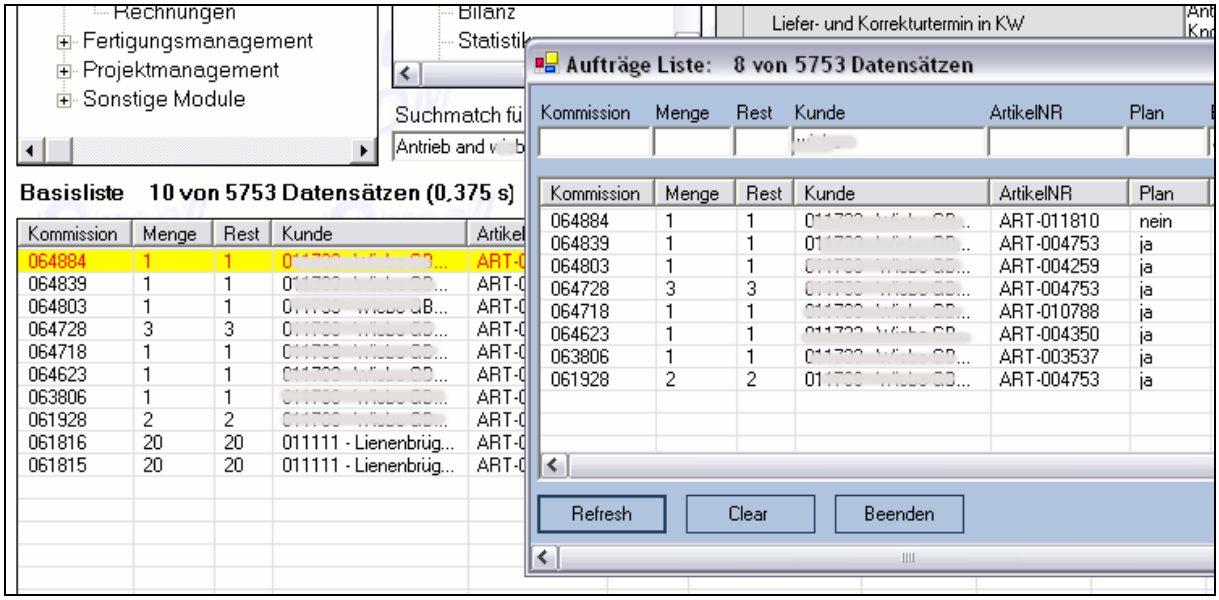

Abbildung 6-38 Erweiterte Suchfunktion

Für eine erweiterte Suchmöglichkeit stehen eigenständige Modul- und Klassenlisten zur Verfügung. Hier können in jedem zur Verfügung stehenden Feld Suchkriterien eingege-

ben werden, wie es in der oberen Abbildung „*Erweiterte Suchfunktion*“ erkennbar ist (vgl. Abbildung 6-38).

6.6 Diskussion der Faktoren zur Beurteilung der Umsetzung

Zur Beurteilung der Umsetzung sind nun nachfolgend die einzelnen Faktoren, wie Schulung, Motivation, Controlling und Bewertung der Entwicklung der Auftragskomponente, aufzuzeigen.

6.6.1 Schulung und Motivation

Zur Einführung des Prototyps haben die Anwender und die Entwickler den Umgang mit der Komponente diskutiert und die notwendige Schulung definiert. Etappenweise entstanden nach der Einführung Fragen und Verbesserungs- bzw. Optimierungsvorschläge zur Komponente. Die Gespräche, die sich daraus entwickelten hatten, besaßen gleichzeitig einen schulischen Charakter. Durch diese intensive Einbeziehung der Anwender und die ständige Unterrichtung der Geschäftsführung entstand ein hochmotiviertes Team, das keine Zweifel über die Richtigkeit der Vorgehensweise aufkommen ließ.

6.6.2 Controlling

Das zugrunde liegende Projektmanagement mit der engen Bindung zwischen den Teammitgliedern und dem ständigen Kontakt unter ihnen vereinfachte stark die Projektüberwachung bei der Entwicklung bezüglich der Auftragskomponente.

6.6.3 Darstellung und Bewertung des Projektablaufes

Nach der detaillierten Darstellung der Komponentenarchitektur an einem Beispiel gilt es jetzt den Projektablauf von der IST-Analyse bis hin zum Ersteinsatz zu diskutieren, um eine Gesamtbewertung durchführen zu können. Die IST-Analyse, die Bildung des Teams und die konkrete Aufgabenformulierung, fanden zu Beginn parallel statt. Das Team kristallisierte sich bei den ersten Sondierungsgesprächen und der IST-Analyse heraus. Aus den Ergebnissen der IST-Analyse und der Teamgespräche formulierte sich die Aufgabenstellung des Projektes der Auftragsbearbeitung. Die klare Abgrenzung des Aufgabenbereiches und eine intensive Prozessbeobachtung mit anschließender Einschätzung legten ein solides Fundament für das Pflichtenheft.

Nach dem Abschluss der Vorbereitungen begannen die programmiertechnischen Arbeiten. Zuerst wurden die Tabellen, dann die Initialisierungs- und Speicherprozedur sowie zusätzliche Anweisungen erstellt. Anschließend entstanden die Klasse, das Interface und

letztlich das Benutzercontrol. Nachdem der erste Prototyp fertig gestellt wurde, fingen die zeitaufwendigeren Etappen der Programmierung an. Die Standardmethoden sowie alle Eigenschaften der Klasse waren Routine und schnell zu implementieren, jedoch verlangten die komponentenabhängigen Methoden nachhaltigere Überlegungen und einen höheren Zeitaufwand. Die Einbeziehung der Teammitglieder auf der Benutzerebene wurde intensiviert, sie waren die Probanden der Komponente. Über mehrere Tage mussten sie mitunter doppelte Arbeit erledigen, komplette Aufträge sollten im Prototyp mitgepflegt werden, bis mindestens die Anforderungen der bisherigen Datenbankanwendung vollständig erfüllt wurden.

Nach ca. 14 Werktagen konnten alle neuen Aufträge mit der neuen Auftragsbearbeitung verwaltet werden, obwohl die Entwicklung der Komponente noch nicht abgeschlossen war. Jetzt begannen die Feinheiten und Abstimmungen. Da nur mit einer Mindestanzahl von Datensätzen sinnvoll zu arbeiten ist, wurden vorbereitete Abfragen und Listen angepasst und optimiert. Im nächsten Schritt wurden die Übergänge und Verknüpfungen zu den anderen Modulen abgerundet. Aus entwicklungstechnischer Sicht fanden bereits jetzt, vor Abschluss der Entwicklung der Auftragskomponente, erste projektvorbereitende Maßnahmen statt, wie SOLL-Analyse und Struktur- und Begriffsdefinitionen für die nächste Komponente (Fertigungsauftrag). Aber auch Korrekturen hinsichtlich der Arbeitsplanung mussten akzeptiert werden, diese waren zwar nicht tief greifend, aber eine Erwähnung ist zweckmäßig.

Nach ca. vier Wochen konnte das Projekt Auftragsbearbeitung, d.h. laut Pflichtenheft, vorab als beendet betrachtet werden. Einem kompletten Abschluss entspricht es aber sicher nicht, da mit jeder weiter hinzugefügten Komponente das Änderungspotential wächst. Darauf folgend konnte die Projektbewertung an Hand des Projekttestes von Prof. Dr.-Ing. S. Seibert durchgeführt werden.

Projektbewertung der Entwicklung der Auftragskomponente: Der Projekttest wurde mit den Teammitgliedern in unregelmäßigen Zeitabständen durchgeführt. Die Teammitglieder, vertreten durch Benutzer, Geschäftsführung und Entwickler, bewerteten den Test zuerst selbstständig und unabhängig voneinander. Das Ergebnis lässt, entsprechend der unterschiedlichen Stellung, Tendenzen aufzeigen, die eine intensivere Betreuung der Betroffenen fordern. Zu bemerken ist jedoch, dass das notwendige fachliche Wissen unterschiedlich ausgeprägt ist. Diesem Mangel vorzubeugen ist schwer, da Mitarbeiter unterschiedlicher Berufsgruppen beteiligt sind und auf sie nicht verzichtet werden kann. Aus der Berufspraxis ist bekannt, dass Schulungen nur in einem bestimmten Rahmen durchgeführt werden können, da der geschäftliche Ablauf nicht beeinträchtigt werden darf. Ausgewogene Verhältnisse zwischen Schulungsaufwand, Geschäftsprozessbeeinträchtigung und privaten Entbehrungen der Mitarbeiter sind zusätzliche, nicht zu verachtende

Punkte bei einer Projektdurchführung. Ferner beeinflusst die Methodik die Motivation der Beteiligten.

Die mit einem Sternchen (*) gekennzeichneten Stellen weichen von der Vorlage des Projekttestes von Prof.-Dr.-Ing. S. Seibert ab.

Auswertung des Projekttestes: Die Beurteilungen beziehen sich auf 3 Vollzeitkräfte, d.h. 3 Führungskräfte zu je 1/3, 2 Endbenutzer zu je 1/4 und 1 und 1/2 Entwickler die an dem Projekt Auftragsbearbeitung mitwirken. Der Test wurde 3 Mal durchgeführt.

Gruppe Benutzer: Die Bewertung durch die Anwender ergab ein durchschnittlich befriedigendes Ergebnis mit 59 Punkten.

Kommentar des Projekttestes: Die Bewertung ist durchschnittlich. Ein solches Projekt muss hohe Belastungen und eine wechselhafte Teamdynamik verkraften. Das Endergebnis hat wahrscheinlich eine geringere Funktionalität und Qualität und wird mehr kosten und später fertig gestellt* werden als erwartet. Projekte in dieser Kategorie können von einem professionelleren Projektmanagement am meisten profitieren.

Gruppe Geschäftsführer: Bei der Befragung des Managements fiel der Projekttest durchschnittlich aus, es wurden 52 Punkte erreicht.

Kommentar des Projekttestes: Eine Bewertung in dieser Kategorie liegt bereits über dem Industriedurchschnitt. Ein solches Projekt hat gute Chancen, seine Produktziele entweder in der festgelegten Zeit- oder in der festgelegten Kostenbandbreite zu erreichen, aber wahrscheinlich nicht alles gleichzeitig.

Entwicklungsteam: Laut Ergebnis des Projekttestes hat das Vorhaben aus der Sicht der Entwickler mit durchschnittlich 82 Punkten sehr gute Chancen.

Kommentar vom Projekttest: Ein Projekt auf dieser Stufe liegt deutlich über dem Industriedurchschnitt. Ein solches Projekt hat eine hohe Wahrscheinlichkeit, sowohl seine Produkt- als auch seine Termin- und Kostenziele annähernd zu erreichen.

Anmerkung: Die vorhandenen Kenntnisse des Entwicklungsteams über Projektarbeiten allgemein begünstigen die Projektbewertung, was bei der Gesamtbetrachtung des Projektmanagements berücksichtigt werden muss.

Ergebnis nach Zusammenfassung der Gruppen:

Die unterschiedlichen Resultate aus den drei Gruppen sind auf den ersten Blick bedenklich. Doch nach der Analyse der gruppierten Projektbewertung konnten diese Unstimmigkeiten erklärt werden. Denn die unterschiedlichen fachlichen Kenntnisse der Gruppen

führten dazu, das Fragen nicht richtig verstanden oder wiedergegeben wurden, die auf dem Bewertungsbogen aufgeführt waren. Nun stellt sich die Frage, wie dies vermeidbar ist. Wie schon erwähnt, können Schulungen nur in einem begrenzten Rahmen durchgeführt werden, d.h. eine Ausweitung der Schulung würde den Fehler zwar wahrscheinlich beseitigen, aber zusätzliche Schulungen sind nicht immer praktikabel. Hinzu kommt, dass nicht jeder das Projektmanagement und dessen Hintergründe unbedingt beherrschen muss. Die daraus entstehenden Defizite müssen anderweitig kompensiert werden.

Ein Ausgleich dieses Mangels konnte mit dem gemeinsamen Ausfüllen der Fragebogen durch Projektleiter und Teammitglied erzielt werden. Die Erläuterungen der Fragen halfen den Teammitgliedern bei der Beantwortung. Bei Projekten mit unterschiedlichen Fachkenntnissen der Teammitglieder kann zwar dieser Test nur bedingt zur Bewertung herangezogen werden, eine Tendenz lässt sich jedoch auf jeden Fall feststellen. In der Praxis müssen auch Projekte mit verschiedenen Mitarbeitern unterschiedlicher Vorkenntnisse durchführbar sein. Die Anpassung des Testverfahrens an die Ansprüche der Teammitglieder beschreibt eine weitere Variante, einen Ausgleich der Defizite zu erreichen.

Zusammengefasst kann festgestellt werden, dass dem Projekt mindestens gute Chancen für einen erfolgreichen Abschluss zugeordnet werden können.

6.6.4 Ergebnisse der Geschäftsprozessoptimierung

Stellvertretend für mehrere Neuerungen und Verbesserungen der Geschäftsprozesse, welche die Auftragsbearbeitung betreffen oder berühren, seien Verfahren in der Auftragsbearbeitung und Fertigung genannt. Wie bei der Untersuchung und Klassifizierung der Unternehmensorganisation bereits festgestellt wurde, existierten unter anderem Defizite bei der Organisation und dem Fluss der Fertigungsdokumente. Ein erarbeitetes Modell beschreibt das neue Verfahren, das bereits im Pflichtenheft angekündigt wurde. Dieses Verfahren beschreibt die Einbindung der Terminals zur Datenerfassung. Die Mitarbeiter aus der Fertigung werden sukzessive an die Einbeziehung der Terminals herangeführt, die den jeweiligen Arbeitsgangstatus der Datenbank melden. Die Schulungen, die vom Beauftragten der Qualitätssicherung durchgeführt werden, informieren die Mitarbeiter über den Zweck der Betriebsdatenerfassung (Terminals) und bilden sie im Umgang mit ihr aus. Es wurden Flussdiagramme entworfen, die den Mitarbeitern die Prozesse näher bringen um diese auch verifizieren zu können. Unter Berücksichtigung der verschiedenen Fachkenntnisse wurde das ursprüngliche Diagramm stark abstrahiert und veröffentlicht. Es befindet sich an jedem Terminal zur Erinnerung an die Inhalte der Schulung. Im Anhang dieser Arbeit kann der Weg vom Original-Flussdiagramm zur Simplifizierung nachvollzogen werden (vgl. Abbildung 8-17 und Abbildung 8-18).

Inhalte des Verfahrens:

- Dokumentarische Begleitung des Artikels
- Handhabung der fertigungstechnischen Information während der Produktion
- Umgang mit Fertigungsdokumenten

Jeder betroffene Mitarbeiter erhält eine Unterweisung, die er quittieren muss. Darin sind die Inhalte des Verfahrens beschrieben. Zusätzlich findet eine Einweisung am Terminal statt. Eine kurze Prüfung in Form eines Fragebogens über die Unterweisungsthemen bildet den Abschluss der Maßnahme. Wie schon erwähnt, können die Flussdiagramme im Anhang eingesehen werden, ferner sind der Fragebogen und dazugehörige Unterlagen in der Dokumentation des Projektes wiederzufinden. Die Gestaltung und der Inhalt der Unterweisung machen deutlich, wie eng das Verständnis von Geschäftsprozessen mit der eigentlichen Programmanwendung, hier durch simple Betriebsdatenerfassung, gekoppelt ist. Die Bilanz ergibt eine einfache und durchgängige Informationsbeschaffung und gewährleistet den transparenten und aktuellen Informationsbestand. Die Ausarbeitung der Projektgruppe, die das neue Verfahren formulierte, fand in Anlehnung an die Prozessanalysen, die Optimierungen zur Folge hatten, statt.

6.6.5 Wechselwirkungen zwischen Anwender und Entwickler

Vorschläge zur Verbesserung und Vereinfachung der Software hinsichtlich ihrer Bedienung und der zu erwartenden Reaktion bei Eingaben sind am effektivsten vom Benutzer selbst zu erkennen und von ihm ohne Zeitversatz an die Verantwortlichen weiterzuleiten. Bisher konnten kaum negative Erfahrungen gesammelt werden. Die Reaktionen der Anwender sind anfangs pessimistisch, wie bei der Einführung von Standardsoftware auch, später jedoch wachsen Anwender und Entwickler zu einem Team zusammen. Wechselwirkungen, die nachhaltige Auswirkung auf den Geschäftsprozess hatten, wurden nicht registriert. Lediglich harmlose Effekte konnten aufgenommen werden, diese sind aber nicht unbedingt mit dem Konzept in Zusammenhang zu bringen. Die nachfolgend beschriebene Situation kann bei jeder Softwareneuerung auftreten. Das aufgeführte Beispiel betrifft die Schnittstelle zum Lieferanten (Auftragsabschluss, vgl. Abbildung 6-3, Lieferschein, Frachtbrief, Rechnung usw.). Die Mitarbeiter bilden hier als Softwareanwender den Berührungspunkt der zwei Kontaktstellen.

Beispiel einer Reaktion während der Projektphasen:

Negative Reaktionen, die aber nicht zu akzeptieren waren und sind, kamen beispielsweise von einem Lieferanten. Dessen seit Jahren eingefahrene Routine stimmte nicht mehr mit den Neuerungen des Projektes/der Software überein. Innovationen und Veränderungen waren und sind für ihn (den Lieferanten) Fremdwörter. Dabei ging es nur um auf Pa-

pier gebrachte Listen, deren Format nicht dem vergangenen entsprach. Dieser Lieferant muss damit leben, dass jetzt der korrekte Firmenname und nicht irgendein willkürlicher Suchname, der aus den Anfängen der EDV stammt und so genannten Suchmatches entsprach, auf der erwünschten Liste erscheint und sich dadurch die Sortierung ändert (z.B. früher Bahn AG, heute Deutsche Bahn AG). Solche Probleme sind zwar dilettantisch, aber zeitaufwändig. Die meisten negativen Reaktionen basieren auf Oberflächlichkeiten und/oder zwischenmenschlichen Kommunikationsproblemen. Diese Gegebenheit ist weder technischer Natur noch projekt- bzw. konzeptabhängig, sie ist den Menschen eigen und unvermeidbar. Damit geschickt umzugehen ist die Aufgabe des Projektmanagements.

6.7 Ausblicke auf das Komponentendesign basierend auf Webservices

Für künftige Konzeptanwendungen, die weborientiert und plattformunabhängig laufen sollen, wird die Benutzeroberfläche auf ein Webdesign aufbauen und die Kommunikation bzw. der Datenaustausch muss über Webdienste zu realisieren sein. Der schematische Aufbau dieses Entwurfs kann folgendermaßen angenommen werden (siehe Abbildung 6-39).

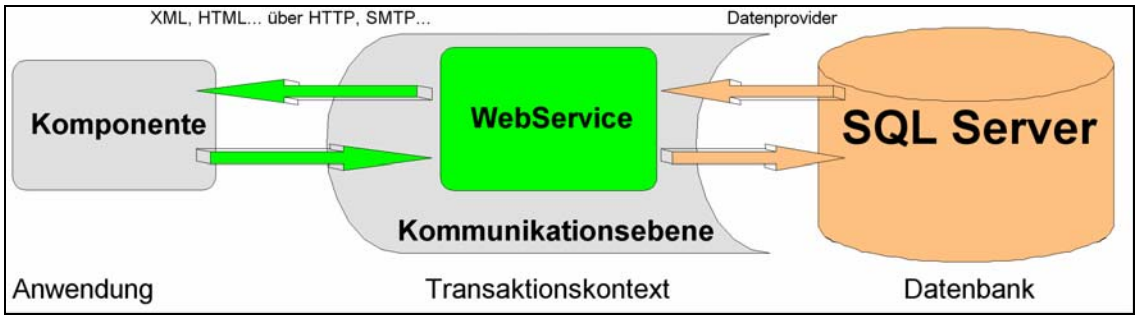

Abbildung 6-39 Webbasierendes Komponentendesign

Die gesamte Geschäftslogik befindet sich ohnehin in der Datenbank, demzufolge müssen lediglich das Userinterface und der Kommunikationskontext umgestaltet werden. Der Vorteil dieser Architektur ist die uneingeschränkte Plattformunabhängigkeit und die Autonomie der Clients. Diese Architektur kann auch dann genutzt werden, wenn der Client windowsbasierend sein soll, der Server nicht vor Ort, sondern irgendwo im Internet steht und die Daten über Webdienste angefordert werden.

6.8 Fazit und Bewertung des Projektes

Angewendet auf den IST-Zustand bezüglich der eingesetzten Software entlang der Wertschöpfungskette (vgl. Abbildung 2-2) ergibt sich folgendes Ergebnis: Die in Unternehmen eingesetzten Systemlösungen bewältigen die Aufgabe Geschäftsprozesse in allen Bereichen zu begleiten. Ferner stimmen die in der Software abgebildeten Prozesse mit den Ansprüchen des Unternehmens überein.

In der Abbildung 6-40 ist deutlich zu erkennen, dass die eingesetzte Individualsoftware diesen Forderungen gerecht wird. Über alle Bereiche hinweg besteht ein durchgängiger Datenfluss mit einer gemeinsamen Datenbasis. Selbst Programme, die andere Funktionen besitzen, wie CAD (Konstruktion) oder CAM (NC-Programmierung), kommunizieren über Schnittstellen mit der Systemlösung und gewähren somit einen anwendungsübergreifenden, einheitlichen Datenfluss. Informationslücken oder Fehlerquellen, die im Zusammenhang mit der eingesetzten Software stehen, werden so minimiert. Folglich bietet die Systemlösung eine optimale Grundlage Informationen schnell und aussagekräftig bereitzustellen sowie alle Prozesse transparent und nachvollziehbar zu begleiten.

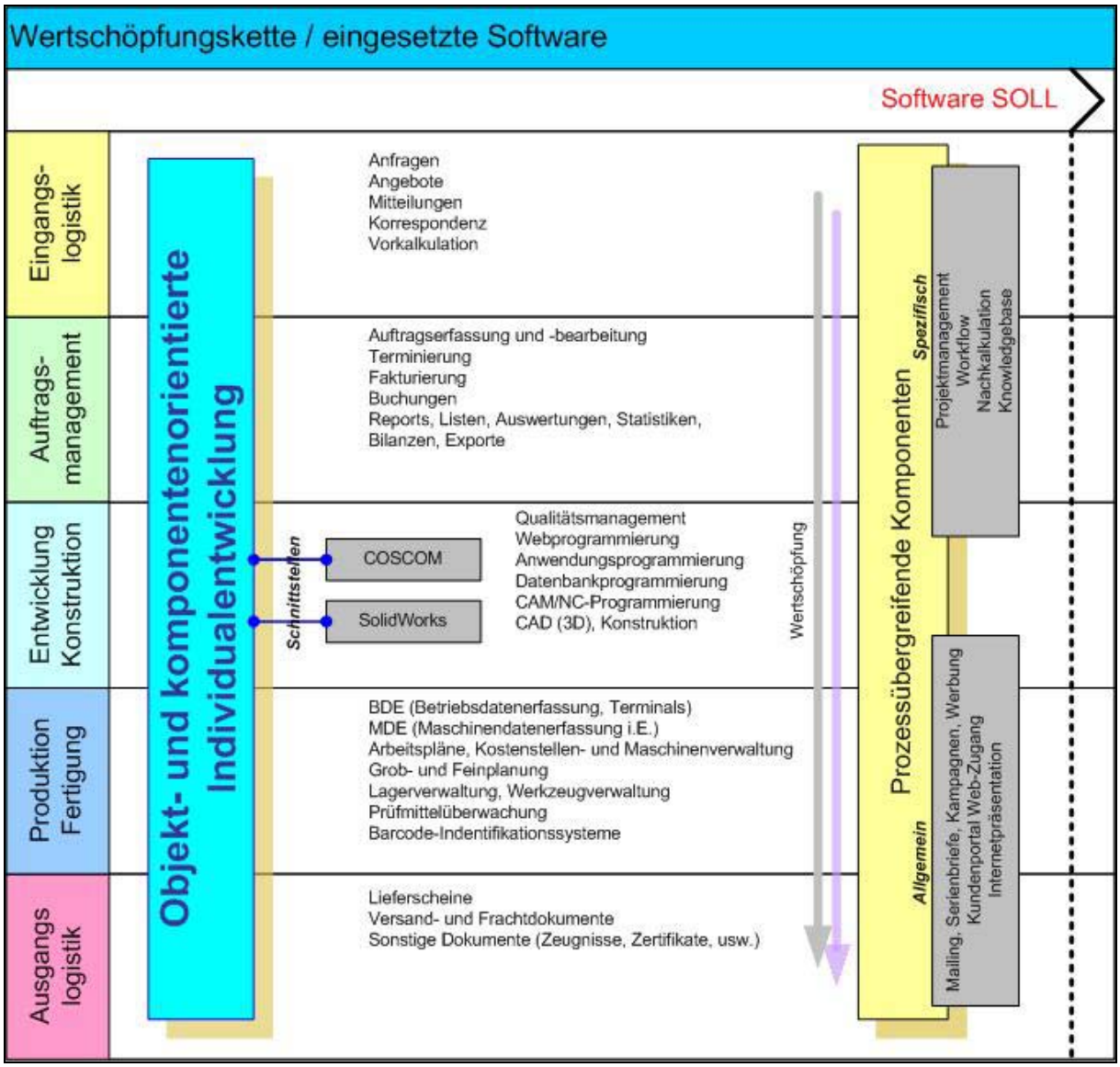

Abbildung 6-40 Eingesetzte Software (SOLL-Zustand)

Ähnlich positive Aussagen können auch für die Betrachtung eines einzelnen Geschäftsprozesses, hier die Auftragsbearbeitung gemäß Abbildung 2-3 gemacht werden. Eine abgerundete Prozessbegleitung für die Auftragsbearbeitung garantiert ein Minimum an Fehlerquellen bei der Datenpflege innerhalb dieses Bereiches. Die direkte Verknüpfung mit dem Artikelstamm vermeidet Zweideutigkeiten bei der Auftragsdefinition. Außerdem können angrenzende Prozesse Informationen aus derselben Datenbasis nutzen und verhindern somit einen Mehraufwand bei Recherchen. Änderungen am Produkt oder am Auftrag sind sofort in jedem Bereich feststellbar. Weiterhin ist bei Bedarf ein schneller Zugriff auf auftrags- und produktbezogene Dokumente bzw. Unterlagen möglich. Durch die nahtlosen Übergänge zu den anschließenden Geschäftsprozessen werden Warte- und Stillstandszeiten vermieden und somit Durchlaufzeiten für Aufträge erheblich verkürzt.

Die Anwendung des Konzepts in Verbindung mit der .NET-Technologie führt zu dem positiven Ergebnis gemäß Abbildung 6-41.

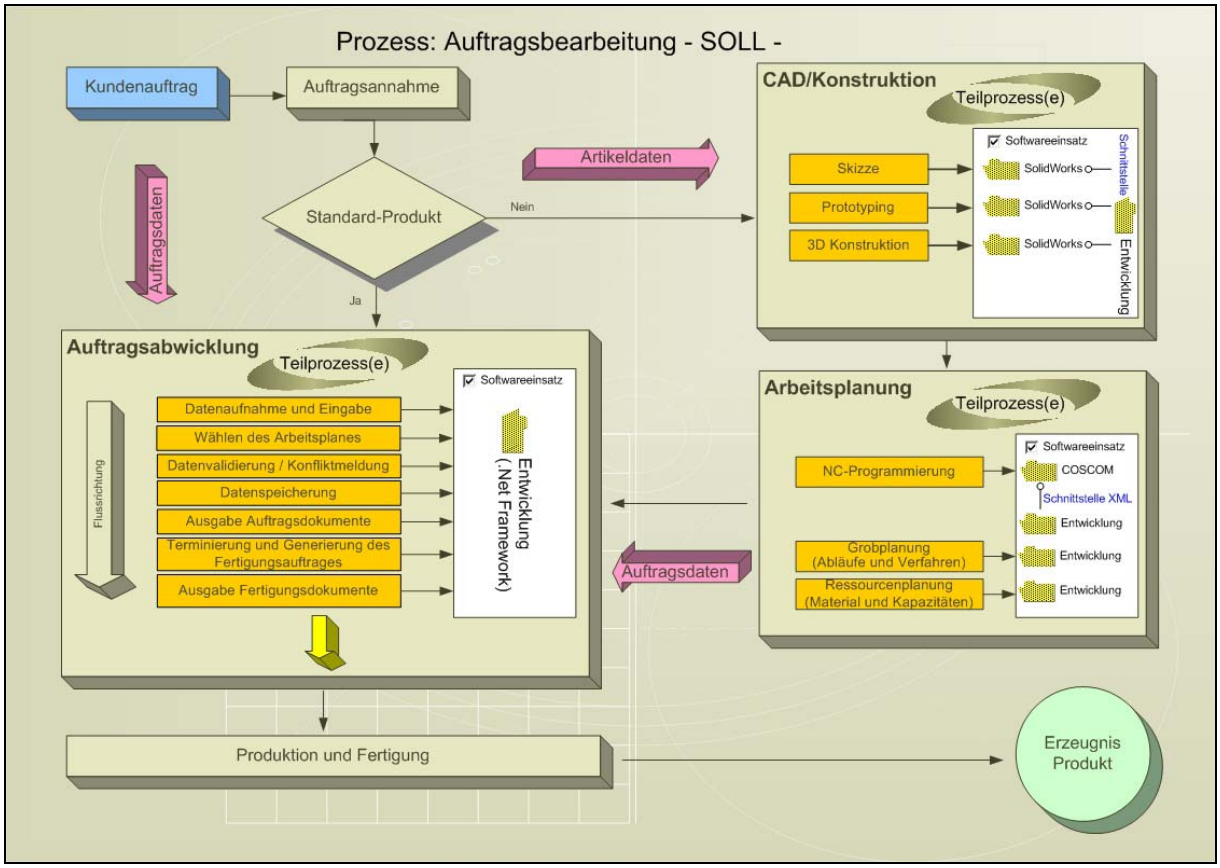

Abbildung 6-41 Geschäftsprozess Auftragsbearbeitung (SOLL-Zustand)

Nach mittlerweile anderthalb jährigem Einsatz der Auftragskomponente kann festgestellt werden, dass keine erwähnenswerten Beeinträchtigungen des Geschäftsprozesses in Kauf genommen werden mussten. Ausschlaggebend ist das zugrunde liegende Konzept für die Komponentenentwicklung mit dem dazugehörigen Projektmanagement. Hinzu kommt, dass die Teammitglieder und die Geschäftsführung sich der Innovation bewusst waren und eventuellen Komplikationen prophylaktisch entgegentraten. Nur wenn beide Seiten, Entwickler (Anbieter) und Anwender (Kunde), die Komplexität des Projektes erkennen und die notwendige Motivation zur Lösung von Aufgaben mitbringen, ist der Erfolg garantiert. Bei dem Anwender waren nicht nur diese Voraussetzungen gegeben, auch die Notwendigkeit der zusätzlichen Neuerung in der Umbruchphase, von der konventionellen Programmierung hin zur .NET Framework-Lösung, wurde erkannt. Dieser Umbruch stellt einen Generationswechsel und Quantensprung in der Programmierung dar. Selbst dieser fundamentale Wandel konnte bei dem Anwender gänzlich ohne Probleme vollzogen werden. Innerhalb kürzester Zeit wurden alle Voraussetzungen geschaffen, um auf dem

Framework aufzusetzen. Natürlich ist der relativ frühe Wechsel zu Framework aus der Gesamtprojektsicht Voraussetzung für das reibungslose Gelingen.

Nach nunmehr über zweijähriger .NET Framework-Entwicklung und dessen Einsatz kann ohne Zweifel festgestellt werden, dass die richtige Entscheidung getroffen wurde. In diesem Zeitraum wurden nicht nur die Auftragsbearbeitung und ihre vorausgesetzten Module/Komponenten mit dem Framework umgesetzt, auch einfachere und komplexere Komponenten konnten erfolgreich bereitgestellt werden. Mittlerweile ist das Ausmaß der Anwendung auf über 30 Komponenten angewachsen, die zu über 80 Prozent täglich genutzt werden. Nicht berücksichtigt sind dabei Hilfskomponenten, wie z.B. Urlaubsplanung, Stundenabrechnung, SQL-Tools und diverse kleinere Applikationen, die den Geschäftsablauf unterstützen. Die folgende Tabelle gibt eine Übersicht über den derzeitigen Entwicklungsstand des gesamten Projektes. Die Expansion der Anwendung hat bereits einen Funktionsumfang zur Folge, der annähernd mit PPS-Systemen zu vergleichen ist. In der nachstehenden Tabelle sind die Komponenten und deren Status sowie die bisherige Entwicklungsdauer (Entwicklerstunden) alphabetisch aufgelistet.

Status: 1 = unter Vorbehalt fertig, da Software nie „fertig" ist

Status: 2 = momentan fortgeschrittene Entwicklung

Status: 3 = aktueller Entwicklungsbeginn

Status: 4 = geplante Komponente

Komponentenname	**Status**	**Dauer**	**Beschreibung**
Anfragen	1	200	Verwaltung von Anfragen an Firmen
Angebote	1	200	Angebotserstellung für Kunden
Arbeitsgänge	1	100	Arbeitsgangdefinition (Arbeitsplan-Grundlage)
Arbeitspläne	1	400	Kommissionsunabhängige Arbeitsvorschriften
Artikel	1	400	Stammdaten Fertigungsartikel (Produktdefinitionen)
Aufträge	1	200	Kundenauftragsbearbeitung
Baugruppen	1	300	Hierarchische Stücklisten
Benutzer	1	50	Benutzerverwaltung

Buchungen	1	300	Eingangs- und Ausgangsbelege
DIN-Teile	1	200	Stammdaten DIN-Teile
Einkauf/Beschaffung	2	300	Einkauf- und Bestellwesen, Beschaffungen
Feinplanung	2	600-750	Feinplanung der Fertigungsaufträge
Fertigungsaufträge	1	600	Kommissionsabhängige Arbeitsvorschriften
Firmeneinsatz	3	50	Bilanzspezifisches Modul
Globalsuche	1	50	Suche über mehrere Objekt- bzw. Komponententypen
Inventar	3	50	Bilanzspezifisches Modul
Knowledge	2	100	Wissensdatenbank
Kostenstellen	1	250	Gruppierung der Maschinen/Arbeitsorte (Arbeitsplan-Grundlage)
Kunden	1	100	Kundenstamm
Lagerverwaltung	1	400	Bestandsverwaltung der Module Artikel, DIN-Teile, Rohmaterial, Werkzeuge usw.
Lieferanten	1	100	Lieferantenstamm
Lieferscheine	1	200	Lieferscheinwesen
Maschinen/Arbeitsorte	1	100	Definition der Arbeitsorte/Maschinen (Arbeitsplan-Grundlage)
Mitarbeiter	1	100	Mitarbeiter- und Personalverwaltung
NC-Programme	1	100	CAM-Verwaltung
Projekte	3	150	Projektverwaltung / Objektgruppierung
Rechnungen	1	200	Rechnungswesen
Rohmaterial	1	100	Stammdaten Rohmaterial
Scans	1	50-100	Stammdaten sonstiger Dokumente
Werkzeuge	1	300	Werkzeugverwaltung
Workflow	1	300	Workflow-Mechanismus
Zeichnungen	1	100	Stammdaten technischer Dokumente
Zukaufteile	1	100	Stammdaten der Zukaufteile

Abbildung 6-42 Komponentenübersicht

Die Summe der investierten Entwicklerstunden entspricht etwa zwei Jahren Entwicklungszeit bei anderthalb Programmierern. Zu beachten ist, dass alle Komponenten neu entworfen und entwickelt wurden und diese Prototyp-Zeit nicht mit einer Zeit zu vergleichen ist, bei der das Konzept mit den Komponenten- und Datenbankvorlagen angewendet wird. Diese Erkenntnisse lassen die Annahme zu, dass bei einer Anwendung des Konzeptes mit den entsprechenden Vorlagen in diesem Umfang die Entwicklungszeit um die Hälfte reduziert wird. D.h., wird von einer ca. einjährigen Zeitspanne für die Individualentwicklung ausgegangen, steht sie sogar aus der Sicht des zeitlichen Aufwandes gegenüber der konventionellen Softwareeinführung, in nichts nach. Die finanzielle Belastung der Unternehmen hingegen ist geringer und der Nutzen höher, da die Software absolut kundenorientiert ist. Ein weiteres Argument ist die Tatsache, dass das Resultat der Entwicklung nach dem Konzept realer Gegenstand im Geschäftsalltag des Anwenders ist und sich gegenwärtig bewährt. Somit ist die erfolgreiche Anwendung des Konzeptes bestätigt und praktisch bewiesen.

7 ZUSAMMENFASSUNG

Abschließend kann festgestellt werden, dass die Unzufriedenheit mit der momentanen Situation vorhandener Systemlösungen, vor allen für den mittelständischen Bereich, durch die folgenden softwaretechnischen Punkte gekennzeichnet wird: fehlende Industriestandards, keine Interoperabilität, ungenügende Integrationsfähigkeit, technologisch überholte Programmierung, hoher Finanzaufwand für Kunden und fehlende Kundenorientierung. Diese Erkennungszeichen sind nicht unbedingt auf den ersten Blick festzustellen, erst beim genauen Betrachten der einzelnen Systeme werden diese Merkmale offensichtlich. Da bei der Anschaffung oder Erweiterung von Systemlösungen Vorbetrachtungen zwar stattfinden, aber die Inhalte technologischer Aspekte unzureichend einbezogen werden, ist das Ergebnis die heutige Situation. Am schwersten trifft es dabei die mittelständischen Unternehmen, denn deren Finanzbudget ist begrenzt. Darüber hinaus vertreten sie meistens die Branchen der schwer abzubildenden Geschäftsprozesse, wie Lohn-, Auftrags- und Einzelfertiger sowie Sonder- und Spezialmaschinenbau.

Gerade die Kundenorientierung der Software, mit der viele Anbieter werben, ist in Wahrheit keine echte Kundenorientierung. Unternehmen, die auf stark dynamischen und flexiblen Geschäftsabläufen basieren, können Systemlösungen, die für eine Branche programmiert wurden, nur unzureichend nutzen. Hinzu kommen die Unterschiede innerhalb einer Branche, sie verlangen unwiderruflich nach individuellen Lösungen. Die Realität zeigt, dass Software nur zum Teil genutzt wird, Datenbestände lückenhaft sind, der Umgang mit der Software kompliziert ist und aus Anwendersicht der Softwareaufbau nicht logisch erscheint. Viele kleine oder mittelständische Unternehmen setzen deshalb gar keine Systemlösungen ein, sondern behelfen sich mit einer Dokumentenverwaltung in Verzeichnisstrukturen. Ein weiterer Umstand ist die fehlende Kompetenz durch Beratungsmangel, sie hindert kleinere Unternehmen daran, Software effektiv einzusetzen. Fehlendes Fachpersonal im Unternehmen, was bei kleinen Unternehmen verständlich ist, oder hohe Kosten für eine externe Beratung, tragen zu diesem Umstand bei.

Ein Lösungsansatz für diese Problematik stellt das hier vorgelegte Konzept dar. Basierend auf der neuen Generation der Softwareentwicklung und einer innovativen Datenbank- und Softwarearchitektur, wurde ein Weg gezeigt und nachgewiesen, wie individuelle Lösungen realisiert werden können. Die dazugehörigen fortschrittlichen Entwicklungswerkzeuge unterstützen die Verwirklichung des Konzeptes. Die Durchführung des Konzepts erfolgt auf der Grundlage eines konzeptbezogenen Projektmanagements, welches komponentenorientiert aufgebaut ist. Die gesamte Entwicklung geschieht vor Ort beim Kunden, parallel zum Geschäftsalltag und unter Einbeziehung der Mitarbeiter/Anwender.

Vorhandene Vorlagen für Komponenten und das feststehende Datenbankdesign komprimieren die Entwicklungszeit bis auf bekannte konventionelle Software-Einführungszeiten. Das Datenbankdesign stützt sich auf ein objektstrukturiertes Tabellenlayout und die zentrale Ablage der Geschäftslogik in der Datenbank. Die Softwarearchitektur ist durch mehrere Ebenen und eine funktionale Trennung charakterisiert (komponentenorientiert, modular, objektorientiert). Die Datenbankkommunikationsebene bildet eine selbstständige Komponente, die auf einer direkten Datenbankverbindung, auf Remoting oder Webservices beruht. Die Komponente oder auch das Assembly besteht aus drei Modulen, die von der Datenkapslungsklasse, dem Interface und der Benutzeroberfläche repräsentiert werden.

Die Kooperation zwischen Entwickler und Anwender in einem Projektteam garantiert nicht nur eine wirkliche Kundenorientierung, sondern fördert die Kommunikation zwischen den unterschiedlichen Parteien. Die Schulungen der Anwender werden synchron zur Komponentenveröffentlichung durchgeführt und beinhalten die Themen der neuen Komponente. So ist der Schulungsinhalt übersichtlicher und dadurch gegenüber komplexen Blockschulungen verständlicher für den Anwender. Durch die komponentenorientierte Methode wird das mögliche Optimierungspotential der Geschäftsprozesse voll ausgenutzt und das Projektcontrolling lässt sich einfach realisieren.

Die Anwendung des Konzeptes wurde bei kleinen mittelständischen Maschinenbau-Unternehmen erfolgreich nachgewiesen. In dieser Arbeit wurde speziell die Entwicklung der Auftragskomponente detailliert begleitet. Die Entwicklung der anderen Komponenten erfolgte ebenfalls auf der Basis dieses Konzeptes, d.h. eine Bestätigung dieser Methode konnte mehrfach demonstriert werden. Mittlerweile nutzen mehr als 50 Mitarbeiter die Software über einen PC-Arbeitsplatz, mit Hilfe mobiler Datenerfassung und/oder an Terminals. Die Resonanzen im Unternehmen, die im Zusammenhang mit der Softwareentwicklung und Anwendung stehen, sind positiv und unterstreichen die erfolgreiche Konzeptanwendung.

8 ANHANG

```
Public Sub ExecuteProcedur(ByVal sID As Integer, ByVal sCL As Short, ByVal sStructurList As Object, _
                                                                    ByVal sProcedur As String)
    Try
        'bei dieser Prozedur müssen die ObjektID und KlassenID immer Übergeben werden,
        'die restlichen Parameter als Parameter-Struktur
        'die BenutzerID ist standardmäßig bekannt.
        '############################
        'Demo der Struktur (sStructurList), die übergeben werden muß:
        'Dim mData(1) As Object
        'Dim mPar1 As New MyParms("AGRelID", SqlDbType.Int, 4, ParameterDirection.Input, mID)
        'Dim mPar2 As New MyParms("AGRelCL", SqlDbType.SmallInt, 2, ParameterDirection.Input, mCL)
        'mData(0) = mPar1
        'mData(1) = mPar2
        'Ende Demo
        '############################
        If Not TypeOf sStructurList Is Array Then
            Call ShowMyMessage("Die zu übergebene Struktur-Liste entspricht nicht dem Paradigma!", _
                                                                                "Execute Procedur")
            Exit Sub
        End If
        Dim mArray As Object = sStructurList
        If Not mArray.Length > 0 Then
            Call ShowMyMessage("Die zu übergebene Struktur-Liste hat die Länge null!", _
                                                                                "Execute Procedur")
            Exit Sub
        End If
        If sProcedur.Trim = "" Then
            Call ShowMyMessage("Der Prozedur-Aliasname ist leer!", "Execute Procedur")
            Exit Sub
        End If

        Dim i As Integer
        Dim mCount As Integer = mArray.Length + 2
        If mMyProcedur Is Nothing Then mMyProcedur = New SQLProcedur.cMyProcedur()

        Dim pName(mCount) As String
        Dim pType(mCount) As SqlDbType
        Dim pSize(mCount) As Integer
        Dim pDirection(mCount) As ParameterDirection
        Dim pValue(mCount) As Object

        i = 0 : pName(i) = "@ObjektID" : pType(i) = SqlDbType.Int : pSize(i) = 4
        pDirection(i) = ParameterDirection.Input : pValue(i) = sID

        i = 1 : pName(i) = "@KlassenID" : pType(i) = SqlDbType.SmallInt : pSize(i) = 2
        pDirection(i) = ParameterDirection.Input : pValue(i) = sCL

        For i = 2 To mCount - 1
            If TypeOf mArray(i - 2) Is MyParms Then
                Dim x As MyParms = mArray(i - 2)
                pName(i) = "@" & x.parName : pType(i) = x.parTyp : pSize(i) = x.parSize
                pDirection(i) = x.parDir : pValue(i) = x.parValue
            Else
                Call ShowMyMessage("Die Struktur " & i.ToString & " entspricht nicht dem Paradigma, " & _
                                                "der Vorgang wird abgebrochen!", "Execute Procedur")
                Exit Sub
            End If
        Next

        pName(i) = "@BenutzerID" : pType(i) = SqlDbType.Int : pSize(i) = 4
        pDirection(i) = ParameterDirection.Input : pValue(i) = mMyBenutzerID

        With mMyProcedur
            .ParName = pName
            .ParType = pType
            .ParSize = pSize
            .ParDirection = pDirection
            .ParValue = pValue
            Call .getSQLDatabase(mMyConnection, sProcedur, i, True, False)
        End With

    Catch e As Exception
        Call ShowMyErrorMessage(e, "Execute Procedur")
    End Try
End Sub
```

Abbildung 8-1 Prozeduraufbau für Datenübergabe

```
Alter Trigger TB_AUFTRAG_Update
On dbo.TB_AUFTRAG
For Update
As

DECLARE @mObjektID int,
        @mKlassenID smallint,
        @mKommissionsNR int,
        @mClientID int,
        @mClientCL smallint,
        @mChildID int,
        @mChildCL smallint,
        @mLiefertermin datetime,
        @mKorrekturtermin datetime,
        @mFertigungsstand tinyint,
        @mPrioritaet bit,
        @mAuftragsstatus tinyint,
        @mPositionsPreis money,
        @mPositionsPreisStr varchar(20),
        @mAuftragsmenge int,
        @mVariante varchar(200),
        @mVariantenParentID int,
        @mVariantenParentCL smallint,

        --####
        @mFAAktiv bit,
        @mNewPrioritaet bit,
        @mPreChildID int,
        @mPreChildCL smallint,
        @mBenutzerID int,
        @mInnerID int,
        @mInnerCL smallint

SELECT  @mObjektID = ObjektID FROM inserted
SELECT  @mKlassenID = KlassenID FROM inserted
SELECT  @mKommissionsNR = KommissionsNR FROM inserted
SELECT  @mFertigungsstand = Fertigungsstand FROM inserted
SELECT  @mAuftragsstatus = Auftragsstatus FROM inserted
SELECT  @mLiefertermin = Liefertermin FROM inserted
SELECT  @mKorrekturtermin = Korrekturtermin FROM inserted
SELECT  @mPrioritaet = Prioritaet FROM inserted
SELECT  @mPreChildID = ChildID FROM deleted
SELECT  @mPreChildCL = Childklasse FROM deleted
SELECT  @mChildID = ChildID FROM inserted
SELECT  @mChildCL = Childklasse FROM inserted
SELECT  @mClientID = ClientID FROM inserted
SELECT  @mClientCL = Clientklasse FROM inserted
SELECT  @mPositionspreis = Positionspreis FROM inserted
SELECT  @mAuftragsmenge = Menge FROM inserted
SELECT  @mBenutzerID = GeaendertVon FROM inserted
```

Abbildung 8-2 Trigger (Teil 1 von 6) TB_AUFTRAG

```
IF (SELECT  COUNT(TB_FERTIGUNGSAUFTRAG.ObjektID)
    FROM    TB_FERTIGUNGSAUFTRAG
    WHERE   TB_FERTIGUNGSAUFTRAG.AuftragsID = @mObjektID
    AND     TB_FERTIGUNGSAUFTRAG.Auftragsklasse = @mKlassenID) > 0

    BEGIN
        SELECT  @mFAAktiv = 1
    END
IF ISNULL(@mKorrekturtermin, 0) > @mLiefertermin
    BEGIN
        UPDATE  TB_AUFTRAG
        SET     TB_AUFTRAG.Prioritaet = @mPrioritaet,
                TB_AUFTRAG.GeaendertVon = @mBenutzerID,
                TB_AUFTRAG.GeaendertAm = GETDATE()
        WHERE   TB_AUFTRAG.ObjektID = @mObjektID
        AND     TB_AUFTRAG.KlassenID = @mKlassenID
        AND     TB_AUFTRAG.Prioritaet <> @mPrioritaet
    END
IF @mKorrekturtermin > @mLiefertermin AND @mFertigungsstand <> 4
    BEGIN
        UPDATE  TB_FERTIGUNGSAUFTRAG
        SET     TB_FERTIGUNGSAUFTRAG.FAPrioritaet = @mPrioritaet,
                TB_FERTIGUNGSAUFTRAG.Liefertermin = @mKorrekturtermin,
                TB_FERTIGUNGSAUFTRAG.Geaendert_Von = @mBenutzerID,
                TB_FERTIGUNGSAUFTRAG.Geaendert_Am = GETDATE()
        WHERE   TB_FERTIGUNGSAUFTRAG.AuftragsID = @mObjektID
        AND     TB_FERTIGUNGSAUFTRAG.Auftragsklasse = @mKlassenID
        AND     TB_FERTIGUNGSAUFTRAG.FAStatus <> 4          --nicht bei beendet!!!
        AND     (TB_FERTIGUNGSAUFTRAG.FAPrioritaet <> @mPrioritaet
        OR      TB_FERTIGUNGSAUFTRAG.Liefertermin <> @mKorrekturtermin)
    END
IF UPDATE(Liefertermin)
    BEGIN
        UPDATE  TB_FERTIGUNGSAUFTRAG
        SET     TB_FERTIGUNGSAUFTRAG.Liefertermin = @mLiefertermin,
                TB_FERTIGUNGSAUFTRAG.Geaendert_Von = @mBenutzerID,
                TB_FERTIGUNGSAUFTRAG.Geaendert_Am = GETDATE()
        WHERE   TB_FERTIGUNGSAUFTRAG.AuftragsID = @mObjektID
        AND     TB_FERTIGUNGSAUFTRAG.Auftragsklasse = @mKlassenID
        AND     TB_FERTIGUNGSAUFTRAG.FAStatus <> 4          --nicht bei beendet!!!
        AND     TB_FERTIGUNGSAUFTRAG.Liefertermin <> @mLiefertermin
    END
IF UPDATE(Menge)
    BEGIN
        UPDATE  TB_FERTIGUNGSAUFTRAG
        SET     TB_FERTIGUNGSAUFTRAG.FAMenge = @mAuftragsmenge,
                TB_FERTIGUNGSAUFTRAG.Geaendert_Von = @mBenutzerID,
                TB_FERTIGUNGSAUFTRAG.Geaendert_Am = GETDATE()
        WHERE   TB_FERTIGUNGSAUFTRAG.AuftragsID = @mObjektID
        AND     TB_FERTIGUNGSAUFTRAG.Auftragsklasse = @mKlassenID
        AND     TB_FERTIGUNGSAUFTRAG.FAStatus <> 4          --nicht bei beendet!!!
        AND     TB_FERTIGUNGSAUFTRAG.FAMenge <> @mAuftragsmenge
    END
```

Abbildung 8-3 Trigger (Teil 2 von 6) TB_AUFTRAG

```
IF @mFAAktiv = 1
    BEGIN
        IF @mPreChildID <> @mChildID OR @mPreChildCL <> @mChildCL
            OR @mClientID = 0 OR @mClientCL = 0

            BEGIN
                RAISERROR ('Dieser Auftrag ist in der Fertigung!
Der Artikel kann nicht geändert bzw. der Kunde nicht gelöscht werden!', 16, 1)
                ROLLBACK TRANSACTION
            END
    END

IF @mAuftragsstatus = 4 --beendet
    BEGIN

        UPDATE  TB_BESTELLUNG
        SET     TB_BESTELLUNG.Bestellstatus = 5,
                TB_BESTELLUNG.Bearbeitet_von = @mBenutzerID,
                TB_BESTELLUNG.Bearbeitet_am = GETDATE()
        WHERE   TB_BESTELLUNG.Bestellt_fuer_ID = @mObjektID
        AND     TB_BESTELLUNG.Bestellt_fuer_CL = @mKlassenID

        DELETE  FROM SUB_TB_ZWISCHENLAGER
        WHERE   SUB_TB_ZWISCHENLAGER.KommissionsNR = @mKommissionsNR

        DELETE  FROM SUB_TB_LAGER_RESERVIERUNG
        WHERE   SUB_TB_LAGER_RESERVIERUNG.Reserviert_fuer = @mKommissionsNR

        IF (SELECT  COUNT(ARCHIV_TB_AUFTRAG.ObjektID)
            FROM    ARCHIV_TB_AUFTRAG
            WHERE   ARCHIV_TB_AUFTRAG.ObjektID = @mObjektID
            AND     ARCHIV_TB_AUFTRAG.KlassenID = @mKlassenID) > 0

            BEGIN

                DELETE  FROM    ARCHIV_TB_AUFTRAG
                WHERE   ARCHIV_TB_AUFTRAG.ObjektID = @mObjektID
                AND     ARCHIV_TB_AUFTRAG.KlassenID = @mKlassenID

            END

        INSERT  INTO    ARCHIV_TB_AUFTRAG
        SELECT  * FROM TB_AUFTRAG
        WHERE   TB_AUFTRAG.ObjektID = @mObjektID
        AND     TB_AUFTRAG.KlassenID = @mKlassenID

        UPDATE  ARCHIV_TB_AUFTRAG
        SET     ARCHIV_TB_AUFTRAG.GeaendertVon = @mBenutzerID,
                ARCHIV_TB_AUFTRAG.GeaendertAm = GETDATE()
        WHERE   ARCHIV_TB_AUFTRAG.ObjektID = @mObjektID
        AND     ARCHIV_TB_AUFTRAG.KlassenID = @mKlassenID
```

Abbildung 8-4 Trigger (Teil 3 von 6) TB_AUFTRAG

```
        IF (SELECT  COUNT(TB_FERTIGUNGSAUFTRAG.ObjektID)
            FROM    TB_FERTIGUNGSAUFTRAG
            WHERE   TB_FERTIGUNGSAUFTRAG.AuftragsID = @mObjektID
            AND     TB_FERTIGUNGSAUFTRAG.Auftragsklasse = @mKlassenID) > 0

            BEGIN

                IF (SELECT  COUNT(ARCHIV_TB_FERTIGUNGSAUFTRAG.ObjektID)
                    FROM    ARCHIV_TB_FERTIGUNGSAUFTRAG
                    WHERE   ARCHIV_TB_FERTIGUNGSAUFTRAG.AuftragsID = @mObjektID
                    AND     ARCHIV_TB_FERTIGUNGSAUFTRAG.Auftragsklasse = @mKlassenID) > 0

                    BEGIN

                        DELETE  FROM ARCHIV_TB_FERTIGUNGSAUFTRAG
                        WHERE   ARCHIV_TB_FERTIGUNGSAUFTRAG.AuftragsID = @mObjektID
                        AND     ARCHIV_TB_FERTIGUNGSAUFTRAG.Auftragsklasse = @mKlassenID

                    END

                INSERT  INTO    ARCHIV_TB_FERTIGUNGSAUFTRAG
                SELECT  * FROM TB_FERTIGUNGSAUFTRAG
                WHERE   TB_FERTIGUNGSAUFTRAG.AuftragsID = @mObjektID
                AND     TB_FERTIGUNGSAUFTRAG.Auftragsklasse = @mKlassenID

                UPDATE  ARCHIV_TB_FERTIGUNGSAUFTRAG
                SET     ARCHIV_TB_FERTIGUNGSAUFTRAG.FAStatus = 4,
                        ARCHIV_TB_FERTIGUNGSAUFTRAG.Geaendert_Von = @mBenutzerID,
                        ARCHIV_TB_FERTIGUNGSAUFTRAG.Geaendert_Am = GETDATE()
                WHERE   ARCHIV_TB_FERTIGUNGSAUFTRAG.AuftragsID = @mObjektID
                AND     ARCHIV_TB_FERTIGUNGSAUFTRAG.Auftragsklasse = @mKlassenID

                DELETE FROM TB_FERTIGUNGSAUFTRAG
                WHERE   TB_FERTIGUNGSAUFTRAG.AuftragsID = @mObjektID
                AND     TB_FERTIGUNGSAUFTRAG.Auftragsklasse = @mKlassenID
            END

        UPDATE  TB_AUFTRAG
        SET     TB_AUFTRAG.State = 3,
                TB_AUFTRAG.GeaendertVon = @mBenutzerID,
                TB_AUFTRAG.GeaendertAm = GETDATE()
        WHERE   TB_AUFTRAG.ObjektID = @mObjektID
        AND     TB_AUFTRAG.KlassenID = @mKlassenID
```

Abbildung 8-5 Trigger (Teil 4 von 6) TB_AUFTRAG

```
        --Preis in Variantentabelle eintragen
        IF @mChildCL = 31
            BEGIN
                SELECT  @mVariantenParentID = @mChildID
                SELECT  @mVariantenParentCL = @mChildCL
            END
        ELSE IF @mChildCL = 135
            BEGIN
                SELECT  @mVariantenParentID = ISNULL(REL_ARTIKEL_ARBEITSPLAN.ParentID, 0),
                        @mVariantenParentCL = ISNULL(REL_ARTIKEL_ARBEITSPLAN.Parentklasse, 0)
                FROM    REL_ARTIKEL_ARBEITSPLAN
                WHERE   REL_ARTIKEL_ARBEITSPLAN.ChildID = @mChildID
                AND     REL_ARTIKEL_ARBEITSPLAN.Childklasse = @mChildCL
            END
        ELSE
            BEGIN
                SELECT  @mVariantenParentID = 0
                SELECT  @mVariantenParentCL = 0
            END

        IF @mPositionspreis > 0 AND @mAuftragsmenge > 0 AND @mVariantenParentCL = 31
            AND @mVariantenParentID <> 0 AND @mKommissionsNR > 0

            BEGIN
                SELECT  @mVariante = 'Preis aus Auftrag ' + 'KOM-' + RIGHT('000000' +
                                        convert(varchar, @mKommissionsNR), 6)
                                    + ' bei einer Stückzahl von: '
                                    + cast(@mAuftragsmenge as varchar)

                SELECT  @mPositionsPreisStr = cast(@mPositionsPreis as varchar)

                EXECUTE VariantenSave_100
                                        0,
                                        0,
                                        @mVariantenParentID,
                                        @mVariantenParentCL,
                                        0,                          --Primaer
                                        @mVariante,                 --Variante
                                        @mPositionspreisStr,        --Wert
                                        2,                          --WaehrungID    Euro
                                        6,                          --EinheitID     Stck.
                                        @mAuftragsmenge,
                                        0,                          --ClientID
                                        0,                          --ClientCL
                                        @mObjektID,
                                        @mKlassenID,
                                        @mBenutzerID,
                                        1,                          --dbAction
                                        0                           --ret
            END

        DECLARE InnerSekundaerKomAbmelde_Cursor CURSOR LOCAL FOR
```

Abbildung 8-6 Trigger (Teil 5 von 6) TB_AUFTRAG

```
        SELECT  REL_AUFTRAG_AUFTRAG.ChildID,
                REL_AUFTRAG_AUFTRAG.ChildCL
        FROM    REL_AUFTRAG_AUFTRAG
        WHERE   REL_AUFTRAG_AUFTRAG.ParentCL = @mKlassenID
        AND     REL_AUFTRAG_AUFTRAG.ParentID = @mObjektID

        OPEN InnerSekundaerKomAbmelde_Cursor
        FETCH NEXT FROM InnerSekundaerKomAbmelde_Cursor

        INTO    @mInnerID,
                @mInnerCL

        WHILE @@FETCH_STATUS = 0
            BEGIN

                UPDATE  TB_AUFTRAG
                SET     TB_AUFTRAG.Auftragsstatus = 4,
                        TB_AUFTRAG.GeaendertVon = @mBenutzerID,
                        TB_AUFTRAG.GeaendertAm = GETDATE()
                WHERE   TB_AUFTRAG.ObjektID = @mInnerID
                AND     TB_AUFTRAG.KlassenID = @mInnerCL
                AND     TB_AUFTRAG.Auftragsstatus <> 4

                FETCH NEXT FROM InnerSekundaerKomAbmelde_Cursor

                INTO    @mInnerID,
                        @mInnerCL

            END

        CLOSE InnerSekundaerKomAbmelde_Cursor
        DEALLOCATE InnerSekundaerKomAbmelde_Cursor

    END
```

Abbildung 8-7 Trigger (Teil 6 von 6) TB_AUFTRAG

```
Alter Procedure AuftragSave_100
                                @ObjektID int output,
                                @KlassenID smallint,
                                @AuftragsNR int output,
                                @ChildID int,
                                @ChildCL smallint,
                                @ClientID int,
                                @ClientCL smallint,
                                @Menge float,
                                @BestellNR varchar(50),
                                @Bestelldatum datetime,
                                @IhrZeichen varchar(50),
                                @Positionspreis money,
                                @PreisPlus money,
                                @Liefertermin datetime,
                                @Korrekturtermin datetime,
                                @Positionsinfo nvarchar(500),
                                @Printinfo nvarchar(500),
                                @Anmerkung nvarchar(500),
                                @APLInfo nvarchar(500),
                                @IsPrimaerkommission bit,
                                @Prioritaet bit,
                                @PrioritaetInfo varchar(100),
                                @ZBSkonto float,
                                @ZBTage smallint,
                                @EinheitID int,
                                @Sonderbedingungen nvarchar(500),
                                @CheckPreis bit,
                                @IstFertigung bit,
                                @FATermin datetime,
                                @PreisPlusPlus money,
                                @PreisPlusPlusInfo varchar(200),
                                @Listensortierung varchar(20),
                                @IstInclusiveChildren bit,
                                @OhneLTAusgabe bit,
                                @BenutzerID int,
                                @dbAction tinyint,
                                @ret bit output

AS
DECLARE @LastIDUpdate bit, @Gesamtpreis money, @mUrDate datetime,
        @mIsPrimaerkommission bit,
        @mIsChildkommission bit,
        @mOldIsPrimaerkommission bit


SELECT  @LastIDUpdate = 0
SELECT  @mUrdate = convert(datetime, '01.01.1900')
SELECT  @Gesamtpreis = (@Menge * (@Positionspreis + @PreisPlus)) + @PreisPlusPlus

IF ISNULL(@Gesamtpreis, 0) <= 0
    BEGIN
        SELECT  @CheckPreis = 1
    END
```

Abbildung 8-8 Prozedur Save (Teil 1 von 5) TB_AUFTRAG

```
IF @Liefertermin <= @mUrdate
    BEGIN
        SELECT  @Liefertermin = NULL
    END
IF @Korrekturtermin <= @mUrdate
    OR convert(varchar, @Liefertermin, 104) = convert(varchar, @Korrekturtermin, 104)
    BEGIN
        SELECT  @Korrekturtermin = NULL
    END
IF NOT @Korrekturtermin IS NULL
    BEGIN
        SELECT  @Prioritaet = 1
    END
ELSE
    BEGIN
        SELECT  @Prioritaet = 0
        SELECT  @PrioritaetInfo = ''
    END
IF @FATermin <= @mUrdate OR @FATermin IS NULL BEGIN SELECT @FATermin = @Liefertermin END

IF ISNULL(@ClientID, 0) <> 0 AND ISNULL(@ClientCL, 0) <> 0
    BEGIN
        SELECT  @Sonderbedingungen =
                    CASE ISNULL(@dbAction, 0)
                        WHEN 1 THEN CASE ISNULL(TB_KUNDEN.Auftragsbedingungen, '')
                                        WHEN '' THEN @Sonderbedingungen
                                        ELSE    ISNULL(TB_KUNDEN.Auftragsbedingungen, '')
                                                + CASE ISNULL(@Sonderbedingungen, '')
                                                    WHEN '' THEN ''
                                                    ELSE char(13) + char(10)
                                                         + @Sonderbedingungen
                                                END
                                    END
                        WHEN 2 THEN CASE ISNULL(@Sonderbedingungen, '')
                                        WHEN '' THEN ISNULL(TB_KUNDEN.Auftragsbedingungen, '')
                                        ELSE ISNULL(@Sonderbedingungen, '')
                                    END
                        ELSE ''
                    END
        FROM    TB_KUNDEN
        WHERE   TB_KUNDEN.ObjektID = @ClientID
        AND     TB_KUNDEN.KlassenID = @ClientCL
        AND     TB_KUNDEN.State = 1
    END
IF (ISNULL(@Menge, 0) = 0 OR @Liefertermin IS NULL
    OR ISNULL(@ClientID, 0) = 0
    OR ISNULL(@ClientCL, 0) = 0
    OR LTRIM(RTRIM(ISNULL(@BestellNR, ''))) = '') AND @dbAction <> 3

    BEGIN
        RAISERROR ('Es fehlen notwendige Daten zum Speichern des Auftrages (Kunde, Menge, LT oder Be
        ROLLBACK TRANSACTION
    END
ELSE
```

Abbildung 8-9 Prozedur Save (Teil 2 von 5) TB_AUFTRAG

```
BEGIN
    IF @dbAction = 1 --DB_INSERT
        BEGIN

            /* Prüft, ob Nr schon vorhanden, wenn ja Nr + 1! */
            WHILE ((SELECT COUNT(TB_AUFTRAG.KommissionsNR) FROM TB_AUFTRAG
                    WHERE TB_AUFTRAG.KommissionsNR = @AuftragsNR) > 0 )
                BEGIN
                    SELECT @AuftragsNR = @AuftragsNR + 1
                END

            INSERT INTO TB_AUFTRAG
                        (KommissionsNR, Menge, ClientID, Clientklasse, ChildID,
                        Childklasse, BestellNR, Bestelldatum, IhrZeichen, Positionspreis,
                        Zuschlagpreis, Gesamtpreis, Liefertermin, Korrekturtermin, FATermin,
                        Positionsinfo, Printinfo, Anmerkung, APLInfo, IsPrimaerkommission,
                        PreisPlusPlus, PreisPlusPlusInfo, Listensortierung,
                        PrioritaetInfo, Prioritaet, ZBSkonto, ZBTage, Einheit, CheckPreis,
                        IstFertigung,
                        IstInclusiveChildren, OhneLTAusgabe,
                        Sonderbedingungen, erstelltAm, erstelltVon)
            VALUES      (@AuftragsNR, @Menge, @ClientID, @ClientCL, @ChildID,
                        @ChildCL, @BestellNR, @Bestelldatum, @IhrZeichen, @Positionspreis,
                        @PreisPlus, @Gesamtpreis, @Liefertermin, @Korrekturtermin, @FATermin,
                        @Positionsinfo, @Printinfo, @Anmerkung, @APLInfo, @IsPrimaerkommission,
                        @PreisPlusPlus, @PreisPlusPlusInfo, @Listensortierung,
                        @PrioritaetInfo, @Prioritaet, @ZBSkonto, @ZBTage, @EinheitID, @CheckPreis,
                        @IstFertigung,
                        @IstInclusiveChildren, @OhneLTAusgabe,
                        @Sonderbedingungen, GETDATE(), @BenutzerID)

            SELECT @ObjektID = SCOPE_IDENTITY()
            SELECT @LastIDUpdate = 1

        END
    ELSE IF @dbAction = 2 --DB_UPDATE
        BEGIN

            IF (SELECT  COUNT(REL_AUFTRAG_AUFTRAG.ObjektID)
                FROM    REL_AUFTRAG_AUFTRAG
                WHERE   REL_AUFTRAG_AUFTRAG.ParentID = @ObjektID
                AND     REL_AUFTRAG_AUFTRAG.ParentCL = @KlassenID) > 0

                BEGIN
                    SELECT  @mIsPrimaerkommission = 1
                END
            IF (SELECT  COUNT(REL_AUFTRAG_AUFTRAG.ObjektID)
                FROM    REL_AUFTRAG_AUFTRAG
                WHERE   REL_AUFTRAG_AUFTRAG.ChildID = @ObjektID
                AND     REL_AUFTRAG_AUFTRAG.ChildCL = @KlassenID) > 0

                BEGIN
                    SELECT  @mIsChildkommission = 1
                END
```

Abbildung 8-10 Prozedur Save (Teil 3 von 5) TB_AUFTRAG

```
            SELECT  @mOldIsPrimaerkommission = ISNULL(TB_AUFTRAG.IsPrimaerkommission, 0)
            FROM    TB_AUFTRAG
            WHERE   TB_AUFTRAG.ObjektID = @ObjektID
            AND     TB_AUFTRAG.KlassenID = @KlassenID

            IF (@mIsPrimaerkommission = 1 OR @mIsChildkommission = 1)
                AND @mOldIsPrimaerkommission <> @IsPrimaerkommission
                BEGIN
                    RAISERROR ('Dieser Auftrag ist eine Haupt- oder Childkommission, der
                    ROLLBACK TRANSACTION
                END
            ELSE
                BEGIN

                    UPDATE  TB_AUFTRAG
                    SET     KommissionsNR = @AuftragsNR,
                            Menge = @Menge,
                            ClientID = @ClientID,
                            Clientklasse = @ClientCL,
                            ChildID = @ChildID,
                            Childklasse = @ChildCL,
                            BestellNR = @BestellNR,
                            IhrZeichen = @IhrZeichen,
                            Bestelldatum = @Bestelldatum,
                            Positionspreis = @Positionspreis,
                            Zuschlagpreis = @PreisPlus,
                            Gesamtpreis = @Gesamtpreis,
                            Liefertermin = @Liefertermin,
                            FATermin = @FATermin,
                            Korrekturtermin = @Korrekturtermin,
                            Positionsinfo = @Positionsinfo,
                            Printinfo = @Printinfo,
                            Anmerkung = @Anmerkung,
                            PreisPlusPlus= @PreisPlusPlus,
                            PreisPlusPlusInfo = @PreisPlusPlusInfo,
                            APLInfo = @APLInfo,
                            PrioritaetInfo = @PrioritaetInfo,
                            CheckPreis = @CheckPreis,
                            Prioritaet = @Prioritaet,
                            OhneLTAusgabe = @OhneLTAusgabe,
                            IsPrimaerkommission = @IsPrimaerkommission,
                            ZBSkonto = @ZBSkonto,
                            ZBTage = @ZBTage,
                            Einheit = @EinheitID,
                            IstFertigung = @IstFertigung,
                            Sonderbedingungen = @Sonderbedingungen,
                            IstInclusiveChildren = @IstInclusiveChildren,
                            Listensortierung = @Listensortierung,
                            geaendertAm = GETDATE(),
                            geaendertVon = @BenutzerID
                    WHERE   ObjektID = @ObjektID
                    AND     KlassenID = @KlassenID

                END
        END
```

Abbildung 8-11 Prozedur Save (Teil 4 von 5) TB_AUFTRAG

```
        ELSE IF @dbAction = 3 --DB_DELETE
            BEGIN

                IF (SELECT  COUNT(TB_FERTIGUNGSAUFTRAG.ObjektID)
                    FROM    TB_FERTIGUNGSAUFTRAG
                    WHERE   TB_FERTIGUNGSAUFTRAG.AuftragsID = @ObjektID
                    AND     TB_FERTIGUNGSAUFTRAG.Auftragsklasse = @KlassenID) = 0

                    BEGIN

                        DELETE FROM REL_AUFTRAG_AUFTRAG
                        WHERE       REL_AUFTRAG_AUFTRAG.ParentID = @ObjektID
                        AND         REL_AUFTRAG_AUFTRAG.ParentCL = @KlassenID

                        DELETE FROM REL_AUFTRAG_AUFTRAG
                        WHERE       REL_AUFTRAG_AUFTRAG.ChildID = @ObjektID
                        AND         REL_AUFTRAG_AUFTRAG.ChildCL = @KlassenID

                        DELETE FROM TB_BESTELLUNG
                        WHERE       TB_BESTELLUNG.Bestellt_fuer_ID = @ObjektID
                        AND         TB_BESTELLUNG.Bestellt_fuer_CL = @KlassenID

                        DELETE FROM TB_AUFTRAG
                        WHERE       TB_AUFTRAG.ObjektID = @ObjektID
                        AND         TB_AUFTRAG.KlassenID = @KlassenID
                    END

                ELSE
                    BEGIN
                        RAISERROR ('Zu diesem Auftrag existiert ein Fertigungsauftrag, der Vorga
                        ROLLBACK TRANSACTION
                    END
            END
    END

IF (@LastIDUpdate = 1)
    BEGIN
        EXECUTE setLastID_100 @AuftragsNR, 63, 0
    END
SELECT  @ret = 1
RETURN
```

Abbildung 8-12 Prozedur Save (Teil 5 von 5) TB_AUFTRAG

Attributname	Typ	Größe	Standard	Beschreibung
ObjektID	int	4		Primärschlüssel
KlassenID	smallint	2	63	KlassenID aus der Superklassentabelle
ArchivCL	smallint	2	64	KlassenID der Archivtabelle
KommissionsNR	int	4		Laufende Auftragsnummer, wird automatisch von der Datenbank vergeben
Menge	int	4		Auftragsmenge
Einheit	int	4		1:n Beziehung zur Tabelle der Einheiten, Zugriff über die ObjektID
ClientID	int	4		1:n Beziehung zur Tabellen der

				Klienten, Zugriff über die ObjektID
ClientCL	smallint	2		1:n Beziehung zu Tabellen der Klienten, dient zur Unterscheidung des Klienttyps
ChildID	int	4		1:n Beziehung zu Tabellen des Objekts, welches beauftragt wird
ChildCL	smallint	2		1:n Beziehung zu Tabellen der Objekte, dient zur Unterscheidung des Objekttyps (Artikel, Zukauf, DIN oder Arbeitsplan)
BestellNr	varchar	50		Bestellnummer des Klient
Bestelldatum	datetime	8		Datum des Auftragseingangs
IhrZeichen	varchar	50		Kontaktname der Bestellung
Positionspreis	money	8		Preis der Auftragsposition
Zuschlagpreis	money	8		Preiszuschlag pro Stück
PreisPlusPlus	money	8		Preiszuschlag pro Auftrag
PreisPlusPlusInfo	varchar	200		Informationen zum Preiszuschlag pro Auftrag
Gesamtpreis	money			Auftragsvolumen
Liefertermin	datetime	8		Wunschtermin der Lieferung
Korrekturtermin	datetime	8		Korrigierter Liefertermin
FATermin	datetime	8		Termin für Terminierung
Positionsinfo	nvarchar	500		Auftragspositionstext
Printinfo	nvarchar	500		Informationen zur Druckausgabe
Anmerkung	nvarchar	500		Anmerkungen zum Auftrag
Fertigungsstand	tinyint	1	1	Status der Fertigung
APLInfo	nvarchar	500		Informationen für den Arbeitsplan

PrioritaetInfo	varchar	100		Informationen für angemahnte Aufträge
Prioritaet	bit	1	0	Einstufung der Wichtigkeit
Auftragsstatus	tinyint	1	1	Status des Auftrages
ZBSkonto	float	8	2	Zahlungsbedingungen Skonto des Auftrages
ZBTage	smallint	2	10	Zahlungsbedingungen Tage des Auftrages
Sonderbedingungen	nvarchar	500		Sonderbedingungen zum Auftrag
IstPrimaerkommission	bit	1	1	Definition der Haupt- oder Oberkommission
CheckPreis	bit	1	0	Preisprüfung vor Rechnungslegung
IstFertigung	bit	1	1	Festlegung ob Fertigung oder Warenverkauf
Listensortierung	varchar	20		Spezielle Sortierung für Aufträge
IstInclusiveChildren	bit	1	0	Ausgabe der Auftragsbestätigung mit aufgeführten Childkommissionen
OhneLTAusgabe	bit	1	0	Liefertermin auf Auftragsbestätigung
ErstelltAm	datetime	8		Erstellungsdatum
ErstelltVon	int	4		1:n Beziehung zur Tabelle der Benutzer
GeaendertAm	datetime	8		Änderungsdatum
GeaendertVon	int	4		1:n Beziehung zur Tabelle der Benutzer
State	tinyint	1	1	Objektstatus, gelöscht oder aktiv

Abbildung 8-13 Auftragstabelle

Eigenschaftname	Beschreibung	Typ
IStatus		String
IVersion		String
IBenutzerID		Integer
ICaller	Objekt, welches die Instanz erstellt hat	Object
ICaption		String
IConnection	Datenbankverbindung des SQLClient	Object
IDataOK	ReadOnly-Eigenschaft	Boolean
IGeaendert	Objektänderungsstatus	Boolean
IInterfaceDic	Auflistung benutzerdefinierter Objekte die benötigt und übergeben werden	Object
IMsgTitle		String
IKlassenID		Short
IObjektID		Integer
IVorhanden		Boolean
IRechte	Rechte des aktuellen Benutzers	Object
IControl	Das Control, welches die Komponente nutzt	Object
IRefreshList	Teilt der Hauptanwendung mit ob Listen aktualisiert werden müssen	Boolean
IRechteKey	Rechteschlüssel der Komponente	Short
IWithEditForm	Teilt mit, ob sich das Control auf einer eigenständigen Form befindet	Boolean

Abbildung 8-14 Übersicht der Standardeigenschaften

Methodenname	Typ	Beschreibung
ReadObjektInfo	Prozedur (Public Sub)	Gibt Informationen über den aktuellen Auftrag
ReadGlobalInfo	Prozedur (Public Sub)	Gibt Informationen über andere auftragsabhängige Daten
ReadGesamtpreis	Prozedur (Public Sub)	Errechnet das aktuelle Auftragsvolu-

		men
ReadPositionspreis	Funktion (Public Function)	Holt den aktuellen Artikelpreis aus der Datenbank
GetMyList	Funktion (Public Function)	Ruft eine Funktion auf, die eine Liste instanziiert und diese Instanz übergibt
GetClassRecht	Funktion (Public Function)	Gibt den Rechteschlüssel zurück

Abbildung 8-15 Spezielle Methoden

Eigenschaftname	**Beschreibung**	**Typ**
IIhrZeichen		String
IAnmerkung		String
IAPLInfo		String
IAuftragsNr	Kommissionsnummer	Integer
IBestelldatum		Date
IBestellNr		String
IChildCL	Klasse des Auftragsobjektes (Artikelklasse)	Short
IChildID	Objekt-ID des Auftragsobjektes (Artikel-ID)	Integer
IChildInfo	Beschreibung des Auftragsobjektes	String
IClientCL	Klassen-ID des Kunden	Short
IClientID	Objekt-ID des Kunden	Integer
IGesamtpreis	Die Methode ReadGesamtpreis füllt diese Eigenschaft	Double
IGlobalinfo	Die Methode ReadGlobalInfo füllt diese Eigenschaft	String
IVarianteninfo	Eigenschaft der Variante des Auftragsobjektes	String
IKorrekturtermin	Korrigierter Liefertermin	Date
ILiefertermin		Date
IPositionsinfo		String
IPositionspreis		Double

IPreisPlus	Aufschlag des Positionspreises pro Stück	Double
IPrintInfo	Text für Druckausgabe	String
IMenge		Integer
IObjektinfo	Die Methode ReadObjektInfo füllt diese Eigenschaft	String
IIsPrimaerkommission	Ob Auftrag Hauptkommission darstellt	Boolean
IPrioritaet		Boolean
IPrioritaetInfo		String
IKalkulationspreis	Gibt den Selbstkostenpreis zurück	Double
IZBSkonto	Zahlungsbedingungen Skonto	Double
IZBTage	Zahlungsbedingungen Tage	Short
IEinheit		String
IEinheitID	Einheit für Auftrag (Stück, Paar usw.)	Integer
ISonderbedingungen		String
ICheckPreis	Setzt Flag, ob Preis geprüft werden muss	Boolean
IIstFertigung	Informiert, ob Lohnarbeit vorliegt	Boolean
IFATermin	Datum für Terminierung	Date
IPreisPlusPlus	Aufschlag auf Kommission	Double
IPreisPlusPlusInfo	Aufschlagsinformation	String
IListensortierung	Spezielle Sortierung für Aufträge	String
IIsInclusiveChildren	Auftragsbearbeitung inklusive Unterkommissionen	Boolean
IOhneLTAusgabe	Auftragsbestätigung ohne konkreten Liefertermin	Boolean
IErstelltAm		Datum
IErstelltVon		String
IGeaendertAm		Datum
IGeaendertVon		String

Abbildung 8-16 Spezielle Eigenschaften

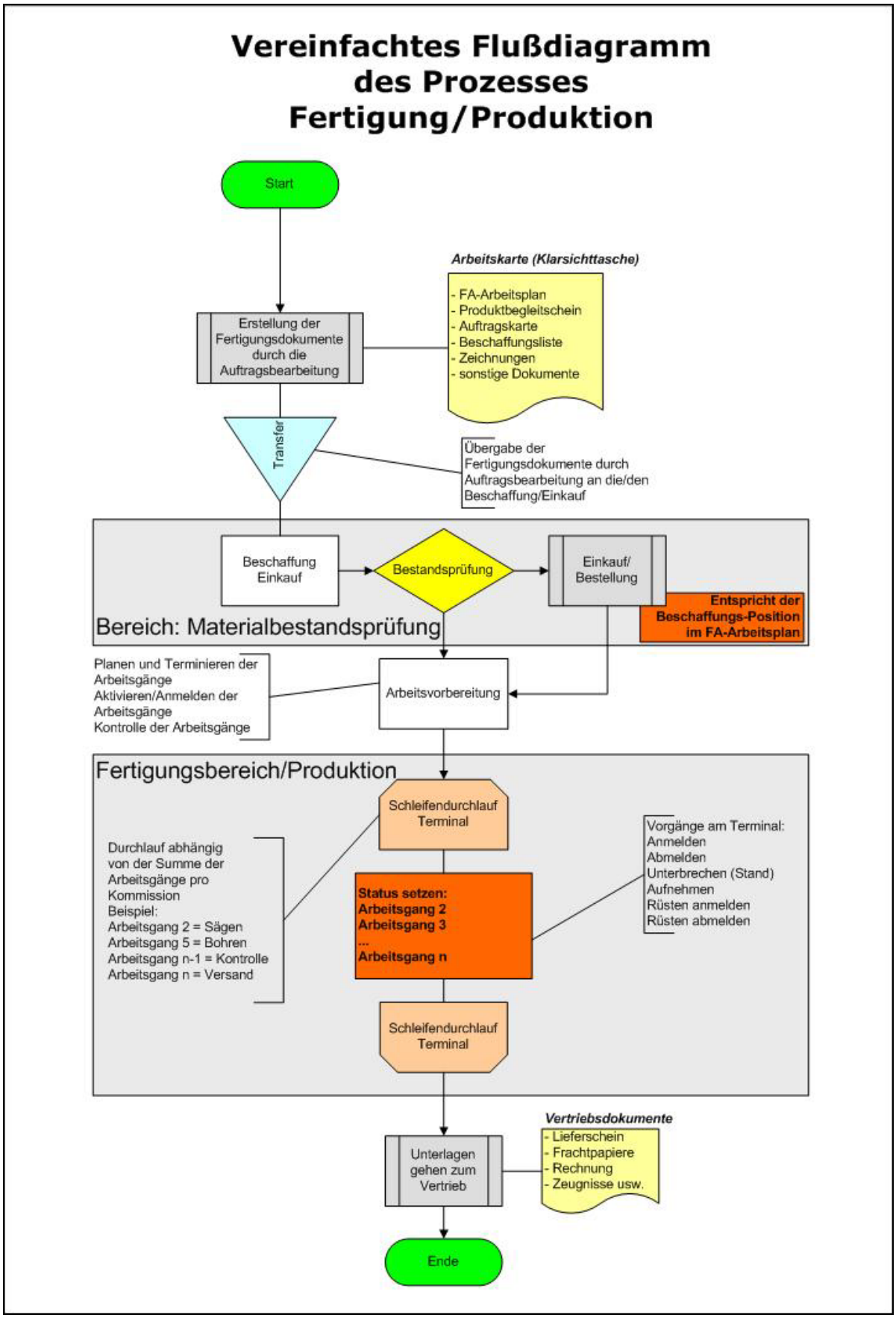

Abbildung 8-17 Flussdiagramm original

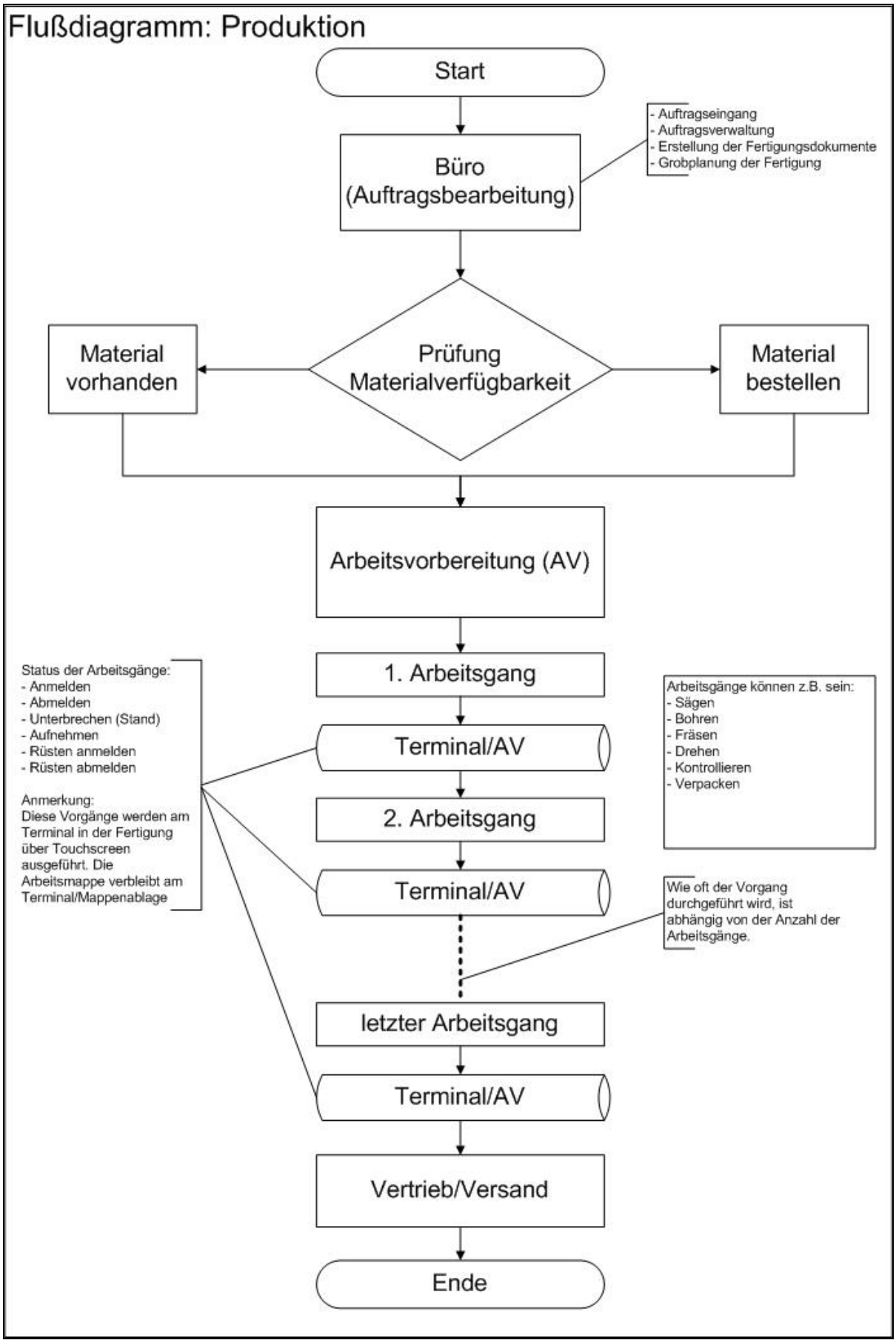

Abbildung 8-18 Flussdiagramm abstrahiert

9 ABBILDUNGSVERZEICHNIS

10 LITERATURVERZEICHNIS

[01] Frielingsdorf, H.: PDM-Marktstudie, Fraunhofer IAO, http://www.rdm.iao.fhg.de/, Jahr 2003

[02] Adelsberger, H.: PPS-Grundlagen und Produktionslogistik, Universität -GH Essen, FB5, Jahr 2002

[03] Stein, D.: Lernzettel ERP (Enterprise Resource Planning), Universität -GH Essen, FB5, Jahr 2002

[04] Scheer: Informationsweek, Institut für Wirtschaftsinformatik, Universität Saarbrücken, Ausgabe 13-2002

[05] Bundesministerium für Bildung und Forschung: IID – Initiative Informationsgesellschaft Deutschland, Förderprogramm für Informationstechnik, http://www.iid.de/it-pro/, Jahr 2002

[06] Rukzio, E.: Vorlesung COM, DCOM Objektmodell und OLE Architektur, Lehrstuhl Multimediatechnik, TU Dresden, Jahr 1999

[07] Microsoft: .NET Framework in der Praxis, MSDN Developer Days 2001

[08] Microsoft: .NET Framework Grundlagen, http://www.microsoft.de/germany/msdn,

[09] Microsoft: .NET Framework as Solution Platform, MSDN Roadshow, Jahr 2001

[10] Microsoft: Übersicht über das .NET Framework, .NET Entwicklerhandbuch, Online-Dokumentation Version 2003, Jahr 2003

[11] Landgrave, T.: Systemschutz mit Code Access Security, CNET Networks, Inc., http://www.zdnet.de/builder/architect/, Jahr 2003

[12] Prosise, J.: .NET Entwicklerhandbuch, Microsoft Press, Kap. I, S. 15-17, Jahr 2002

[13] Doberenz, W. und Kowalski, T.: Datenbankprogrammierung, Microsoft Press 1999, Kap. 8, S. 287ff

[14] Box, D.: Essential XML, Addison Wesley, Programmer's Choice, Jahr 2000

[15] Microsoft: Übersicht über ADO.Net, http://de.gotdotnet.com, Jahr 2003

[16] Ulm, T.: Remoting in Microsoft .NET, http://www.devtrain.de/, Jahr 2003

[17] Eckstein, H.: Informationssysteme in der Produktentwicklung, Rapid Prototyping, Vorlesung WS 2002/03, Competence Center FuE-Management, www.rdm.iao.fhg.de, Fraunhofer-Institut für Arbeitswirtschaft und Organisation IAO in Stuttgart

[18] Schwichtenberg, H.: Einführung in das Microsoft .NET Framework, IT-Vision, Jahr 2002

[19] Zicari, R.: Objektorientierte Datenbanksysteme, Vorlesung Sommersemester, Universität Frankfurt, Jahr 2002

[20] Kraneis, J.: IFC-Seminar C#, Universität Osnabrück, Jahr 2001

[21] Bergers, D.: Vorlesung Management technischer Projekte und Geschäftsprozesse, Produktionstechnologie und Produktentwicklung, Institut für Prozess- und Datenmanagement, Universität Essen, Jahr 2002

[22] Bergers, D.: Vorlesung Produktionstechnik, Institut für Prozess- und Datenmanagement, Universität Essen, Jahr 2003

[23] Seibert, S.: Bewertungsfragen für Projekte, Fachhochschule Darmstadt, Ursprung: Survival Guide Website – http://www.construx.com/survivalguide/, Jahr 2002

[24] Lelic, S.: UI-Design, MS-TechTalk (Universität Essen), Jahr 2002

[25] Borchert, C.: Vorlesung Datenbanken und lokale Netze, IKB Universität Essen, Jahr 1996

[26] Microsoft: Übersicht zu den Komponenten von SQL Server 2000, Dokumentation MS SQL-Server 2000, Jahr 2000

[27] Microsoft: SQL Server-Architektur, Dokumentation MS SQL-Server 2000, Jahr 2000