eXBO-Framework für eCommerce Anwendungen

vorgelegt von
Dipl.-Ing. Oliver Fox
geboren am 27.08.1969 in Berlin

Vom Fachbereich IV Elektrotechnik und Informatik
der Technischen Universität Berlin

zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
-Dr.-Ing-

genehmigte Dissertation

Vorsitzender:	Prof. Dr. Krallmann
Berichterstatter:	Prof. Dr. Dr. h.c. Radu Popescu-Zeletin
	Prof. Dr. Bernd Mahr
Tag der wissenschaftlichen Aussprache:	7.Mai 2003

Berlin 2003
D83

Danksagung

Diese Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter im Forschungsinstitut für offene Kommunikationssysteme (Fokus) des Forschungszentrums für Informationstechnik GmbH (GMD).

Zu allererst gilt mein besonderer Dank meinem langjährigen Mentor und Betreuer Gerd Schürmann, der es mir durch die Schaffung entsprechender Freiräume von Anfang an ermöglichte, meine Ideen und Konzepte zu entwerfen und umzusetzen. Von besonderer Bedeutung waren – und sind – seine wegweisenden Anregungen und die Motivationsschübe, die sich beinahe aus jedem Gespräch ergaben.

Mein besonderer Dank geht auch an meinen Doktorvater, Herrn Prof. Dr. Dr. h. c. Radu Popescu-Zeletin, dem Leiter des Forschungsinstituts für Offene Kommunikationssysteme, für die Übertragung der Forschungsaufgabe. Seine kritischen Anregungen und die Diskussionen mit ihm, haben entscheidend zum Gelingen dieser Arbeit beigetragen.

Ein weiterer besonderer Dank geht an Herrn Prof. Dr. Manuel Mendes. Durch die mit ihm geleisteten Arbeiten und durch seine Impulse hat er sehr zur Fertigstellung dieser Arbeit beigetragen.

Bei Herrn Prof. Dr. Bernd Mahr bedanke ich mich für die freundliche Übernahme des Koreferats.

Meinen Kollegen Ali Hafezi, Marcin Solarski und Robert Fischer danke ich für die fruchtbare Zusammenarbeit und für die Hilfsbereitschaft bei allen angefallenen Problemen.
Natürlich gilt mein Dank auch den weiteren Mitarbeitern des Forschungsinstituts für Offene Kommunikationssysteme, die mir durch die fachliche Diskussion stets zur Seite standen und die durch ein freundliches Arbeitsklima ebenfalls zum Gelingen dieser Arbeit beigetragen haben.

In diesem Zusammenhang möchte ich noch ein besonderes Lob für Herrn Dr. Eckhard Moeller, dem Leiter des Competence Center for Distributed Object Technology, Platforms and Services (PLATIN) aussprechen, dessen Wirken ein entscheidender Faktor für die Entfaltung eines fördernden und freundlichen Arbeitsklimas ist.

Weiterhin möchte ich mich bei den Gründern und Mitarbeitern der Firma MetaObject bedanken, die entscheidenden Anteil daran haben, dass die theoretischen Konzepte umgesetzt

und in der Praxis erprobt werden konnten.

Bei Herrn Herwart Schmidt-Joos bedanke ich mich darüber hinaus, für die zahlreichen fruchtbaren Diskussionen die meine theoretischen Erkenntnisse untermauerten.

Ein ganz besonderer Dank geht an Herrn Daniel Bleisteiner, zweifelsohne einer der begabtesten Programmierer dieser Zeit, der sich stets selbstlos bemühte, meinen Wissensdrang zu stillen und mir, wo es nur ging, die Gelegenheit gab, von seinem reichhaltigen Erfahrungsschatz zu profitieren. Letztlich war er es, der mich in die Lage versetzte, große komplexe Anwendungen zu entwerfen und umzusetzen. Ohne seine Mitwirkung wäre diese Arbeit nicht zustande gekommen.

Bei Herrn Frank Winkler möchte ich mich für die Gespräche und kritischen Reflexionen bedanken, die entscheidend dazu beitrugen, die wesentlichen Punkte der Arbeit zu identifizieren und deutlicher herauszustellen.

Bedanken möchte ich mich auch bei all denen, die indirekten Anteil am Entstehen dieser Arbeit haben.

Bei Herrn Dipl.-Ing. Karl-Heinz Stapel, der mir als erster zeigte, wie man durch zielstrebiges wissenschaftliches Arbeiten gesteckte Ziele erreichen kann. Die damalige, gemeinsame, zielstrebige Arbeit beeinflusst meinen Lebensweg bis zum heutigen Tag.

Bei meiner Freundin möchte ich mich für ihre Geduld, ihre Liebe und ihre Unterstützung bedanken.

Last but not least bedanke ich mich bei allen meinen „Kumpels“. Besonders bei Dipl.-Ing. Udo Elzholz, der mich so erfolgreich durch das Grundstudium „geboxt“ hat. Bei Dipl.-Ing. Mustafa Demirtas und Dr. Ing. Ayman Ghazi bedanke ich mich für alles was sie für mich getan haben und was ich mit ihnen erleben durfte.

Für meine Familie

Oliver Fox

Zusammenfassung

In dieser Arbeit wird ein objektorientertes Framework für die Entwicklung von eCommerce-Systemen vorgestellt.

Mit diesem Framework werden Unternehmen in die Lage versetzt, deutlich schneller als bisher und deutlich kostengünstiger als bisher, elektronische Lösungen für ihre Geschäftsmodelle zu erstellen.

Den Betrachtungsschwerpunkt bilden dabei Geschäftsobjekte, die ihre Entsprechung in der real existierenden Geschäftswelt haben. Diese Betrachtungsweise ermöglicht eine Annäherung zwischen der betriebswirtschaftlich-anwendungsbezogenen Sichtweise von Auftraggebern und Managern und der technisch-implementierungsbezogenen Sicht von Entwicklern. Dadurch wird die für eine schnelle und korrekte Umsetzung von Geschäftsmodellen in einsatzfähige Softwaresysteme notwendige, enge Zusammenarbeit von Software-Ingenieuren und Experten aus den jeweiligen Geschäftsbereichen wesentlich unterstützt.

In dieser Arbeit werden zahlreiche Business-Objects (Geschäftsobjekte) präsentiert, die das Grundgerüst der meisten Geschäftsanwendungen bilden können.

Die grundlegende Idee der Business-Objects (BO) wird dabei erweitert, und es werden die sogenannten eXtended Business-Objects (eXBO) eingeführt. Diese erweitern die Konzeption herkömmlicher Business-Objects in zwei wesentlichen Punkten:

- eXBO besitzen zusätzliches „Wissen“ über ihre Darstellung. Dies ermöglicht den Aufbau von eCommerce-Systemen die gleichzeitig für verschiedene Endgeräte spezialisiert sind.

- eXBO besitzen zusätzliches „Wissen“ über ihre Bearbeitung. Dies ermöglicht es ihnen mit denen in dieser Arbeit vorgestellten eXBO-Facility-Komponenten bearbeitet zu werden. Die Kombination von eXBO und eXBO-Facilities ermöglicht bisher unerreichte Entwicklungsgeschwindigkeiten bei der Konstruktion von eCommerce-Anwendungen.

Neben den eXBO und eXBO-Facilities werden weitere, wiederverwendbare Objekte und Komponenten vorgestellt, die ebenfalls dazu beitragen, daß zukünftige eCommerce-Anwendungen schneller und effizienter aufgebaut werden können.

Desweiteren werden in den einleitenden Kapiteln allgemeine Architekturen und Entwurfskonzepte aufgezeigt, die gewählt werden sollten, um ein Maximum an Erweiterbarkeit, Flexibilität und Integrationsfähigkeit in vorhandene und zukünftige IT-Strukturen zu gewährleisten.

In den letzten Kapiteln dieser Arbeit wird ein Teil der teilweise preisgekrönten eCommerce-Systeme vorgestellt, die mit dem vorgestellten Framework bereits erfolgreich umgesetzt werden konnten.

Inhaltsverzeichnis

Kapitel 1

Einleitung und Überblick

Im Jahre 1990 entwickelte eine Gruppe von Wissenschaftlern um Tim Bernes Lee am CERN (CERN - European Organization for Nuclear Research) die ersten Web-Server und Web-Browser. Diese Technologie ermöglichte es, auf bis dahin unbekannte, genial einfache Weise, Daten, Ergebnisse und Erfahrungen vollkommen unkompliziert, unabhängig von Entfernungen und verwendeter Computer-Hardware oder verwendetem Computer-Betriebssystem auszutauschen.

Sehr schnell erkannte die Wirtschaft das enorme Potential direkt und kostengünstig eine riesige und ständig wachsende Zahl potentieller Kunden zu erreichen. Zunächst begnügte man sich mit der Bereitstellung von statischen Informationen. Daraus entwickelte sich rasch der Wunsch, die dargebotenen Waren auch unmittelbar über das Internet verkaufen zu können. Aus diesem Bedarf entstanden die ersten Internet-Shop-Systeme, die – im nachhinein betrachtet – nur den Anfangspunkt einer weltweiten eCommerce-Euphorie markierten.

Längst existieren neben den Shop-Systemen zahlreiche weitere Geschäftsmodelle für die Abwicklung von Geschäftsprozessen im Internet und beinahe täglich kommen neue hinzu. Trotz teilweiser immenser Kosten, die bei der Umsetzung von vorhandenen Geschäftsmodellen in elektronische Lösungen entstehen, werden meist Eigenentwicklungen bevorzugt, da die wenigen existierenden Standard-Lösungen nur selten auf die vorhandenen oder geplanten Geschäftsmodelle anwendbar sind. Zur Zeit existiert ein gewaltiger Bedarf für

die Entwicklung von eCommerce-Systemen, die eine elektronische Umsetzung von Geschäftsmodellen darstellen. Dieser Bedarf, so die einheitliche Aussage fast aller Prognosen soll in absehbarer Zukunft zudem noch weiter ansteigen [1-4], [8-33].

Ein wesentlicher Faktor dafür ist der Umstand, daß die Unternehmen im Begriff sind zu erkennen, wie sie unter Nutzung der Informationstechnologie und des Intra- und Internets, nicht nur neue Kunden und neue Absatzmärkte erschließen können, sondern, daß sie ebenfalls enorme Einsparungspotentiale erreichen können, indem sie die vorhandenen Geschäftsprozesse durch eCommerce-Systeme optimieren und rationalisieren.

Einige Studien gehen noch darüber hinaus. Sie proklammieren, daß der erfolgreiche Einsatz von eCommerce-Systemen nicht nur zu entscheidenden Wettbewerbsvorteilen führt, sondern in naher Zukunft zu einem überlebensnotwendigen Faktor für die meisten Unternehmen wird.

In der Praxis existieren zur Zeit aber noch verschiedenste technologische, gesellschaftliche und wirtschaftliche Eintrittsbarrieren, die vor allem die kleinen und mittelständigen Unternehmen bisher daran hindern, die vorhandenen Potentiale des eCommerce zu nutzen.

Zu den wesentlichen Einstiegshürden gehören zu hohe Entwicklungskosten und zu lange Einführungs- und Entwicklungszeiträume. Dies führt beispielsweise dazu, daß die Amortisation von Investitionen in Rationalisierungsmaßnahmen durch eCommerce-Lösungen in weite Ferne gerückt wird.

Das in dieser Arbeit vorgestellte eXBO-Framework (eXBO-Framework - eXtended Business-Objects-Framework) versetzt Unternehmen in die Lage, deutlich schneller als bisher und deutlich kostengünstiger als bisher, elektronische Lösungen für die Optimierung und Rationalisierung verschiedenster Geschäftsmodelle zu erstellen.

1.1 ECommerce-Systeme

In diesem Abschnitt wird zunächst auf die Problematik der Definition des Begriffs eCommerce eingegangen. Um einen besseren Überblick geben zu können, wird anschließend eine Kategorisierung der verschiedenen Bereiche des eCommerce vorgenommen.

Wie viele neue Begriffe kommt auch „eCommerce“, aus dem Englischen, übersetzt man den Begriff wörtlich, bedeutet eCommerce „elektronischer Handel“ oder „elektronisches Geschäft“. In Ermangelung einer eindeutigen Begriffsdefinition versuchen verschiedenste Autoren eigene Definitionen zu finden. Einige von ihnen sind:

- „eCommerce ist die elektronisch realisierte Anbahnung, Aushandlung und Abwicklung von Geschäftsprozessen zwischen Wirtschaftssubjekten.“
- „Zu eCommerce gehört jede Art von geschäftlichen Transaktionen bei denen die Beteiligten auf elektronischem Weg miteinander kommunizieren und nicht in direktem physischen Kontakt stehen.“
- „eCommerce-Systeme stellen elektronische Umsetzungen von Geschäftsmodellen dar.“

Andere Definitionen schränken den Bereich des eCommerce stärker ein und beziehen das Präfix „e“ auf die Verwendung von Internet-Intranet-Technologie:

- „Unter eCommerce kann allgemein die Abwicklung von Geschäftsprozessen über das Internet-Intranet verstanden werden.“
- „eCommerce ist der elektronische Handel von Gütern und Dienstleistungen über das Internet.“

Am häufigsten jedoch (vor allem von Laien) wird unter dem Begriff eCommerce der Verkauf von Produkten im Internet, insbesondere im World Wide Web verstanden. Der Verkauf von Produkten im World Wide Web ist aber nur ein kleiner Teil des Spektrums von eCommerce. Will man eCommerce umfassend für Unternehmen wirklich vorteilhaft einsetzen, ist es notwendig, einen Blickwinkel anzuwenden, der nicht ausschließlich verkaufsorientiert ist sondern eher beziehungsorientiert. Am besten nähert man sich der Thematik wenn man eine Wirtschaft als Geflecht von vielfältigen Beziehungen (Wirtschaftsbeziehungen) zwischen den einzelnen Akteuren in der Wirtschaft versteht. Vielversprechende Definitionsansätze wären

demnach:

- „Elektrifizierung von Wirtschaftsbeziehungen.“
- „Elektronische Umsetzung von Geschäftsmodellen bzw. Geschäftsprozessen.“

Die Grundlegenden Einheiten der Wirtschaftsordnung sind Unternehmen, öffentliche Verwaltungen und der Bürger. Unternehmen produzieren in arbeitsteiligen Prozessen Güter und Dienstleistungen. Dabei nehmen die Unternehmen Güter und Dienstleistungen anderer Unternehmen (z.B. von Zulieferern in Anspruch). Die Ergebnisse ihrer Arbeit verkaufen Unternehmen an ihre Kunden, den Konsumenten. Der Konsument kann dabei ein Bürger, ein anderes Unternehmen oder auch der öffentliche Sektor sein.

Neben diesen grundsätzlichen Lieferanten- und Kundenbeziehungen existieren außerdem Beziehungen von Unternehmen und Bürgern zum Staat. In diesen Beziehungen werden durch einen gesetzlichen Ordnungsrahmen die Rechte und Pflichten von Unternehmen und Bürgern in einer Marktwirtschaft geregelt. Dieses Beziehungsgeflecht in einer Gesellschaft wird nachfolgend vereinfacht dargestellt. Es gibt sechs wesentliche Beziehungen:

deutsche Bezeichnung	englische Bezeichnung	Abkürzung
Unternehmen zu Unternehmen	Business to Business	B2B
Unternehmen zu Bürger	Business to Citizen	B2C
Staat zu Unternehmen	Government to Business	G2B
Bürger zu Bürger	Citizen to Citizen	C2C
Staat zu Bürger	Government to Citizen	G2C
Staat zu Staat	Government to Government	G2G

Abbildung 1: Tabellarische Darstellung der sechs wesentlichen Wirtschaftsbeziehungen.

Diese sechs wesentlichen Beziehungen werden im allgemeinen den beiden Oberkategorien eCommerce und eGovernment zugeordnet (vgl. Abb. 2).

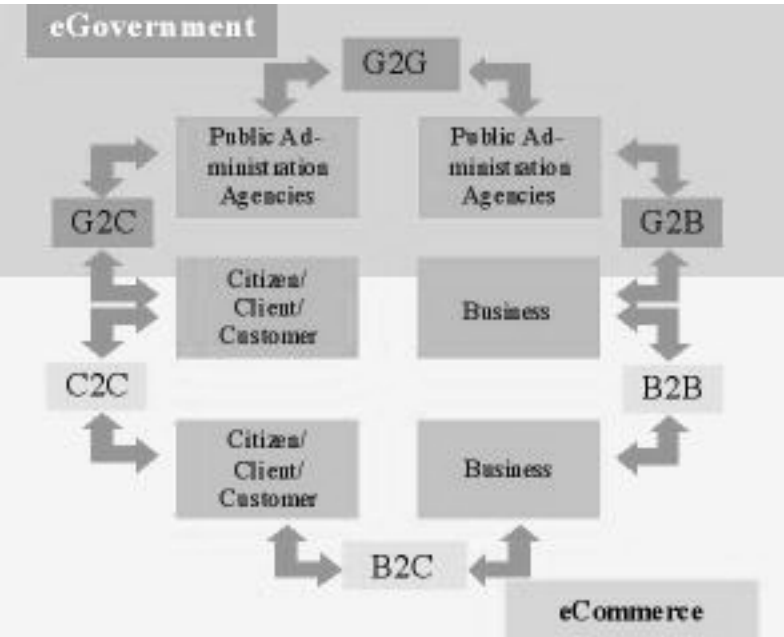


Abbildung 2: Kategorien von eCommerce und eGovernment.

Innerhalb dieser Kategorien können wiederum verschiedenste Geschäftsmodelle bzw. Ausprägungen identifiziert werden, die ständig durch neue Geschäftsmodelle bzw. Geschäftsideen erweitert werden. Nachfolgend soll kurz auf den aktuellen Entwicklungsstand der verschiedenen Bereiche eingegangen werden.

1.1.1 Electronic Commerce

Der Bereich „Electronic Commerce“ bzw. wird dabei in die folgenden Kategorien unterteilt:

- Consumer to Consumer (C2C),
- Business to Consumer (B2C) und
- Business to Business (B2B).

1.1.1.1 Consumer to Consumer (C2C)

Beispielhafte Vertreter dieser Kategorie von Electronic Commerce sind: Gebrauchtwarenmarktplätze wie z.B. ZweiteHand oder AutoScout24 oder auch Versteigerungen wie z.B. eBay, Ricardo oder QXL. Mischformen von C2C und B2C stellen z.B. Ratgeberplätze dar, bei denen einerseits Produkte von Händlern angeboten werden, aber die Meinungsbildung über die öffentliche Diskussion von Verbrauchern zum Kauftipp führt. Weiterführende Literatur zu verschiedenen Ausprägungen des C2C findet man unter anderem in [1] und [2].

1.1.1.2 Business to Consumer (B2C)

Der Business to Consumer-Bereich wird von eCommerce-Shop-System dominiert. Zu den bekanntesten (und erfolgreichsten) Vertretern gehören z.B. *www.dell.com* und *www.apple.com,* die jeweils mehrere Millionen Dollar Umsatz pro Monat erzielen.

Neben den normalen eCommerce-Shop-Systemen gibt es weitere Ausprägungen des Business to Consumer-Bereichs. Zu ihnen gehören beispielsweise Marktplätze bei denen mehrere Shops in einem Portal zusammengefügt werden. Durch erweiterte Dienste, z.B. einer übergreifenden Suche können dadurch Vorteile bzw. Mehrwerte für den Kunden entstehen. Eine weitere Anwendung im Bereich des B2C stellt das sogenannte „Power-Shopping“ (Gruppenkauf) dar. Bei dieser Variante sammeln sich Kaufinteressierte für ein bestimmtes Produkt und bestellen später – ähnlich wie ein Großhändler – größere Mengen mit entsprechendem Mengenrabatt. Weitere Literatur zu den verschiedenen Geschäftsmodellen im Business to Consumer Bereich findet man in [1-5].

1.1.1.3 Business to Business (B2B)

Innerhalb des Bereichs Business to Business haben sich bis jetzt die Rollen Buyer, Seller und Marketmaker herauskristallisiert. Zu den wichtigsten Anwendungen im Bereich Business to Business gehören Buyer-Managed-, Seller-Managed- und Marketmaker-Managed-Marktplätze. Neben diesen drei etablierten Anwendungsfällen, die nachfolgend noch einmal näher beschrieben werden, existieren zahlreiche andere Geschäftsmodelle und verschiedenste Entwicklungsperspektiven. In diesem Zusammenhang sei besonders auf [2-4], verwiesen.

a) Buyer-Managed-Marktplätze

Beispiele für Buyer-Managed-Marktplätze sind die Handelsplattformen der US-Auto-Riesen Ford und General Motors. Über diese Plattformen sollen die Geschäftsbeziehungen mit den

jeweils bis zu 30.000 verschiedenen Geschäftspartnern abgewickelt werden. Ziel ist es, das gesamte Zulieferergeschäft „besser, schneller und effizienter“ [12] zu gestalten. Bei Auto-Riesen wie z.B. Ford deren Zulieferergeschäft auf ca. 300 Milliarden Dollar geschätzt wird, bedeutet die erhoffte Senkung der durchschnittlichen Transaktionskosten von 100 Dollar auf 10 Dollar eine Einsparung von bis zu 16 Milliarden Dollar pro Jahr.

Andere Beweggründe für große Einkäufer, Buyer-Managed-Marktplätze zu etablieren bestehen in der Möglichkeit, erhöhte Transparenz über alle Vorgänge bei den Zulieferern zu bekommen. Dies kann z.B. dafür genutzt werden, die Lieferzeiten weiter zu verkürzen und erhebliche Kostensenkungspotentiale im Bereich der Lagerhaltung auszunutzen.

Industriezweige wie z.B. die Automobilindustrie, die durch wenige aber sehr große Hersteller und viele, aber kleinere Zulieferer geprägt sind, sind besonders für Buyer-Managed-Markplätze prädestiniert. Weitere Literatur zum Thema Buyer-Managed-Marktplätze findet man in: [1-2].

b) Supplier-Managed-Marktplätze

Industrien, die für Supplier-Managed-Marktplätze prädestiniert sind, weisen wenige große Hersteller und viele kleinere Abnehmer auf. Beispiele für Supplier-Managed-Marktplätze finden sich z.B. in der Stahlindustrie, oder in der Medizintechnik. Wenngleich zwischen B2B- und B2C-Anwendungen erhebliche Unterschiede bestehen (besonders in Umfang und Aufwand der getätigten Transaktionen, vgl. [1] und [2]), weisen einfache Supplier-Managed-Marktplätze ähnliche Funktionalitäten wie B2C-Shop-Systeme auf. Weitere Literatur zum Thema findet man in [1-5].

c) Marketmaker-Marktplätze

Marketmaker-Marktplätze sind Marktplätze, die weder von „Supplier“, noch von „Buyer“ beherrscht werden. Sie verstehen sich vielmehr als „... intersection of demand and supply chain.“ [2], der Schnittstelle also zwischen verschiedensten Anbietern und Herstellern. Zur Verfügung gestellt werden sie von sogenannten „Marktplatzbetreibern“ (Marketmaker), die auf verschiedenste Weisen (z.B. durch prozentuale Beteiligung an Transaktionen, monatlichen Pauschalbeträgen, Werbung, Vermittlungsprovision, etc.) profitieren können. Weitere Literatur zum Thema findet man in: [1-3].

1.1.2 eGovernment

Der gesamte Bereich des eGovernment ist noch weniger erschlossen als der Bereich des eCommerce. Es gibt so gut wie keine Standardanwendungen, oder gar etablierte Rollen. Einflussfaktoren sind regionale und nationale Unterschiede in den Verwaltungsstrukturen, und erhebliche Sicherheitsbedenken. Das sehr frühe – erst beginnende – Stadium des eGovernment wird in die Kategorien

- Government to Business,
- Government to Government und
- Government to Consumer

unterteilt, auf die nachfolgend kurz eingegangen werden soll.

1.1.2.1 Government to Business (G2B)

Diese Kategorie beinhaltet alle Formen von Transaktionen zwischen Unternehmen und öffentlichen Verwaltungen. Diese können sowohl von geschäftlicher Natur sein (wie z.B. bei öffentlichen Ausschreibungen für Bauaufträge), oder zur Erfüllung des gesetzlichen Ordnungsrahmens dienen (z.B. Gewerbeanmeldungen oder Steuerbescheide). Weiterführende Literatur zum Thema Government to Business und insbesondere den möglichen Überschneidungen von Business to Business und Government to Business findet man in: [1-5].

1.1.2.2 Government to Government (G2G)

Diese Form in der Beziehung beinhaltet den elektronischen Austausch von Informationen zwischen verschiedenen öffentlichen Verwaltungen und die Vereinheitlichung und Zusammenführung von Datenbeständen. Der gegenwärtige Stand ist durch verschiedenste, teilweise komplexe Verwaltungsstrukturen und Verwaltungsvorschriften und durch größtenteils inkompatible, historisch gewachsene Datenverarbeitungssysteme geprägt. Sollen die Bereiche Government to Business und Government to Consumer durch elektronische Verfahren schneller und effizienter gestaltet werden, so muß auch die innerbehördliche Kommunikation den gestiegenen Leistungsanforderungen angepaßt werden. Bestrebungen wie die der Europäisierung erfordern darüber hinaus eine effiziente und geregelte Kommunikation über Landes- und Regierungsgrenzen hinweg. Weiterführende Literatur

findet man in: [1-5].

Zwischenzeitlich sind einige Projekte, die die Problematik der innerbehördlichen Kommunikation angehen wollen, gestartet. Informationen finden sich z.B. in folgenden Projektanträgen [174-175].

1.1.2.3 Government to Citizen (G2C)

Diese Kategorie des eGovernment beinhaltet alle Formen der Interaktionen zwischen Bürgern und öffentlichen Verwaltungen. Der heutige, zögerliche Beginn der Kategorie Government to Citizen ist durch statische Informationsbereitstellungen im World Wide Web gekennzeichnet. Alle aktuellen Projekte, wie z.B. VeZuDa [176], die sich mit der Vereinheitlichung und Zusammenführung von Datenbeständen beschäftigen, ermöglichen in einem ersten Schritt die Schaffung von „Bürgerbüros“, die es ermöglichen, verschiedene öffentliche Verwaltungsvorgänge von einem physikalischen Ort aus durchzuführen. Zukünftige Anwendungsfälle sind Bürgerportale im World Wide Web, in denen Bürger selbständig, z.B. durch die Unterstützung von Lebenslagen (Heirat, Umzug, etc.) Verwaltungsvorgänge initiieren und ausführen können. Konkrete Projekte sind z.B. „eGOV“, „VeZuDa“ oder „ProBüd“. Weitere Literatur zu diesem Thema findet sich in: [175-176].

Wie auch bei der Kategorie eCommerce sind nicht alle Modelle sauber in die drei Bereiche G2B, G2G und G2C einzuordnen. Aktuelle Ansätze, wie z.B. die von „Virtuellen Kommunen“ in denen Städte oder ganze Regionen sich selbst, ihre Unternehmen, ihre Geschäfte, ihre Bürger und die öffentlichen Institutionen präsentieren, können Elemente aus allen eGovernment- und eCommerce-Kategorien beinhalten.

1.1.3 Weitere Ausprägungen des eCommerce

Bevor nachfolgend auf die grundlegenden Anforderungen und die verschiedenen Transaktionsphasen von eCommerce eingegangen wird, muß an dieser Stelle noch einmal betont werden, daß die obige Auflistung nur einige der wichtigsten und allgemein anerkannten Kategorien und Ausprägungen von eCommerce aufzeigt und keineswegs den Anspruch auf Vollständigkeit hat. Der Bereich des eCommerce weist viele weitere interessante Bereiche auf, die bisher noch nicht einmal erwähnt werden konnten. Dazu gehören z.B. eFinance (die elektronische Abwicklung von Zahlungsverkehr, z.B. mit Banken und Kreditinstituten) und

die innerbetriebliche Optimierung und Rationalisierung von Geschäftsmodellen durch elektronische Lösungen, um nur zwei weitere wesentliche Bereiche zu erwähnen. Die weitere ausführliche Betrachtung aller Ausprägungen des eCommerce würden den Rahmen dieser Arbeit sprengen. Weitere Literatur zu den Themen eCommerce, eGovernment, eFinance, Rationalisierung und Optimierung durch eCommerce, eBusiness B2C, B2B, B2C, G2G, G2B, G2C usw. findet man in [1-33].

1.1.4 Anwendungsfunktionen

Bei genauerer Betrachtung von eCommerce-Lösungen können einzelne Anwendungsfunktionen identifiziert werden, die zusammen die Funktionalität eines gesamten Systems widerspiegeln. Viele dieser Anwendungsfunktionen, wie z.B. Content-Management, (das Eingeben und Pflegen von Daten) oder Ranking-Services (Beurteilungen über Produkte und Geschäftsgebaren von Handelspartnern) sind in vielen Anwendungen quer durch die Geschäftsmodelle wiederzufinden. Die einzelnen Anwendungsfunktionen können laut [4] vier verschiedenen Phasen der Geschäftsprozeßunterstützung zugeordnet werden:

- Informationsphase
- Vereinbarungsphase
- Abwicklungsphase
- After-Sales-Phase

Auf die verschiedenen Phasen und die ihnen zugeordneten Anwendungsfunktionen wird nachfolgend kurz eingegangen.

1.1.4.1 Informationsphase

In der Informationsphase geht es für einen potentiellen Käufer darum sich die benötigten Informationen über Anbieter und Produkte zu beschaffen. Je nach Anwendungsfall ist der Informationsbedarf unterschiedlich groß. Bei normalen B2C-Anwendungen genügen meist wenige Produktdaten, ein Bild und ein Preis. Die Informationen die ein „Buyer“ (Einkäufer) auf einem B2B-Marktplatz benötig,t können hingegen weitaus detaillierter sein. Zu den benötigten Informationen gehören z.B. Preisstaffeln für unterschiedliche Mengen, Garantiezusagen für Lieferzeiten und Qualitätsanforderungen oder gar der aktuelle

Lagerbestand beim Hersteller.

Nachfolgend werden einige Anwendungsfunktionen aufgeführt, die Bestandteil der Informationsphase sein können:

i) Content-Management

Das Eingeben und Pflegen der Datenbestände und die entsprechende Darstellung inklusive Benutzernavigation und Suchfunktionen ist der entscheidende Kern fast jeder eCommerce-Anwendung.

ii) Upload-Facilities

Upload-Facilities werden teilweise auch dem Content-Management zugeordnet. Man versteht darunter die Möglichkeit, größere Datenmengen mit einem Mal über das Netz zum Server zu übertragen, wo sie dann automatisch eingepflegt werden. In der Praxis dominieren dabei heute noch „halbautomatische“ ASCII-Exporte und -Importe aus Datenbanksystemen. In Zukunft werden modernere Konzepte, wie die Kommunikation mit verteilten Objekten und vor allem dem Datenaustausch mit XML größere Bedeutung zukommen.

iii) Virtuelle Assistenten

Für den Vertrieb von Produkten hoher Komplexität, für welche es z.B. eine Vielzahl anwendungsspezifischer Varianten gibt (z.B. eine ISDN-Anlage für mittelständige Unternehmen) wäre die Bereitstellung von virtuellen Assistenten bzw. virtuellen Einkaufshilfen von großem Nutzen. Sie können den Käufer beraten und eine sinnvolle Zusammenstellung von Einzeloptionen garantieren.

iv) Werbung

Vor allem für Anwendungen mit großen Nutzerzahlen ist das Einbringen bzw. Vermarkten von Werbung von Interesse. Das gleiche gilt für „kleinere“ Sites, die einen ausgewählten Kundenkreis betreuen. In der einfachsten Form der Werbeunterstützung ist das Vermieten von dafür vorgesehenem Platz auf entsprechenden Werbebannern möglich. Intelligentere Lösungen verbinden das Einblenden von Werbung mit dem Content-Management und ermöglichen ein gezieltes, personalisiertes und situationsbedingtes Einblenden von Werbung.

v) Multimedia-Informationen

Multimediale Informationen sind eine große Bereicherung für alle Phasen der Geschäftsprozeßunterstützung. Dies gilt besonders für die Informationsphase. Trotzdem unterstützen nur die wenigsten Anwendungen einen ausgiebigen Gebrauch von multimedialen Informationen. Nur bei wenigen können z.B. vorhandene Produktinformationen durch beliebig viele multimediale Zusatzinformationen angereichert werden.

vi) Profiling

Die Verknüpfung von Benutzer-Profilinformationen mit den Katalogdaten ermöglicht eine benutzerindividuelle Aufbereitung von Inhalten und Darstellung. Wodurch die spezifischen Bedürfnisse eines Kunden optimaler abgedeckt werden können und eine größere Kundenzufriedenheit und damit eine stärkere Kundenbindung erreicht werden kann.

Wie auch die anderen „Anwendungsfunktionen“ ist das Profiling natürlich nicht auf die Informationsphase beschränkt. Idealer Weise sollte es durchgängig alle Phasen der Geschäftsprozeßunterstützung begleiten. In der Informationsphase kann es sich z.B. auf die Gestaltung bzw. das „look and feel“ eines Systems auswirken, auf Preise, auf Shopping-Tipps, auf Zahlungsmethoden, Liefervereinbarungen, Suchergebnisse, angebotene Zusatzdienste, individuelle Benachrichtigungen, Cross-Selling und vieles mehr.

1.1.4.2 Vereinbarungsphase

In der Vereinbarungsphase geht es darum, Einigkeit zu erzielen über die Konditionen und Bedingungen, unter denen es zum Abschluß eines rechtsgültigen Kaufvertrages kommt. Dies kann z.B. im Rahmen eines Katalogsystems oder eines Versteigerungssystems geschehen. Während es bei normalen B2C-Shop-Lösungen ausreicht, eine Preis- und Konditionspolitik nach dem „Take it or leave it“-Prinzip anzubieten, müssen z.B. B2B-Lösungen eine ausgefeilte und individuelle Preis- und Vereinbarungsfindung unterstützen. Neben der individuellen Kalkulation (z.B. von Lieferpreisen in Abhängigkeit von geographischer Region und Nettogewicht) gehören online verfügbare „Bonitätsprüfungen“ (inklusive der Kontrolle von Kredit- und Bankinformationen) und eine ganze Reihe von „Transaction-Services“ zur Vereinbarungsphase. Funktionen die der Vereinbarungsphase zugeordnet werden können sind:

a) Kontrolle von eingegebenen Kredit- und Bankinformationen

Ein großer Teil der Fehler bei der Eingabe von Kreditkarten- oder Bankinformationen (die

z.B. durch einen Tippfehler bei der Eingabe der Kontonummer entstehen können) können durch Prüfsummen und Kontrollrechnungen frühzeitig erkannt werden. Dies erspart dem Anwendungsbetreiber eine Vielzahl von zusätzlichen „Bearbeitungsgebühren“ bei seinem Geldinstitut.

b) Online-Bonitätsprüfungen

Aufwendigste und teuerste Variante der Kontrolle von Zahlungsinformationen, die nicht nur die Richtigkeit der Angaben überprüft sondern zusätzlich eine Auskunft über die tatsächliche Zahlungskraft des jeweiligen Käufers gibt. Die Online-Bonitätsprüfung kommt vor allem bei B2B-Transaktionen zum Einsatz, bei denen Bestellungen entsprechende Größenordnungen aufweisen.

c) Individuelle Lieferpreise

Genau wie die Artikelpreise können auch die Lieferpreise individuell vereinbart werden bzw. einem Kalkulationsschema unterworfen sein. Denkbar wären z.B. Lieferpreise in Abhängigkeit von Entfernung zum Hersteller, in Abhängigkeit des Gesamtgewichtes oder -volumens oder in Abhängigkeit vom Gesamtpreis der Bestellung.

d) Wahl der Versandart

Die Lieferzeiten und Konditionen verschiedener Logistikunternehmen weichen zum Teil stark voneinander ab. Viele eCommerce-Systeme erlauben es deshalb dem Kunden selbständig aus einer Palette von Logistikunternehmen und unterschiedlichen Angeboten auszuwählen.

e) Wahl der Zahlungsmethode

Wie bei der Wahl der Versandart gehört es mittlerweile zum guten Ton, dem Kunden eine Palette von Zahlungsmethoden anzubieten, aus denen er dann frei wählen kann. Typischerweise werden Kreditkarte, Bankeinzug, Rechnung und Nachnahme angeboten. Wobei zu bedenken ist, daß es bei realen Systemen sinnvoll ist, die Auswahl unter Berücksichtigung von Randbedingungen beeinflussen zu können. (Die Bezahlung „per Rechnung“ soll z.B. nur den Stammkunden vorbehalten sein, Bestellungen „per Nachnahme“ z.B. aus Australien sollten automatisch verhindert werden.) Von den reinen „Internet-Zahlungsmethoden“, wie z.B. Cybercash, eCash oder TeleCash konnte sich noch keine durchsetzen. Dies liegt vor allem an dem Aufwand der vor der ersten Bestellung nötig ist

(Anmeldung, Briefverkehr, Wartezeiten, usw.), und an der geringen Verbreitung einer bestimmten Zahlungsmethode. Mit großer Sicherheit werden Online-Zahlungsmethoden in der Zukunft eine größere Bedeutung bekommen. Zur Zeit ist aber die Kreditkartenzahlung die am häufigsten angewandte Methode. Die dementsprechend von fast jedem eCommerce-System per Default unterstützt wird.

f) Steuermatrix

Im internationalen Handel sind die entsprechenden produktspezifischen und landesspezifischen Steuersätze zu berücksichtigen. ECommerce-Anwendungen müssen Beispielsweise alleine in Deutschland drei verschiedene Steuerklassen behandeln (Konsumgüter 16%, Bücher 12%, Lebensmittel 6%). In anderen Ländern gibt es abweichende Steuersätze und teilweise auch andere Steuerklassen. Soll ein eCommerce-System von Händlern in verschiedensten Ländern eingesetzt werden können, muß es die verschiedensten Besteuerungsmodelle flexibel unterstützen.

g) Transaction-Services

Bei einfachen B2C-Shop-Anwendungen genügt meist eine einfache Bestelltransaktion mit nachfolgender Bestätigung. Im B2B-Bereich kommen häufig verschiedene andere „Transaction-Services“ zum Einsatz. Bei einigen B2B-Marktplatzsystemen (wie z.B. bei CommerceOne) wird dann jede einzelne in Anspruch genommene Transaktion in Rechnung gestellt. Zu den Transaction-Services gehören:

- Availability-Request und Availability-Acknowledgement,
- Price-Request und Price-Acknowledgement,
- Order-Request und Order-Acknowledgement und
- Order-Status-Request und Order-Status-Acknowledgement

1.1.4.3 Abwicklungsphase

In dieser Phase der Geschäftsabwicklung erfolgt die eigentliche Abwicklung des Kaufvertrages. Neben der Methode der Bezahlung muß beim Vertrieb von physischen Gütern ein entsprechendes Versandverfahren angestoßen werden. Bei kleineren Mengen genügt die

Auswahl eines Logistikdienstleisters sowie der eventuelle Abschluß einer Transportversicherung. Sobald täglich mehrere hundert oder gar Tausende von Paketen versendet werden, müssen entsprechend effiziente Auslieferungs- und Kommunikationstechniken zu Logistikunternehmen und Produktionsstätten eingeführt werden.

Eine besondere Bedeutung für zukünftige eCommerce-Anwendungen hat der gesamte Bereich „Fullfilment“ [1-2]. Dazu gehören unter anderem Online-Informationen über Herstellungsprozeß, Lagerbestand oder Lieferstatus mit deren Hilfe enorme Einsparungspotentiale (z.B. in der Lagerhaltung) möglich sind. Dazu sei an dieser Stelle [1] zitiert: „Creating virtual supply chains collaborating in real time represents the largest opportunity in eBusiness in our view. At stake are billions of dollars in inventory reduction, transporting costs and process improvement“.

Funktionen, die der Abwicklungsphase zugeordnet werden können:

- Transparenz Lagerbestand,
- Transparenz Herstellungsstatus,
- Einleitung des Transportvorgangs,
- Transparenz Lieferstatus und
- Bezahlung

1.1.4.4 After-Sales-Phase

In der After-Sales-Phase geht es darum, die Kunden auch nach dem Kauf eines Produktes optimal zu betreuen, dadurch die Kundenzufriedenheit zu stärken und auf diesem Weg die Kundenbindung zu festigen. Zu den Anwendungsfunktionen die der After-Sales-Phase zugeordnet werden können gehören z.B.:

- Individuelle Benachrichtigungen,
- die Möglichkeit Beurteilungen über Produkte oder Geschäftspartner abzugeben,
- Schwarze-Bretter, News-Groups, Mailing-Listen, Chat,
- Supportangebote, wie Datenbanken mit Problemlösungen (Knowledge Bases)

und

- Unterstützung durch Call-Center.

1.1.5 Eingrenzung und Entstehung der Arbeit

In der vorliegenden Arbeit wird nicht auf die Bereiche eGovernment und eFinance eingegangen. Die Arbeit beschr nkt sich auf die Bereiche des eCommerce im Sinne von B2B, B2C, C2C und den Optimierungs- und Rationalisierungspotentialen durch elektronische L sungen.

Analog zu den historischen Entwicklungen des eCommerce, deren euphorischer Aufstieg mit den B2C-Shop-Systemen begann, starteten auch die der vorliegenden Arbeit zugrundeliegenden Untersuchungen, mit dem Bereich der eCommerce-Shop-Systeme. Dementsprechend war der erste Arbeitstitel: ªFramework f r eCommerce-Shop-Systeme .

Im Laufe der Untersuchungen — und w hrend der ausf hrlichen Implementierungs- und Erprobungsphasen — konnte festgestellt werden, da§, wie oben beschrieben, den verschiedensten eCommerce-Kategorien und unterschiedlichsten Gesch ftsmodellen zu gro§en Teilen identische Anwendungsfunktionen zugrundeliegen. Die Identifikation dieser Anwendungsfunktionen nebst der Erfahrungen bei Entwurf und Konstruktion von eCommerce-Systemen (siehe Kapitel 8), die Besch ftigung mit der Entwicklung von Internet-Anwendungen im allgemeinen (siehe Kapitel 2 und 3), die Bereitstellung von wiederverwendbarem User-Interface-Komponenten (siehe Kapitel 6) und die Besch ftigung mit geeigneten Informationsobjekten und Verarbeitungskomponenten (siehe Kapitel 4, Kapitel 5 und Kapitel 6) gaben den Ausschlag zur endg ltigen Form der vorliegenden Arbeit.

1.2 Motivation

Das Ziel dieser Arbeit ist der Entwurf und die Realisierung eines „eXBO-Frameworks für eCommerce-Systeme“, mit dem leistungsfähige individuelle eCommerce-Systeme deutlich schneller als bisher und deutlich kostengünstiger als bisher erstellt werden können. Diese Zielstellung leitet sich aus folgenden Fragestellungen ab:

Warum individuelle eCommerce-Systeme?

a) Verglichen mit der Anzahl von existierenden, unterschiedlichen Geschäftsmodellen, die in der real existierenden Wirtschaft existieren gibt es nur eine verschwindend kleine Zahl von „passenden Standardanwendungen“.

b) Die bisher entwickelten eCommerce-Standard-Lösungen sind meist abgeschlossene Anwendungen, die höhere Anforderungen an Ausbaufähigkeit, Flexibilität und Integrierbarkeit in vorhandene IT-Systeme nicht erfüllen können. Nur für die wenigsten Geschäftsmodelle, wie z.B. B2C-Shop-Lösungen existieren überhaupt Standard-Lösungen.

c) Trotz hoher Kosten wird meist die Entwicklung von eigenen Lösungen bevorzugt, da Standard-Lösungen, bei der enormen Vielfalt von real existierenden Geschäftsmodellen und unterschiedlichsten internen Geschäftsabläufen nur selten auf die vorhandenen und geplanten Geschäftsprozesse anwendbar sind.

d) Das moderne Geschäftsleben zeichnet sich durch ein stetiges Anwachsen von Angeboten, globalem Wettbewerb und Kundenerwartungen aus. Der ständig steigende globale Konkurrenzdruck erfordert von den Unternehmen in zunehmendem Maße Leistungen, die über „den Standard“ hinausgehen. Als Antwort darauf sind Unternehmen in der ganzen Welt damit beschäftigt, Lösungen zu entwickeln, die über den „Standard“ hinausgehen und die es ihnen erlauben, entsprechende Produkte und Dienstleistungen anbieten zu können. Individuelle Lösungen sind deshalb sogar auch für Geschäftsbereiche bzw. Geschäftsmodelle gefragt, für die es bereits Standard-Lösungen gibt, besonders dann, wenn diese Standard-Lösungen nur unzureichend erweiterbar und anpassbar sind.

e) Die Erfolgspotentiale von eCommerce-Anwendungen, wie Zeiteinsparung, Kosteneinsparung, Qualitätserhöhung und die Ausnutzung strategischer Potentiale können nur durch individuelle, auf das jeweilige Geschäftsmodell eines Unternehmens angepaßte Lösungen maximiert werden.

Warum ein Framework?

a) Ein großer Teil der grundlegenden Entitäten (wie z.B. Kunde, Rechnung, Lieferadresse, Zahlungsadresse, Produkt, Produktkategorie usw.) und ein großer Teil der Anwendungsfunktionen (wie z.B. Content-Management, Produktpräsentation, Benutzernavigation, Suche, Eingabevaledierung usw.) sind in den meisten Geschäftsmodellen identisch (siehe auch [1-2] und [4]). Durch das Zusammensetzen von fertigen, wiederverwendbaren, grundlegenden Komponenten aus einem „Baukasten“, ist man in der Lage, in minimaler Zeit zu einem Basissystem zu gelangen, das dann beliebig ausgebaut und den individuellen Bedürfnissen angepaßt werden kann.

b) Framework- bzw. "Baukasten"-Systeme für eCommerce sind, wie ihre Vorbilder (Lego- oder Fischertechnik-Baukästen), schon von ihrem grundlegenden Konzept her, für flexible Erweiterungen ausgelegt. Dies prädestiniert sie für das heutige Geschäftsleben, das durch immer schneller wechselnde Anforderungen und Geschäftsstrukturen geprägt ist. Die erhöhte Flexibilität wirkt sich dabei positiv auf alle heutigen und zukünftigen Erweiterungs-, Veränderungs- und Integrationsanforderungen aus. Durch die Wiederverwendung von fertigen, erprobten Komponenten werden die Entwicklungszeiten drastisch verkürzt, die Herstellungskosten herabgesetzt und die Zuverlässigkeit und Qualität erhöht.

Warum für eCommerce-Systeme?

a) Kaum ein anderer Bereich hat in den letzten Jahren eine solch rapide Entwicklung vollzogen wie eCommerce, obwohl sogar der Begriff „Electronic Commerce“ erst seit der jüngsten Verbreitung des World Wide Web in den Mittelpunkt des öffentlichen Interesses gerückt ist, existieren bereits heute Hunderte von Analysen und Prognosen über die Entwicklungspotentiale von eCommerce. In Abb. 3 ist eine der zahlreichen Prognosen für die Entwicklung des Weltmarktes für eCommerce dargestellt.

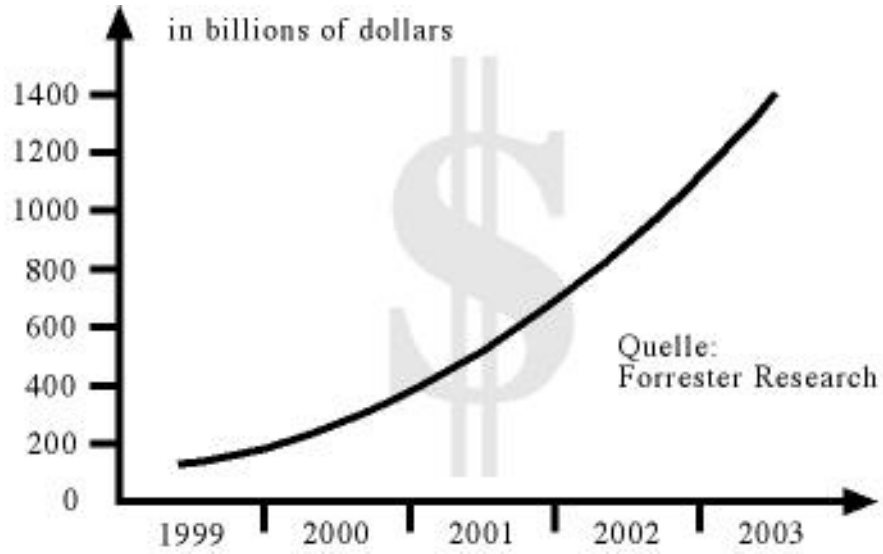


Abbildung 3: Prognostizierter Weltmarkt für eCommerce

b) Viele der Prognosen haben eine Gemeinsamkeit. Sie sagen dem eCommerce nicht nur eine gute sondern eine sehr gute Zukunft voraus. Eine Zukunft in der mit und durch den eCommerce exponentielle Umsatzsteigerungen zu erwarten sind, eine Zukunft in der der gesamte Bereich des eCommerce als eine treibende Kraft nicht nur für die Entwicklung der Software-Industrie, sondern für die Weiterentwicklung der gesamten Weltwirtschaft angesehen wird [1-4], [10-15].

c) Darüberhinaus wird eCommerce mittlerweile von vielen Nationen als ein wesentlicher Schlüssel für die internationale Wettbewerbsfähigkeit betrachtet [4].

d) Die wirtschaftlichen und gesellschaftlichen Auswirkungen des Internet im Allgemeinen und des Electronic Commerce im Speziellen sind so weitreichend und tiefgreifend, daß die Entwicklung von elektronischen Lösungen ein wesentlicher Zukunftsfaktor bzw. sogar ein überlebensnotwendiger Faktor für die Mehrheit der existierenden Unternehmen darstellen wird [1-4], [15].

e) In der Praxis existieren verschiedenste technologische, gesellschaftliche und wirtschaftliche Eintrittsbarrieren, die vor allem die kleinen und mittelständigen Unternehmen bisher daran hinderten, die vorhandenen Potentiale des

eCommerce nutzen zu können. Die wichtigsten von ihnen sind zu lange Entwicklungs- und Einführungszeiten und die damit verbundenen zu hohen Kosten bei der Einführung von eCommerce-Systemen. Die damit einhergehende lange Zeitspanne bis zur Amortisation von Investitionen in Rationalisierungsmaßnahmen durch eCommerce-Lösungen stellt vor allem für kleine und mittlere Unternehmen ein meist unzumutbares Risiko dar.

Warum Business-Objects?

a) Die Betrachtung von Geschäftsobjekten, von Objekten also, die ihre Entsprechung in der real existierenden Geschäftswelt haben, ermöglicht die Annäherung zwischen der betriebswirtschaftlich anwendungsbezogenen Sichtweise von Auftraggebern und Managern und der technisch implementierungsbezogenen Sichtweise von Entwicklern.

b) Eine schnelle und korrekte Umsetzung von Anforderungen aus der Geschäftswelt in einsatzfähige eCommerce-Systeme, erfordert - heute mehr denn je - eine enge Zusammenarbeit von Software-Ingenieuren und Experten aus den jeweiligen Geschäftsbereichen.

c) Durch die Betrachtung von Geschäftsobjekten können geschäftliche Konzepte, Prozesse und Elemente so beschrieben werden, daß sie von beiden Welten gleich gut und eindeutig verstanden werden. Auf diese Weise helfen Geschäftsobjekten dabei, Fehler in der Design- und Planungsphase von eCommerce-Systemen zu minimieren, woduch die Grundlage für eine schnellere und korrektere Umsetzung von Geschäftsmodellen in elektronische Lösungen geschaffen wird.

Warum eXtended Business-Objects?

Die in dieser Arbeit vorgestellten eXtended Business Objects (eXBO) erweitern die Konzeption von herkömmlichen Business-Objects in zwei wesentlichen Punkten:

- eXtended Business-Objects besitzen zusätzliches „Wissen“ über ihre Darstellung. Dies ermöglicht den Aufbau von eCommerce-Systemen, die gleichzeitig für eine Vielzahl von verschiedensten Endgeräten spezialisiert werden können.

- eXtended Business-Objects besitzen zusätzliches „Wissen“ über ihre Bear-

beitung. Dies ermöglicht es ihnen mit sogenannten „eXBO-Facility-Komponenten“ (vgl. Kapitel 5) bearbeitet zu werden. Die Kombination von eXBO und eXBO-Facilities ermöglicht bisher unerreichte Entwicklungsgeschwindigkeiten bei der Konstruktion von mittleren und großen eCommerce-Anwendungen. (Auf die möglichen Einsparungspotentiale wird in Kapitel 5 detailiert eingegangen.

1. Wissen über die Darstellung bzw. universelle Verwendung mit verschiedensten Endgeräten

Im Gegensatz zu herkömmlichen Business-Objects, deren Attribute in den meisten Fällen nur ausreichen, um sie in normalen Web-Anwendungen darstellen zu können, verfügen die eXtended Business-Objects über spezialisierte Attribute, die eine professionelle Darstellung und Bearbeitung mit unterschiedlichsten, heutigen und zukünftigen, Endgeräten ermöglichen.

Beispiel: Ein normales Business-Object (z.B. ein „Artikel“) verfügt beispielsweise über die Attribute „detailPicture“ (ein Bild dessen Größe und Datenmenge für die Darstellung mit einem Web-Browser auf einem PC-Monitor spezialisiert ist) und „longDescription“ (eine Beschreibung des Business-Objects deren Länge bzw. deren Umfang angemessen für die Darstellung in einem Web-Browser ist). Für andere Endgeräte, wie z.B. für einen PDA oder bei der Herstellung von gedruckten Katalogen sind diese Attribute vollkommen unzureichend. Dass „detailPicture“ ist beispielsweise für das PDA viel zu groß, für das WAP-Handy vollkommen unbrauchbar und für die Katalog-Herstellung zu unscharf.

eXBO verfügen über spezialisierte Attribute für unterschiedlichste Endgeräte und ermöglichen somit die Herstellung von eCommerce-Anwendungen, die gleichzeitig verschiedenste Endgeräte optimal unterstützen können.

Die hier vorgestellten eXBO begnügen sich dabei nicht mit einer einfachen Erweiterung von Attributen, die sie dann direkt (eventuell mit leerem Inhalt) zurückgeben. EXBO nutzen sogenannte „Manager-Objekte“ (wie z.B. einen „pictureManager“), die innerhalb der entsprechenden Zugriffs-Methoden aufgefordert werden, ein entsprechendes Bild zu ermitteln. Für den Fall, daß das spezialisierte Bild nicht vorhanden ist, kann ein „pictureManager“ dieses z.B. selbständig aus einem anderen generieren. Falls auch dies nicht möglich sein sollte, kann ein entsprechender „pictureManager“ wenigstens ein spezialisiertes Default-Bild anzeigen oder z.B. bei der Herstellung eines Kataloges eine Warnung ausgeben.

2. Wissen über ihre Bearbeitung

eXBO sind darüber hinaus in der Lage, für jedes einzelne Attribut mitzuteilen, wie es in Content-Management-Komponenten bearbeitet werden soll.

Beispiele: Für ein Attribut „longDescription“ soll in der Benutzeroberfläche ein großes Textfeld vorgesehen werden, ein Attribut „Kurzbeschreibung“ benötigt nur eine Textfeldzeile, für das Attribut „bestDetailPictureForWeb“ soll im User-Interface eine Upload-Funktionalität existieren (die es ermöglicht, lokale Bilder über das Internet zum Server zu überspielen und dort zu speichern), für ein Attribut „Anrede“ soll kein Textfeld existieren sondern eine Choice (Auswahlbox), die die Auswahl aus bestimmten vorgegebenen Werten (z.B. Herr, Frau oder Doktor) ermöglicht usw.

Durch diese Fähigkeit wird die Konstruktion von sogenannten eXBO-Facilities ermöglicht. Diese eXBO-Facilities, die die Verarbeitung von eXBO ermöglichen (zu ihnen gehören z.B. Komponenten mit denen eXBO editiert werden können), müssen fortan nur ein einziges Mal implementiert werden und sind dann für alle Business Objekte nutzbar. Dies bedeutet im Kern, daß mit der Konstruktion eines eXBO auch alle Arbeiten an dem dazugehörigen Content-Management-System abgeschlossen sind. Dass der Entwickler eines eXBO also von der Konstruktion entsprechender Verarbeitungskomponenten für das zugehörige Content-Management-System bzw. für den Backoffice-Bereich befreit wird. Dies ermöglicht bisher unerreichte Entwicklungsgeschwindigkeiten bei der Konstruktion von mittleren und großen eCommerce-Anwendungen. Auf die möglichen Einsparungspotentiale wird in Kapitel 5 im Detail eingegangen.

1.3 Gliederung der Arbeit

Das erste Kapitel (Einleitung und Überblick) gab zunächst eine kurze Einführung in den Bereich Electronic-Commerce. Dabei wurden Definitionen, verschiedene eCommerce-

Kategorien, grundlegende Anwendungsfunktionen und verschiedene Geschäftsmodelle vorgestellt. Anschließend wurde eine Eingrenzung des Themenbereiches der vorliegenden Arbeit vorgenommen und kurz auf die Entwicklungspotentiale und auf die große wirtschaftliche Bedeutung von eCommerce eingegangen.

Im zweiten Kapitel (Grundlegende Überlegungen beim Aufbau von eCommerce-Systemen) werden grundlegende Eigenschaften und Architekturen vorgestellt, die internet-basierte eCommerce-Anwendungen aufweisen sollten, um die erhöhten Anforderungen an eCommerce-Systeme bezüglich Erweiterbarkeit, Flexibilität und Integrationsfähigkeit in vorhandene und zukünftige IT-Strukturen gewährleisten zu können. Zu den angesprochenen Themen gehören die Unabhängigkeit von Datenbank- und Webserver-Herstellern, die Einbeziehung von gängigen Zahlungssystemen und Verschlüsselungstechniken sowie das dynamische Erzeugen von Internet-Seiten und die Session-Verwaltung. Die für eCommerce-Systeme besonders wichtige Eigenschaft der Skalierbarkeit wird in diesem Kapitel sowohl von der Software- als auch von der Hardwareseite betrachtet.

Im dritten Kapitel (Stand der Technik) wird der aktuelle Stand von Technik und Forschung dargelegt. Kapitel 3 ist in drei Bereiche gegliedert. Im ersten Teil wird der Stand von eCommerce-Standardlösungen betrachtet. Im zweiten Teil werden aktuelle Standardisierungsbemühungen, wie z.B. die der Object-Management-Group (OMG) im Bereich von Business-Objects vorgestellt. Der dritte Teil beschäftigt sich mit bestehenden Frameworks und Entwicklungsumgebungen, mit denen eCommerce-Systeme erstellt werden können.

In Kapitel 4 (Entwurfskonzepte für eCommerce-Systeme) werden am Beispiel eines eCommerce-Shop-Systems Entwurfskonzepte (bzw. Design-Pattern) für den Aufbau von eCommerce-Anwendungen vorgestellt. Das Beispiel orientiert sich an vorhandenen Systemen und Anforderungen aus der Praxis. Dabei wird auf die Unterscheidung von verschiedenen Rollen eingegangen und ihre Abbildungen in eCommerce-Systemen vorgestellt.
Die Besprechung von Funktionen und Entitäten, die von einem eCommerce-Shop-System erwartet werden, ermöglicht dabei einen ersten Überblick über grundlegende Komponenten von eCommerce-Systemen und ermöglicht dadurch ein besseres Verständnis der darauf folgenden Kaptitel.

In Kapitel 5 (eXBO-Facilities) werden die in Kapitel 4 dargestellten Erkenntnisse aufgegriffen und es werden konkrete Vorschläge für Verarbeitungskomponenten für Business Objekte gemacht. Dabei wird das zuvor vorgestellte Entwurfskonzept bzw. „Design Pattern“ des „Manager-Musters“, das in den meisten eCommerce-Anwendungen eingesetzt wird, und zudem von fast allen Business-Objects genutzt werden kann, umgesetzt und durch konkrete Implementierungsvorschläge erläutert. Außerdem werden die ersten Erweiterungen bzw.

Anforderungen an „eXBO“ vorgestellt, die eingeführt werden, um sie mit Hilfe der „eXBO-Facilities“ verarbeiten zu können.

Kapitel 6 (eXtended Business-Objects) ist das umfangreichste Kapitel und bildet zugleich den Kern der vorliegenden Arbeit. In diesem Kapitel werden die Bestandteile von eCommerce-Systemen und ihre Abbildung in eXBO behandelt. Bei der Auswahl von Geschäftsobjekten – die natürlich nur unvollständig sein kann – wurde besonderer Wert darauf gelegt, Objekte zu identifizieren, die besonders häufig in elektronischen Umsetzungen von Geschäftsmodellen anzutreffen sind. Passend zu jedem eXBO werden entsprechende, wiederverwendbare Komponenten zur Darstellung der eXBO in web-basierten eCommerce-Anwendungen vorgestellt. Zu Beginn dieses Kapitels wird ein weiteres Konzept eingeführt, das die „eXBO“ von „normalen“ Business-Objects abhebt - durch die Schaffung zusätzlicher Attribute und zugehöriger „Attribut-Manager-Objekte“ wird es möglich, eCommerce-Anwendungen zu erstellen, die gleichzeitig für verschiedene Endgeräte spezialisiert sind.

In Kapitel 7 (Supporting Facilities - Weitere Komponenten) werden weitere wiederverwendbare Komponenten vorgestellt, die geeignet sind, das vorgestellte Framework zu ergänzen und dazu beitragen, die Entwicklungsgeschwindigkeit von eCommerce-Anwendungen weiter zu steigern. Zu ihnen gehören z.B. Komponenten die das Erstellen von mehrsprachigen Anwendungen vereinfachen, Adapter-Objekte zu verschiedensten Datenhaltungssystemen oder allgemeine Komponenten wie Kalender und Login-Panel.

In Kapitel 8 (Realisierte eCommerce-Systeme) werden die mit dem vorgestellten Framework bereits realisierten eCommerce-Systeme vorgestellt. Dabei werden die wesentlichen technischen Aspekte der Systeme kurz zusammengefaßt und die Funktionen und Benutzeroberflächen in Auszügen vorgestellt. Außerdem wird ein kurzer Abriss der praktischen Erfahrungen gegeben, die bei Aufbau und Betrieb der Systeme gewonnen werden konnten.

Kapitel 9 gibt eine abschließende Zusammenfassung und einen Ausblick.

Im Anhang befinden sich Abbildungsverzeichnis, Literaturverzeichnis und Index.

Kapitel 2

Grundlegende Überlegungen beim Aufbau von eCommerce-Anwendungen

In diesem Kapitel werden – vor allem technologische – Grundlagen und Rahmenbedingungen angesprochen, die bei der Entwicklung von eCommerce-Systemen beachtet werden sollten. Geschieht dies zu kurzsichtig oder nicht intensiv genug, besteht die Gefahr, von übereilten Entwicklungen, die das Risiko von später schwer wieder gut zu machenden „Shortcomings“ in sich tragen. In vielen Fällen führt die Nichtbeachtung solcher grundlegenden Überlegungen letztlich sogar bis zum völligen Scheitern eines Systems. Studien wie [2] oder [3] zeigen, daß bis zu 40% aller Softwareprojekte letztendlich nicht erfolgreich sind. Die Gründe für ein Scheitern sind vielfältig und sind in technischen, politischen und wirtschaftlichen Ursachen zu finden.

In diesem Kapitel wird eine Auswahl von grundlegenden Überlegungen bzw. Kriterien vorgestellt, die - bei konsequenter Anwendung - dazu beitragen, wenigstens die technologischen Gründe zu minimieren.

Im Rahmen dieser Arbeit wurden an die 30 Produktvergleiche, Anforderungslisten, Bewertungskriterien, Business Reports etc. analysiert [6]. Dabei konnten weit über 100 verschiedene Kriterien gefunden werden [6-7]. Je nach Betrachtungsstandort, Umfeld und Anwendung wurden den einzelnen Kriterien dabei verschiedene Gewichtungen gegeben. Kriterien, die in einer Studie als fundamental eingestuft werden, werden in anderen nur als Seiten-Effekt wahrgenommen oder überhaupt nicht behandelt.
Die nachfolgende Liste von Kriterien ist ein Auszug aus [6] und stellt eine Auswahl der häufigsten und für am wichtigsten befundenen Kriterien dar:

- Unabhängigkeit von Datenbankherstellern
- Allgemeine Unabhängigkeit von Datenquellen
- Unabhängigkeit von Web-Server-Herstellern
- Getrennte Entwicklung von Design und Logik
- Allgemeine Unabhängigkeit von User-Interface-Technologien und Endgeräten
- Architekturen für eCommerce-Systeme
- Integrationsfähigkeit in vorhandene IT-Systeme
- User-Profiling
- Managing-State
- Erzeugung dynamischer Internetseiten
- Unterstützung verschiedener Hardware und Betriebssystemplattformen
- Unterstützung von Standards
- Sicherheit
- Zahlungssysteme
- Funktionen

- Benutzerfreundlichkeit
- Skalierung und Performance
- Technologie, Programmiersprachen und Entwicklungsumgebungen
- Wiederverwendung und Objektorientierung
- Erweiterbarkeit und Flexibilität

In den nachfolgenden Abschnitten dieses Kapitels werden diese Kriterien in Auszügen erläutert. Im nächsten Kapitel werden aktuelle eCommerce-Standardlösungen und die in dieser Arbeit vorgestellten Prototypen (vergleiche Kapitel 8) anhand dieser Kriterien analysiert.

2.1 Unabhängigkeit von Datenquellen

Viele eCommerce-Systeme werden bereits mit einer (meist relationalen) Datenbank ausgeliefert. Oft ist dabei die Kopplung zur Datenbank so eng, daß ein Wechsel zu einem anderen Datenbankmanagementsystem sehr schwer fällt oder gar unmöglich ist. Ursache hierfür ist z.B. ein häufiges Absetzen von SQL-Statements direkt aus dem Source-Code. Der Wechsel zu einer anderen Datenbank (und dementsprechend zu einer anderen SQL-Dialektik) würde Änderungen im gesamten Source-Code nach sich ziehen.

Eine andere Ursache für eine starke Datenbankabhängigkeit kann darin liegen, daß zuviel Logik (z.B. in Form von Stored-Proceeders) in die Datenbank gelegt wurde.
Die entsprechende Funktionalität geht bei einem Wechsel des Datenbankherstellers verloren.

Größere Flexibilität allerdings zu Lasten der Performance bietet z.B. die Verwendung von Standards wie ODBC, die den Zugriff auf verschiedene Datenbanksysteme erlauben.

Am leistungsfähigsten erwiesen sich bisher Konzepte von Adaptern, wie sie aus der objektorientierten Programmierung bekannt sind. Im Source-Code sieht dabei der Datenbankzugriff bzw. das Ansprechen des Objektes immer gleich aus. Die spezialisierten Adapter-Objekte für die jeweiligen Datenbanksysteme sind in der Lage daraus immer das „richtige“ SQL zu erzeugen. Soll die Datenbank gewechselt werden, genügt das Austauschen des Datenbank-Adapter-Objektes.

In der Praxis genügt es für eCommerce-Systeme häufig nicht sich auf relationale Datenbanksysteme als Datenspeicher bzw. Datenquelle zu begrenzen.
Eines der wichtigsten Aspekte für eCommerce-Systeme ist ihre Möglichkeit, sich in bestehende IT-Infrastrukturen einzupassen. Und in diesen gibt es eine Vielzahl von Altsystemen (z.B. hierarchische Datenbanken), historisch gewachsene Systeme (z.B. Flugabfragesysteme mit einer eigenen Abfragesprache), Warenwirtschaftssysteme, Enterprise Resource Planungssysteme (ERP) usw.
eCommerce-Systeme sollten eine Vielzahl von Datenquellen von Haus aus unterstützen und auf die Integration von verschiedensten Datenquellen vorbereitet sein [1-2], [14], [35-36].
Die Leistungsfähigsten unter ihnen sollten es darüber hinaus erlauben mehrere verschiedene Datenquellen gleichzeitig einzubinden [35], [140], [152].

2.2 Unabhängigkeit von Webserver-Herstellern

Fast alle Web-Server nutzen per Default den CGI-Standard (Common Gateway Interface). Einer der Nachteile von CGI ist die Eigenschaft für jeden CGI-Aufruf einen eigenen Thread zu starten, was bei sehr hohen Zugriffszahlen zu Performance-Beschränkungen führen kann [34].

Große Hersteller wie z.B. Netscape oder Microsoft haben deshalb zusätzlich zum unterstützten CGI optimierte Interfaces entwickelt (Netscape: NSAPI, Microsoft: ISAPI), die wahlweise genutzt werden können und die die Leistungsfähigkeit des Web Servers ausbauen [31].

Wünschenswert für eCommerce-Systeme wäre die Möglichkeit der freien Wahl zwischen CGI, NSAPI oder ISAPI. Am besten unterstützt durch Adapter-Objekte oder ähnliche „Kopplungsstücke“ die einen schnellen und unkomplizierten Wechsel ermöglichen.

In der Praxis unterstützen leider nur die wenigsten Systeme mehr als CGI. Andere Systeme wiederum (z.B. diejenigen, die auf Microsoft-ASP-Technologie basieren) unterstützen zwar ISAPI, dafür aber nichts anderes. Sie funktionieren also mit keinem anderen Web-Server [31].

2.3 Kapselung der Business-Logik

Die konsequente Kapselung der Business-Logik gegenüber der Benutzeroberfläche und gegenüber der Datenquelle hat Auswirkungen auf die Wiederverwendbarkeit, die Flexibilität, die Unabhängigkeit von Endgeräten, die Skalierbarkeit, die Entwicklungsgeschwindigkeit und die Möglichkeit Design und Technik parallel zu entwickeln. Wird beispielsweise die Business-Logik zu großen Teilen in die Benutzeroberfläche gebracht (typisch z.B. für Java-Script, Jscript, ASP-Anwendungen) sind große Mengen Source-Code im HTML enthalten. Eine durchschnittliche Design-Agentur kann dann nicht mehr damit umgehen, um z.B. die

Anwendung den „Corporate-Identity“-Vorgaben des Unternehmens anzupassen. Umgekehrt wäre es für die Entwicklungsabteilung fatal, wenn unbedarfte Designer aus Versehen auch nur ein einziges Zeichen aus dem Source-Code löschen oder verändern würden.

Mit Systemen, die statt Source-Code nur kleine „Tags“ im HTML-Code aufweisen, wird schon heute eine parallele Entwicklung von Design und Anwendung betrieben, was die gesamte Entwicklungszeit bzw. „Time To Web“ verkürzt. Soll eine vorhandene eCommerce-Anwendung später verschiedenste Endgeräte (siehe z.B. die aktuellen Entwicklungen im UMTS-Umfeld in Bezug auf mobile Geräte oder die neuesten Bestrebungen für ein digitales, internetfähiges Fernsehen) und somit verschiedenste Oberflächen unterstützen, so müßte die Business-Logik ebenfalls nicht neu implementiert werden.

Umgekehrt führt eine zu enge Bindung an die Datenbank (durch direkte SQL-Aufrufe oder Stored-Proceeders, vgl. Abschnitt Datenbanken/Datenquellen) zu einer zu starken Abhängigkeit von einem Datenbankenhersteller. Das eCommerce-System wird unflexibel bezüglich der möglichen Datenquellen.

Sind die Bereiche Oberfläche, Business-Logik und Datenspeicherung nicht konsequent (am besten durch entsprechende Zugriffsschichten) voneinander getrennt, fällt auch die Aufteilung auf verschiedene voneinander getrennte Rechner (zwecks Skalierung) schwerer.

In der Praxis preisen viele Systeme ihre n-Tire-Architecture an, meistens ist dabei aber die Kopplung zwischen den einzelnen Ebenen zu starr (z.B. durch die schon angesprochenen direkten SQL-Aufrufe), so daß die Flexibilität in beide Richtungen (Datenquellen und Oberflächen) entsprechend eingeschränkt wird.

Ideal wäre eine Kapselung der Business-Logik verbunden mit der Möglichkeit die „Kapseln“ nach Bedarf auszutauschen, um so flexibel mit verschiedensten Datenquellen und Endgeräten arbeiten zu können.

2.4 Architekturen für eCommerce-Systeme

Abgeleitet aus den vorangegangenen Forderungen bezüglich der Kapselung der Business-Logik, dem Wunsch nach Unabhängigkeit von Datenbankherstellern bzw. gänzlicher Unabhängigkeit von Datenquellen und dem Anspruch verschiedenste Endgeräte unterstützen zu können, wird die nachfolgende Architektur für den Aufbau von eCommerce-Anwendungen vorgeschlagen:

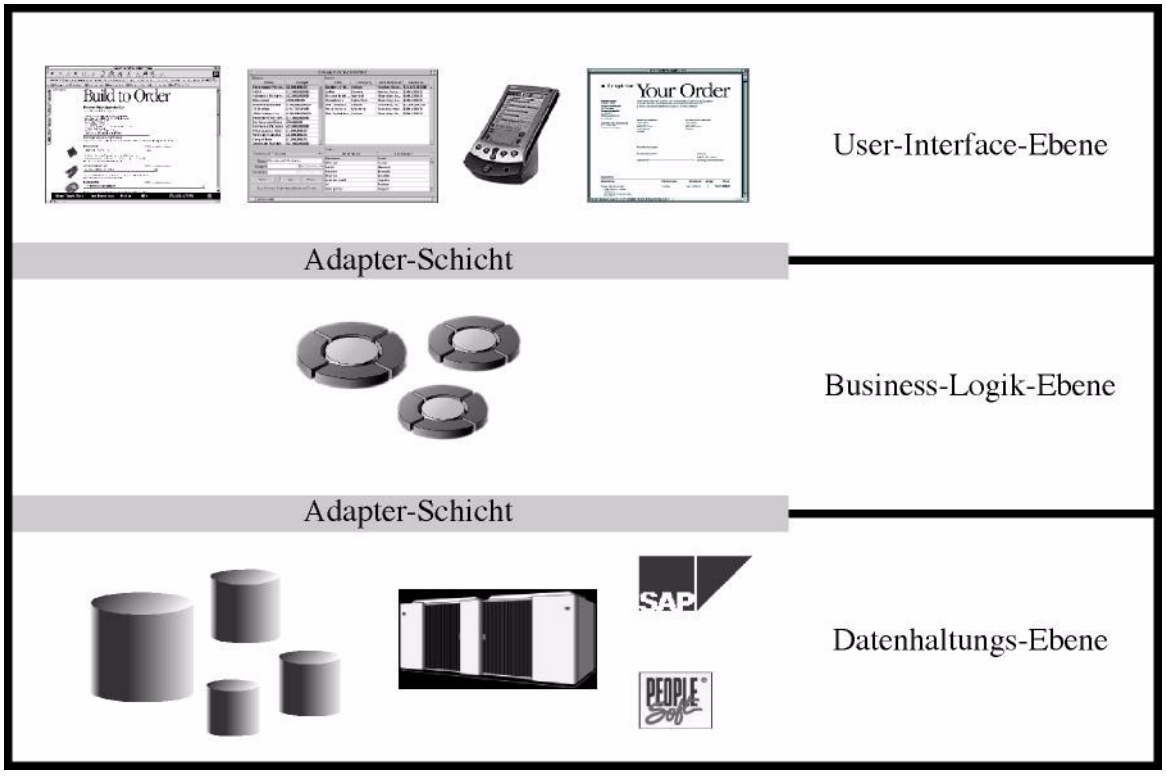


Abbildung 4: Drei-Ebenen-Architektur mit konsequenter Trennung der einzelnen Schichten

Nur wenn die drei grundlegenden Ebenen (Datenhaltung, Business-Logik und User-Interface) durch zusätzliche „Trennschichten“ (die z.B. in Form von Adapter-Objekten realisiert werden können) konsequent voneinander getrennt werden, ist eine maximale Flexibilität bezüglich verwendeter Datenhaltungssysteme und eingesetzter Endgeräte möglich. Die Nichtbeachtung

von grundlegenden architekturellen Überlegungen hat entscheidenden Einfluß auf die Integrationsfähigkeit eines eCommerce-Systems in bestehende und zukünftige IT-Infrastrukturen.

2.5 Session-Verwaltung

Spricht man heutzutage über eCommerce- oder eGovernment-Anwendungen, so sind implizit fast immer Webanwendungen gemeint, die orts- und plattform-unabhängig mit einem Webbrowser bedient werden können. Alle diese Webanwendungen haben eines gemeinsam: sie nutzen das Internet. Das Internet wurde primär für die Präsentation und den Austausch von Daten entwickelt und ist vollkommen zustandslos, bzw. kennt von Haus aus keine Zustände oder gar Sessions.

ECommerce-Anwendungen müssen also eine eigene Zustands- bzw. Session-Verwaltung besitzen um Dinge, wie z.B. getrennte Warenkörbe für viele parallele Nutzer zu ermöglichen. Um einen Request (Anfrage eines Nutzers) eindeutig zu einer Session zuordnen zu können müssen zwei Dinge gewährleistet sein. Eine Session muß eindeutig identifizierbar sein, d.h. Sie muß eine eindeutige Kennung (ID) besitzen und sie muß dem Client (Webbrowser) und dem Server (Webanwendung) bekannt sein. Neben dieser Session-ID müssen, für jede Session getrennt, weitere relevante Daten (wie z.B. die oben angesprochenen Artikel im Warenkorb) gespeichert werden.

Die Speicherung bzw. Verwaltung aller session-relevanten Informationen kann auf verschiedene Weisen und an verschiedenen Orten vorgenommen werden. Grob läßt sich zwischen Sessionverwaltung beim Client und Sessionverwaltung beim Server unterscheiden. Bei der Sessionverwaltung im Client werden die sessionrelevanten Daten über das Internet

zum Client übertragen und auf dem Rechner des Benutzers gespeichert. Die Speicherung beim Client kann z.B. in Cookies, in „hidden fields“ oder in Proxy-Objekten (z.B. Java-Applets) vorgenommen werden [41], [138-139], [153]. Bei der Sessionverwaltung im Server bleiben alle Daten im Serverrechner und werden entweder in der Anwendung oder in einer Datenbank gespeichert.

Beide Verfahren haben Vor- und Nachteile. Die Sessionverwaltung im Client, die in der Praxis häufig durch Cookies realisiert wird, ist technisch einfach zu realisieren und wird deshalb vor allem von „low-end“-Systemen eingesetzt. Diese Methode hat aber zwei entscheidende Nachteile:

- Session-relevante Daten, die eventuell sicherheits-relevante Informationen enthalten, werden ständig über das Netz hin und her übertragen.
- Die Cookie-Methode funktioniert nur wenn der Nutzer auf der Client Seite Cookies akzeptiert. Was ein großer Teil von Internet-Nutzern aber ablehnt [1-5]. Ohne den Cookie allerdings ist die Anwendung dann nicht mehr funktionsfähig. „Man kann es fast schon als peinlich bezeichnen, wenn eine Website gänzlich auf diese Methode baut und somit für den Benutzer, der – aus welchen Gründen auch immer – Cookies ablehnt, unbrauchbar ist“ [41].

Die Sessionverwaltung im Server stößt weder auf Akzeptanzprobleme noch auf Sicherheitsbedenken. Dafür ist sie wesentlich schwieriger umzusetzen. Ist die Sessionverwaltung nicht Bestandteil einer leistungsfähigen Middleware und muß selbst programmiert werden, ist ein ganzes Team von erfahrenen Entwicklern notwendig, um eine stabile und hochskalierbare Sessionverwaltung zu implementieren. Eine serverseitige Sessionverwaltung ist demnach selten bei „low-end“-Systemen anzutreffen und eher dem professionellen bzw. High-End-Bereich zuzuordnen. Für eine ausführlichere Erörterung der verschiedensten Varianten zur Sessionverwaltung siehe zum Beispiel [138-139] oder [41].

2.6 Unterstützung verschiedener Hardware- und Betriebssystem-Plattformen

Bezüglich der Untersuchung verschiedener Betriebssysteme gibt es, wie für viele andere Kriterien auch, kontroverse Diskussionen. Bei den einen ist z.B. die Unterstützung von Windows NT/98 ein absolutes Muß, um der großen Verbreitung dieses Systems gerecht zu werden. Andere wiederum, z.B. solche mit extremen Sicherheitsanforderungen lehnen Systeme, die ausschließlich auf Windows NT/98 basieren, von vornherein ab, da Rechner mit diesen Betriebssystemen in der Regel nicht sicher genug sind oder gar Hintertüren für Geheimdienste aufweisen [44-48].

Allgemein kann man beobachten, daß vor allem kostengünstige Low-End-Systeme oft nur auf der Windows-Plattform verfügbar sind, während High-End-Systeme eine modulare Architektur aufweisen, welche die Verteilung der einzelnen Komponenten auf verschiedene Betriebssysteme und Hardwareplattformen erlaubt. Zusätzlich gilt noch immer die Einschätzung, daß Windows NT/98 basierte Systeme preisgünstiger sind, auf Unix basierte Systeme hingegen wesentlich ausfallsicherer und besser zu skalieren sind [34].

Zusammenfassend läßt sich sagen, daß es für Systemanbieter nicht verkehrt ist, mindestens zwei verschiedene Varianten, eine für Windows und eine für Unix anzubieten.

2.7 Unterstützung von Standards

Die Unterstützung von Standards spielt eine große Rolle bei der Integration in bestehende IT-Infrastrukturen. Besonders die Kommunikations- und Kooperationsmöglichkeiten mit anderen eCommerce-Systemen werden in Zukunft noch weiter an Bedeutung gewinnen [1-5], [37-39]. Für eCommerce-Systeme sind dabei eine ganze Reihe von Standards aus unterschiedlichsten Bereichen von Bedeutung. Dazu gehören Standards wie z.B. OMG-CORBA, DCOM, EJB aus dem Bereich der objektorientierten Anwendungsentwicklung, CGI, NSAPI, ISAPI aus dem Bereich Webserver-Schnittstellen, P3P aus dem Bereich User-Profiling, TINA, TSAS und Parly aus dem Bereich Telekommunikation, FIPA und MASSIV aus dem Bereich Mobile-Agenten, Cybercash, eCash, Telecash, SET aus dem Bereich elektronisches Bezahlen, und SSL, PGP, SHTTP, RSA aus dem Bereich Sicherheit und Kryptographie um nur einige zu nennen. Genauere Informationen zu einzelnen Standards kann der geeigneten Literatur, wie z.B. [5], [9], [14], [53-59], [110] entnommen werden. Je nach Anwendungsfall und Einsatzgebiet eines eCommerce-Systems variiert die Bedeutung eines Standards, so daß an dieser Stelle keine allgemeinen Einstufungen oder Priorisierungen vorgenommen werden können.

2.8 Performance

Die Performance bzw. die Antwortzeiten eines Systems sind ein wesentliches Kriterium für Web-Anwendungen. Je kürzer die Antwortzeit desto besser. Sinkt die Performance unter ein gewisses Mindestmaß ist mit einer Akzeptanz-Einbuße bis hin zum Totalverlust von Kunden zu rechnen. Die Performance von einem System verschlechtert sich mit zunehmender Zahl von parallelen Usern und Größe des Datenbankinhalts (Gegenmaßnahmen s. Skalierung). Allerdings unterscheiden sich verschiedene Systeme von Haus aus (auch bei gleicher Userzahl und DB-Größe) stark in ihrer Performance. Hierzu sei z.B. auf die Vergleichsstudie zu Applicationservern vom Französischen Institut SQLIngenierie hingewiesen [154].

2.9 Skalierbarkeit

Die Skalierbarkeit eines Systems beschreibt die Möglichkeit, sich bei wachsendem Erfolg den ständig steigenden Anforderungen an Performance und Inhalt anzupassen. Es kann mitunter fatal für ein Unternehmen sein, wenn es sein eigenes Wachstum nicht durch ein leistungsfähiges System kompensieren kann. Für jeden eCommerce-Anbieter kann ein instabiles System oder zu lange Ladezeiten den Todesstoß bedeuten. Plant man ein eCommerce-System, das im Erfolgsfall von sehr vielen Anwendern genutzt wird, sollte man sich von vornherein im klaren darüber sein, daß von web-basierten eCommerce-Systemen ein Höchstmaß an Skalierbarkeit gefordert wird. ECommerce-Systeme müssen dabei in zwei verschiedene Richtungen skalierbar sein:

- Skalierbarkeit bezüglich der Antwortzeiten des Systems - auch bei Hunderten oder Tausenden von parallelen Zugriffen soll sich die Antwortzeit nur minimal verschlechtern.

- Skalierbarkeit bezüglich der Inhalte - genau wie eine große Anzahl von parallelen Zugriffen kann auch eine große Anzahl von z.B. Artikeln in der Datenbank zu erheblichen Verlängerungen von Antwortzeiten führen.

Die Qualität der Entwickler und die Ausführlichkeit von diesbezüglichen Tests bei der Systementwicklung sind leider nicht die einzigen Faktoren für eine gute Skalierbarkeit. Hochskalierbare eCommerce-Systeme fordern optimale Performance von jeder einzelnen Komponente: Webserver, Applicationserver, Datenbankserver, Betriebssystem und Hardware.

Für fast alle Komponenten gilt dabei, daß höhere Performance durch einen höheren Preis eingekauft werden muß.

Die zuvor angesprochene Kapselung der eigentlichen Programmfunktionalität zu Oberfläche und zur Datenquelle kommt auch hier, bei den Skalierungs- bzw. Wachstumsmöglichkeiten eines Systems voll zum Tragen. Sind die Ebenen sauber voneinander getrennt, kann ein Unternehmen mit einer freien oder preisgünstigen Datenbank starten und später, bei entsprechendem Erfolg, durch den Austausch einiger weniger Komponenten zu einer professionellen, hochperformanten Datenbanklösung wechseln. Die Anschaffung eines entsprechenden Datenbanksystems wie z.B. eines Oracle-Enterprise-Datenbankmanagementsystems für Webanwendungen (welche mit mehreren hunderttausend Euro Lizenzgebühren einher geht) sind für die Startphasen der meisten Unternehmen unrealistisch.

Wurde aber zu Beginn auf ein System gesetzt, das nicht genügend ausbaufähig ist, muß eine kostspielige Neuentwicklung gestartet werden. Viel schlimmer noch ist die Tatsache, daß die Engpässe des Altsystems bis zur Inbetriebnahme des neuen hingenommen werden müssen. Ein Problem, das in der schnellebigen Internet-Welt, „in der die Konkurrenz nur einen Klick weiter ist“, zu Image- und Akzeptanz-Problemen führt, und letztendlich den Verlust von vielen Nutzern bedeuten kann [4].

2.10 Eingesetzte Technologien und Programmiersprachen

CGI-Pearl, PHP, Java, C++, ObjectiveC, Java-Script, Virtual-Basic, Jscript, ASP, WebObjects und Java-Servlets sind nur eine Auswahl von Programmiersprachen bzw. Technologien die in der Praxis für die Entwicklung von dynamischen Webseiten eingesetzt werden. Eine genaue Analyse der verschiedenen Technologien würde den Rahmen dieses Kapitels sprengen. Deshalb sei hier auf die entsprechende Literatur verwiesen [40-43], [48], [50], [51-52], [93], [110], [121], [102-115], [134-142].

In diesem Abschnitt soll nur auf einige allgemeine Einschätzungen hingewiesen werden. Die seit Jahren anerkannten Vorteile der objektorientierten Programmierung, wie z.B. bessere Wartbarkeit, Wiederverwendung, größere Zuverlässigkeit, und nicht zuletzt erheblich kürzere Entwicklungszeiten (vor allem bei größeren Projekten) sind nur durch die Wahl einer echten objektorientierten Programmiersprache bzw. Programmierumgebung gegeben.

Außerdem ist zu beachten, daß die im Abschnitt „Kapselung der Business-Logik“ angesprochenen architekturellen Grundentscheidungen durch die Wahl der verwendeten

Technologie beeinflußt, bzw. oft sogar vorweggenommen werden. Wird z.B. Java-Script (das ins HTML hineingeschrieben wird) eingesetzt um Teile der Business-Logik zu implementieren, ist eine saubere Trennung zwischen Oberfläche und Logik nicht mehr möglich (siehe auch Abschnitt „Kapselung der Business-Logik“).

In diesem Zusammenhang bleibt anzumerken, daß eine der Hauptursachen für den Erfolg des World Wide Web in der „Demokratisierung“ der Informationsgesellschaft liegt. Die Grundidee des World Wide Web ist es, daß jeder von jedem Ort, mit jedem Rechner und jedem Betriebssystem gleichberechtigt und unkompliziert, Zugang zu Informationen bekommt. Entwicklungen, wie z.B. Java-Script- oder Jscript-Anwendungen, die nur von einigen wenigen Webbrowsern interpretiert werden können verletzen zudem die grundlegenden Ideen des World Wide Web [51-52], [178].

Bei der Wahl der Technologie sollten außerdem die Skalierungsmöglichkeiten und die Leistungsfähigkeit und Zuverlässigkeit der zur Verfügung stehenden Entwicklungswerkzeuge bzw. Entwicklungsumgebungen berücksichtigt werden.

Bei der Planung eines eCommerce-Systems sollte auch an die Kosten gedacht werden, die entstehen, wenn das Unternehmen großen Erfolg hat und die Einzelkomponenten: Webserver, Applicationserver, Datenbankserver. Betriebssystem und Hardware dementsprechend aufgerüstet bzw. skaliert werden müssen. Nach Möglichkeit sollte die zur Systementwicklung herangezogene Anwendungs- und Programmierumgebung eine entsprechende Aufrüstung flexibel unterstützen, damit die getätigten Investitionen in die eCommerce-Anwendung gesichert sind und man nicht zu einer Neuentwicklung gezwungen wird.

Kapitel 3

Stand der Technik

Im Rahmen dieser Arbeit wurden existierende eCommerce-Anwendungen, aktuelle Standardisierungsbemühungen, Frameworks und Entwicklungsumgebungen für eCommerce-Systeme betrachtet. Dieses Kapitel ist in zwei Bereiche gegliedert.

Der erste Teil beschäftigt sich mit existierenden eCommerce-Systemen. In diesem Zusammenhang werden existierende eCommerce-Standard-Lösungen kritisch begutachtet und mit den eCommerce-Anwendungen verglichen, die mit dem vorgestellten Framework bereits realisiert werden konnten.

Der zweite Teil behandelt neben aktuellen Standardisierungsbemühungen, wie z.B. der Object-Management-Group (OMG) im Bereich Business-Objects, existierende Frameworks und Entwicklungsumgebungen für eCommerce-Systeme.

3.1 eCommerce-Systeme

In diesem Bereich wurden zwei größere Untersuchungen durchgeführt. Eine Untersuchung beschäftigte sich mit dem Themenkomplex Business to Business (B2B). Anhand von zahlreichen Bewertungskriterien [6] wurden verschiedene B2B-Plattformen untersucht. Die Ergebnisse dieser Untersuchung sind in dem Report „eBusiness and eGovernment - Technological Aspects and Product Evaluation“ [1] zusammengefaßt und wurden im Rahmen des ersten GMD-Forums von Prof. Mendes vorgestellt. Die Einzelergebnisse der untersuchten Systeme (u.a.: Ariba, Brokat, Siemens-Service-Select, Telekom-Dienste-Plattform, BEA und Oracle), sind in den folgenden Dokumenten festgehalten [1], [91-97]. Die Haupterkenntnisse aus diesen Studien sind:

- Es existieren wesentliche Unterschiede zwischen B2C- und B2B-Anwendungen. B2B-Anwendungen sind in der Regel wesentlich komplexer, weisen wesentlich größere Transaktionsvolumina auf und es werden an sie besonders hohe Erwartungen bezüglich Stabilität und Sicherheit gestellt.
- Es existiert ein großer Bedarf und ein riesiger Markt für B2B-Anwendungen.
- Es besteht der Bedarf elektronische Lösungen für unzählige, verschiedene Geschäftsmodelle zu entwickeln.
- Durch den Einsatz von eCommerce-Systemen im Business to Business-Bereich können enorme Optimierungs- und Rationalisierungspotentiale genutzt werden.

Die zweite Untersuchung beschäftigte sich mit dem Themenkomplex Business to Customer (B2C). In diesem Zusammenhang wurden zahlreiche eCommerce-Shop-Systeme untersucht. Die marktführenden Anwendungen wurden im Rahmen von Projekt- und Studienarbeiten detailliert betrachtet [77-78]. Einzelergebnisse sind folgenden Dokumenten zu entnehmen: Intershop [28], [77-78], [79-81], [98-99] OpenShop [77-78], [86-87], Mercantec Softcart [77-78], [88-89], IBM-Websphere Commerce Suite [77-78], [82], Easymarket [77-78], iCat [77-78], [83-85], [90] und Microsoft Site Server Commerce Edition [48], [77-78], [92-93]. Ergänzende Literatur zu den einzelnen Systemen findet man in: [14], [19-20], [98-99].

Im folgenden wird eine zusammenfassende Darstellung der B2C-Untersuchungen vorgenommen. Dabei werden die untersuchten Systeme sowohl untereinander, als auch mit

denjenigen eCommerce-Systemen verglichen, die mit dem vorgestellten Framework bereits realisiert werden konnten. Zusätzlich werden die Ergebisse der B2B-Untersuchungen genutzt und ebenfalls Vergleiche zu den Marktführenden B2B-Anwendungen vorgenommen. Als Betrachtungskriterien werden – neben ergänzenden Punkten – vor allem die im vorangegangenen Kapitel erläuterten „grundlegenden Überlegungen beim Aufbau von eCommerce-Systemen“ herangezogen.

Entscheidender Punkt bei der Gegenüberstellung der verschiedenen Systeme ist der jeweilige Vergleich mit den in dieser Arbeit vorgeschlagenen Konzepten, bzw. mit den eCommerce-Anwendungen, die mit Hilfe des vorgestellten Frameworks erstellt wurden. Diese werden nachfolgend durch den Namen *eXBO-Anwendung* gekennzeichnet.

3.1.1 Unabhängigkeit von Datenbankherstellern

Die Untersuchungen ergaben, daß die marktführenden Anwendungen fast durchgängig eine Datenbankanbindung vorsehen, wohingegen die Masse der „Low-Budget“-Anwendungen versucht ganz ohne eine Datenbank auszukommen. Die Datenbankanbindungen der untersuchten Systeme sind größtenteils sehr unflexibel. Das heißt ein Wechsel von der jeweiligen, meist mitgelieferten Datenbank, zu einer anderen ist sehr umständlich und in vielen Fällen nicht möglich. Die „innovativsten“ Lösungen unterstützen den ODBC-Standard und ermöglichen so die Zusammenarbeit mit zahlreichen ODBC-kompatiblen-Datenbanken. Flexible Konzepte wie die in Punkt 2.1 vorgestellte „Trennschicht zwischen Datenbank und Anwendung“, z.B. in Form von Adapter-Objekten sind in den untersuchten B2C-Lösungen bisher nicht anzutreffen (im Gegensatz zu den wesentlich komplexeren und weitaus teureren B2B-Anwendungen wie z.B. Ariba [1-2]). Einige Hersteller, die die Notwendigkeit die Datenbank wechseln zu können, erkannt haben, versuchen dieses durch Backoffice-Tools zu unterstützen (z.B. durch Tools, die im gesamten Source-Code nach Datenbankanfragen suchen und diese ersetzen können [48]. [77-78]).

Die Untersuchungsergebnisse werden in der nachfolgenden Tabelle dargestellt. Wie oben erläutert, steht der Name *eXBO-Anwendung* dabei für Anwendungen, die mit dem vorgestellten Framework erstellt worden sind und bezeichnet jeweils die optimale bzw. anzustrebende Lösung (vergleiche Kapitel 2). Wenn nicht explizit anders ausgezeichnet, dann weisen, die mit den vorgestellten Framework realisierten eCommerce-Systeme exakt diese Lösung auf (vergleiche Kapitel 8).

System	Unabhängigkeit von DB-Herstellern + gut; **o** mittelmäßig; - schlecht
eXBO	+ 3 Ebenen mit Adapter-Schichten
ARIBA (B2B)	+ 3 Ebenen mit Adapter-Schichten
Intershop	- festgelegt auf Sybase
Openshop	**o** ODBC
Mercantec Softcart	- keine Datenbank nur Text-Dateien
iCat Commerce Suite	**o** ODBC
Microsoft Site Server Commerce Edition	**o** ODBC
Easymarket	- keine Datenbank
IBM Websphere Commerce Edition	- Festlegung auf zwei DB-Hersteller

Abbildung 5: Unabhängigkeit von Datenbank-Herstellern

3.1.2 Allgemeine Unabhängigkeit von Datenquellen

Wie in Kapitel 2 angesprochen, genügt es für eCommerce-Systeme häufig nicht, sich auf relationale Datenbanksysteme als Datenspeicher bzw. Datenquelle zu begrenzen. Leistungsfähige Systeme sollten ebenso in der Lage sein, andere Systeme wie z.B. hierarchische Datenbanken, historisch gewachsene Systeme (wie z.B. Flugabfragesysteme die eventuelle eigene Abfragesprachen aufweisen), Warenwirtschaftssysteme, oder Enterprise-Ressource-Planing-Systeme als „native“ Datenquelle bzw. Datenspeicher zu akzeptieren. An dieser Stelle sind nicht Import-Export-Möglichkeiten, z.B. durch den Austausch von ASCII- oder XML-Dateien gemeint – diese werden gesondert im Abschnitt „Schnittstellen“ behandelt - sondern die Möglichkeit, die zugrundeliegende Datenbank durch ein anderes System zu ersetzen bzw. zu ergänzen.

Derartige Funktionalitäten konnten nur bei Systemen festgestellt werden, bei denen durch eine Adapter-Schicht die Business-Logik von der Datenhaltung getrennt ist (vgl. Kap. 2), wodurch der Zugriff auf die Datenquelle aus Sicht der Business-Logik völlig transparent gehalten werden kann. Derartige Funktionalitäten konnten bei keinem der untersuchten

eCommerce-Shop-Lösungen gefunden werden. Lediglich einige der führenden Business to Business-Lösungen (vgl. z.B. Ariba [1-2]) weisen entsprechende Architekturen (Adapter-Schichten zwischen Business-Logik und Datenhaltung) auf, um einen transparenten Austausch fast beliebiger Datenquellen zu ermöglichen.

System	Unabhängigkeit von Datenquellen + gut; **o** mittelmäßig; - schlecht
eXBO	+ 3 Ebenen mit Adapter-Schichten
ARIBA (B2B)	+ 3 Ebenen mit Adapter-Schichten
Intershop	-
Openshop	-
Mercantec Softcart	-
iCat Commerce Suite	-
Microsoft Site Server Commerce Edition	-
Easymarket	-
IBM Websphere Commerce Edition	-

Abbildung 6: Allgemeine Unabhängigkeit von Datenquellen

3.1.3 Unabhängigkeit von Webserver-Herstellern

Ein Großteil der untersuchten eCommerce-Systeme unterstützt den CGI-Standard. Lediglich Microsofts eCommerce-Shop-System unterstützt den hauseigenen ISAPI-Standard. Eine klare Trennung von Business-Logik zur User-Interface-Ebene, die die Grundlage für die flexible Unterstützung verschiedenster Endgeräte bildet, ist wiederum nur in Ausnahmefällen zu beobachten. Die nachfolgende Abbildung zeigt die Untersuchungsergebnisse im einzelnen:

System	Unabhängigkeit von Webserver-Herstellern; + gut; **o** mittelmäßig; - schlecht
eXBO	+ 3 Ebenen mit Adapter-Schichten
ARIBA (B2B)	+ 3 Ebenen mit Adapter-Schichten
Intershop	**o** CGI
Openshop	**o** CGI
Mercantec Softcart	**o** CGI
iCat Commerce Suite	**o** CGI
Microsoft Site Server Commerce Edition	**o** ISAPI
Easymarket	**o** CGI
IBM Websphere Commerce Edition	**o** CGI

Abbildung 7: Unabhängigkeit von Webserver-Herstellern

3.1.4 Getrennte Entwicklung von Design und Logik

Wie in Kapitel 2 diskutiert, wird die getrennte Entwicklung von Design und Logik vor allem durch getrennte Orte von Design-Informationen und Business-Logik unterstützt. Idealerweise werden die Business-Komponenten (bzw. die darzustellenden Ausgaben, die den dynamischen Anteil einer Seite ausmachen) durch kurze stellvertretende Tags in das HTML geholt, die später (im Fall von Web-Anwendungen) durch einen Applicationserver aufgelöst werden. Ungeeignet für die getrennte Entwicklung sind Systeme, bei denen zuviel Business-Logik im HTML steht, Design-Information und Business-Logik also vermischt werden. In der nachfolgenden Abbildung sind die Untersuchungsergebnisse im einzelnen dargestellt:

System	Getrennte Entwicklung von Design und Logik: + gut; **o** mittelmäßig; - schlecht
eXBO	+
ARIBA (B2B)	+
Intershop	+
Openshop	**o**
Mercantec Softcart	+
iCat Commerce Suite	+
Microsoft Site Server Commerce Edition	- Jscript
Easymarket	- keine unabhängige Design-Entwicklung möglich
IBM Websphere Commerce Edition	**o**

Abbildung 8: Getrennte Entwicklung von Design und Logik

3.1.5 Allgemeine Unabhängigkeit von Endgeräten

Wie in Kapitel 2 angesprochen, ist die allgemeine Unabhängigkeit von User-Interface-Technologien und Endgeräten nur durch eine entsprechende Architektur mit einer Trenn- bzw. Adapter-Schicht zwischen User-Interface und Business-Logik möglich. In der Praxis wies keines der untersuchten eCommerce-Shop-Lösung eine solche Architektur auf (die z.B. eine kurzfristige Erweiterung zum Wap-Shop ermöglichen würde). Wie zuvor wurden solche Eigenschaften nur bei den führenden Business to Business-Anwendungen, wie. z.B. Ariba [1-2] festgestellt. In der nachfolgenden Abbildung werden die Ergebnissee im einzelnen aufgezeigt.

System	Unterstützung verschiedener Endgeräte: : + gut; **o** mittelmäßig; - schlecht
eXBO	+ bis in die Objektebene
ARIBA (B2B)	- nicht vorgesehen
Intershop	- nicht vorgesehen
Openshop	- nicht vorgesehen
Mercantec Softcart	- nicht vorgesehen
iCat Commerce Suite	- nicht vorgesehen
Microsoft Site Server Commerce Edition	- nicht vorgesehen
Easymarket	- nicht vorgesehen
IBM Websphere Commerce Edition	- nicht vorgesehen

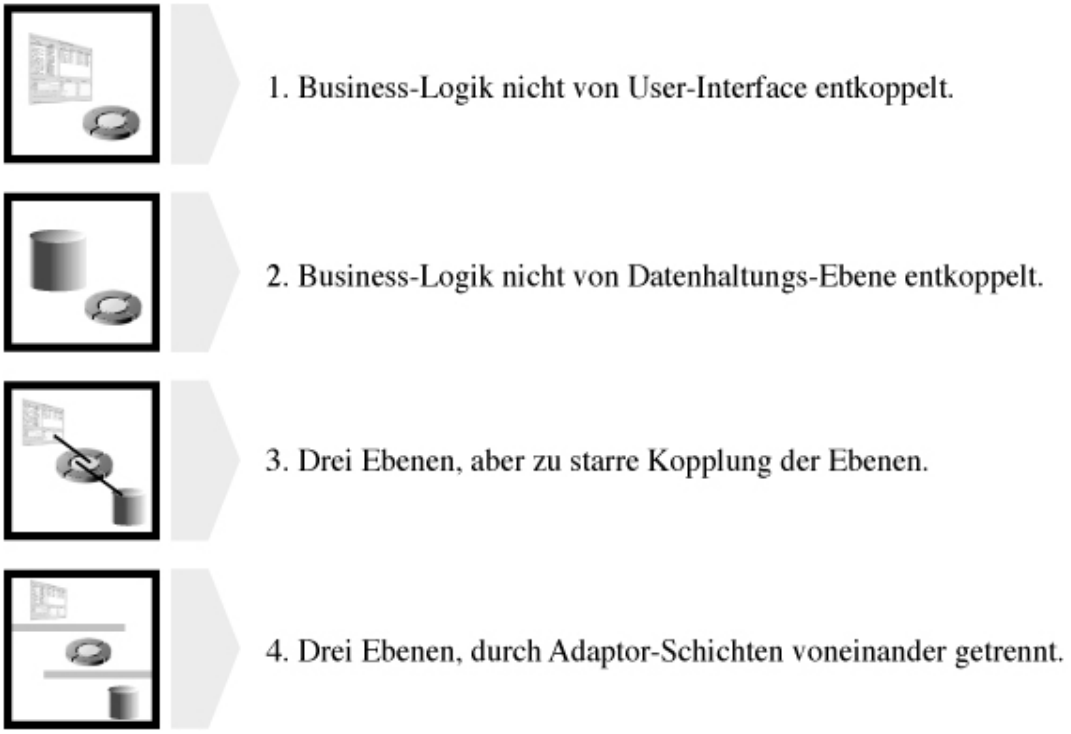
Abbildung 9: Spezialisierte Unterstützung verschiedener Endgeräte

Die sinnvolle Unterstützung von verschiedenen Endgeräten erfordert von einem eCommerce-System darüberhinaus weitere Eigenschaften. Als Beispiel soll an dieser Stelle an die unterschiedlichen Datenübertragungsraten, Bildschirmgrößen und -auflösungen von verschiedenen mobilen und stationären Endgeräten erinnert werden. Ein eCommerce-System, das den Anspruch erhebt, verschiedenste Endgeräte optimal zu unterstützen, sollte z.B. in der Lage sein, für verschiedenartige Endgeräte passende, spezialisierte Bilder verwalten und herausgeben zu können. In diesem Zusammenhang sei insbesondere auf das Kapitel 6 hingewiesen, in dem diesbezügliche Konzepte vorgeschlagen werden.

Ein Bestandteil des dort vorgestellten Konzeptes sind Funktionen, wie „-bestDetailPictureForWeb“, „-bestDetailPictureForWAP“ oder „-bestDetailPictureForPrint“, die es einem eCommerce-System (genauer gesagt sogar den einzelnen Business-Objects) ermöglichen, regelbasiert die bestmögliche Repräsentation von Bildern, Texten oder Multimediadaten zurückzugeben. Entsprechende Konzepte (vergleiche dazu Kapitel 6) wurden bisher in keiner der untersuchten B2C- oder B2B-Anwendungen realisiert.

3.1.6 Architekturen für eCommerce-Systeme

In Kapitel 2 wurden die Auswirkungen der gewählten Architektur auf Wiederverwertbarkeit, Flexibilität, Unabhängigkeit von Endgeräten und Skalierbarkeit ausführlich beschrieben.



System	Architektur gemäß obiger Legende
eXBO	4
ARIBA (B2B)	4
Intershop	3
Openshop	3
Mercantec Softcart	3
iCat Commerce Suite	2
Microsoft Site Server Commerce Edition	1, 3
Easymarket	1
IBM Websphere Commerce Edition	3, 4

Abbildung 10: Architekturen der untersuchten eCommerce-Anwendungen

Keines der untersuchten eCommerce-Shop-Systeme und nur wenige der untersuchten B2B Lösungen (vgl. [1-2], [7], [77-78]) konnten eine flexible Mehrebenenarchitektur mit konsequenter Trennung der verschiedenen Schichten aufweisen. In der Abbildung 10 werden die Gesamtarchitekturen der untersuchten Systeme gegenübergestellt.

3.1.7 Integrationsfähigkeit / Schnittstellen

Wie in Kapitel 2 erörtert, wird die Integrationsfähigkeit in vorhandenen und zukünftigen IT-Strukturen von mehreren Faktoren beeinflußt. Die wesentlichen Faktoren sind die Architektur des Gesamtsystems (die in dem vorangegangenen Abschnitt aufgezeigt wurden), die Unterstützung von Standards (vergleiche dazu die Abschnitte 2.7 und die diesbezüglichen Untersuchungsergebnisse in Abschnitt 3.1.12) und das Vorhandensein von Schnittstellen. In diesem Abschnitt sollen die vorhandenen Schnittstellen betrachtet werden. Die vorangegangene Betrachtung der Architektur ermöglicht die Unterscheidung von Schnittstellen auf User-Interface-Ebene, Business-Logik-Ebene und Datenhaltungs-Ebene. Die Einzelergebnisse der Untersuchungen sind den folgenden Tabellen zu entnehmen.

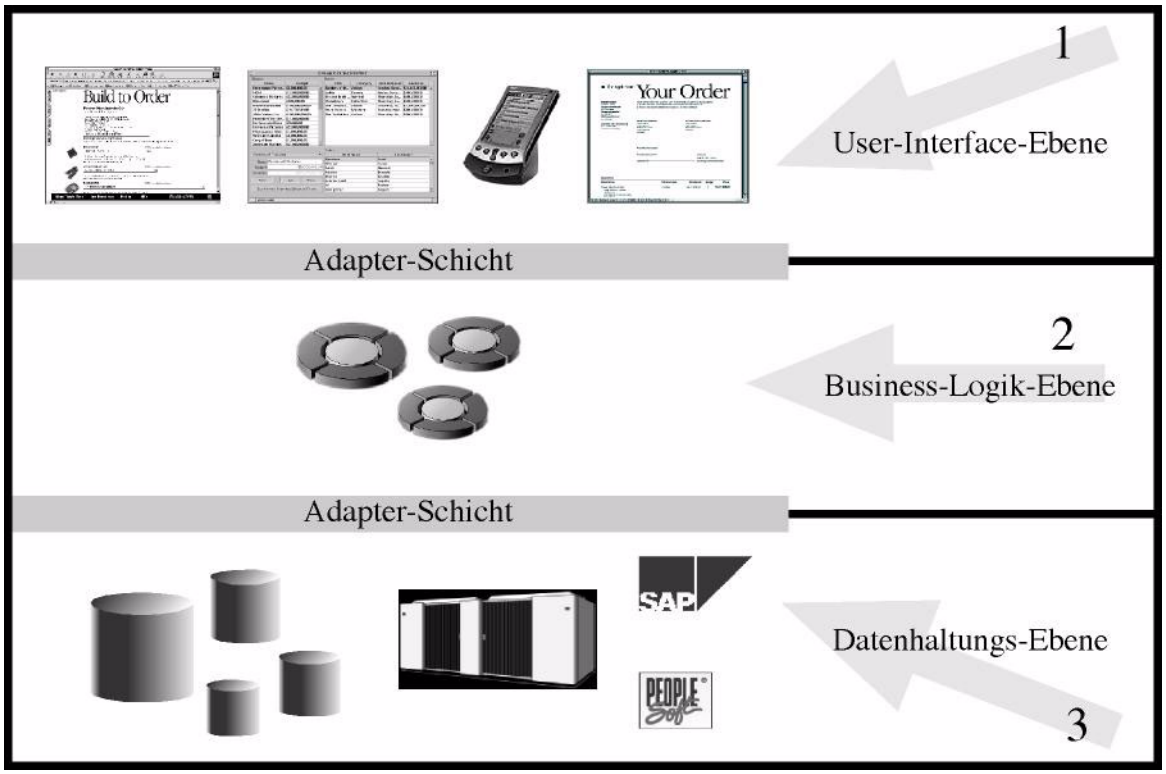


Abbildung 11: Die verschiedenen Ebenen und mögliche Schnittstellen

System	UI-Ebene (1)	Objekt-Ebene (2)	DB-Ebene (3)
eXBO	+	+	+
ARIBA (B2B)	+	+	+
Intershop	**o**	-	**o**
Openshop	**o**	-	+
Mercantec Softcart	**o**	-	**o**
iCat Commerce Suite	**o**	-	**o**
Microsoft Site Server Commerce Edition	+	**o**	+
Easymarket	-	-	**o**

System	UI-Ebene (1)	Objekt-Ebene (2)	DB-Ebene (3)
IBM Websphere Commerce Edition	+	+	+

Abbildung 12: Übersicht der vorhandenen Schnittstellen unter Beachtung der verschiedenen Ebenen

Zusammenfassend läßt sich bemerken, daß die einfachste aller Schnittstellen – der ASCII-Daten-Import – am weitesten verbreitet ist und fast von allen Systemen unterstützt wird. Schnittstellen auf der Objekt-Ebene bzw. der Business-Logik-Ebene sind selten. Schnittstellen, die eine „Direct-Action“-Funktionalität (parametrisierbare Einsprungmöglichkeit) für externe HTML-Aufrufe, die es z.B. ermöglichen, normale statische HTML-Seiten mit einer Anwendung zu verknüpfen) sind ebenfalls nur selten vorhanden.

3.1.8 Profiling

Obwohl das User-Profiling als eines der bedeutendsten eCommerce-Möglichkeiten gepredigt wird [1-4], [12], [14-15], [49], konnten bei den untersuchten Systemen keine, oder nur minimale Ansätze von User-Profiling entdeckt werden. Die Unterstützung neuester Standards, wie z.B. des P3P-Standards des W3C-Konsortiums ist ebenfalls durchgängig noch nicht vorhanden.
Detaillierte Information bezüglich der Profiling-Möglichkeiten der Einzelsysteme sind in: [77-78] zu finden.

3.1.9 Session-Management

In Kapitel 2 wurden die verschiedenen Vor- und Nachteile der unterschiedlichen Arten des Zustands-Managements bzw. „Session-Managements“ beschrieben. In der nachfolgenden Abbildung werden die gefundenen Untersuchungsergebnisse wiedergegeben.

System	Bewertung des Session-Managements
eXBO	+ Serverseitiges Session-Management
ARIBA (B2B)	+ Serverseitiges Session-Management
Intershop	+ Serverseitiges Session-Management
Openshop	+ Serverseitiges Session-Management
Mercantec Softcart	+ Serverseitiges Session-Management
iCat Commerce Suite	- nur Cookies
Microsoft Site Server Commerce Edition	+ Serverseitiges Session-Management
Easymarket	+ Serverseitiges Session-Management
IBM Websphere Commerce Edition	+ Serverseitiges Session-Management

Abbildung 13: Darstellung der Untersuchungsergebnisse „Session-Management“

Zusammenfassend läßt sich anmerken, daß nur die einfachsten „Low-Budget“-Anwendungen ausschließlich Cookies für die Session-Verwaltung verwenden. Anspruchsvollere B2C- und selbstverständlich fast alle B2B-Anwendungen besitzen eine eigene, serverseitige Session-Verwaltung [41], [77-78], [138-139].

3.1.10 Applets-Technologies

Die zusätzliche Unterstützung von Java-Applets (neben HTML) wird von keinem der untersuchten Standardsysteme unterstützt. Wie der nachfolgenden Übersicht zu entnehmen ist sind die, mit dem Framework erstellten eCommerce-Systeme dahingehend erweiterbar, neben HTML auch Java-Applets (und entsprechende Mischformen) zu unterstützen. Die Anwendungsentwicklung mit Hilfe von Java-Applets, die noch vor wenigen Jahren als die zukünftige Entwicklungsform galt, scheint durch die Verwendung von Applicationservern fast vollständig abgelöst zu sein. In Zusammenhang mit den neuesten Entwicklungen im Bereich der UMTS-Endgeräte könnte aber die Applet-Technologie (oder eine neue, ähnliche Variante) erneut in den Mittelpunkt des Interesses rücken.

System	Unterstützung von Applet-Technologie
eXBO	+
ARIBA (B2B)	+
Intershop	-
Openshop	-
Mercantec Softcart	-
iCat Commerce Suite	-
Microsoft Site Server Commerce Edition	**o**
Easymarket	**o**
IBM Websphere Commerce Edition	+

Abbildung 14: Unterstützung von Applet-Technologie

3.1.11 Unterstützung verschiedener Hardware- und Betriebssystem-Plattformen

Bezüglich der Unterstützung verschiedener Betriebssystem- und Hardware-Plattformen konnten im einzelnen folgende Untersuchungsergebnisse festgestellt werden:

System	Bewertung der Plattform-Unterstützung
eXBO	+ Apple, Win, Unix
ARIBA (B2B)	+ Apple, Win, Unix
Intershop	+ Win, Unix
Openshop	+ Win, Unix
Mercantec Softcart	+ Win, Unix
iCat Commerce Suite	+ Win, Unix

System	Bewertung der Plattform-Unterstützung
Microsoft Site Server Commerce Edition	- nur Windows
Easymarket	- nur Windows
IBM Websphere Commerce Edition	+ Win, Unix

Abbildung 15: Unterstützung verschiedener Hardware- und Betriebssystem-Plattformen

3.1.12 Sicherheit

Die Untersuchungen zeigten, daß das SSL-Protokoll zu den am weitesten unterstützten Sicherheitsprotokollen gehört. Anzumerken sei an dieser Stelle aber, daß die SSL-Verschlüsselung nur bedingt ein Teil der eigentlichen eCommerce-Anwendung darstellt. Die Verschlüsselung findet dabei bei der Übertragung von Webserver zu Webserver statt und wird im wesentlichen von den Webservern selbst durchgeführt. Diese Webserver-Erweiterungen sind meist für alle gängigen Betriebssysteme in bereits compilierter Form frei verfügbar. Die im einzelnen, gefundenen Unterstützungen für Sicherheitsstandards werden in der nachfolgenden Übersicht dargestellt. Systemeigenschaften, wie z.B. Erweiterbarkeit und Flexibilität (vergleiche Abschnitt 3.1.18), die es ermöglichen ein System im Nachhinein zu erweitern um verschiedene Sicherheitssysteme integrieren zu können, werden durch ein + gekennzeichnet:

System	Unterstützung von Sicherheitssystemen
eXBO	SSL, +
ARIBA (B2B)	SSL, +
Intershop	SSL
Openshop	SSL
Mercantec Softcart	SSL, PGP
iCat Commerce Suite	SSL

System	Unterstützung von Sicherheitssystemen
Microsoft Site Server Commerce Edition	SSL, Zertifikate, SET
Easymarket	-
IBM Websphere Commerce Edition	SSL, SET

Abbildung 16: Unterstützung von Sicherheitssystemen

3.1.13 Elektronisches Bezahlen

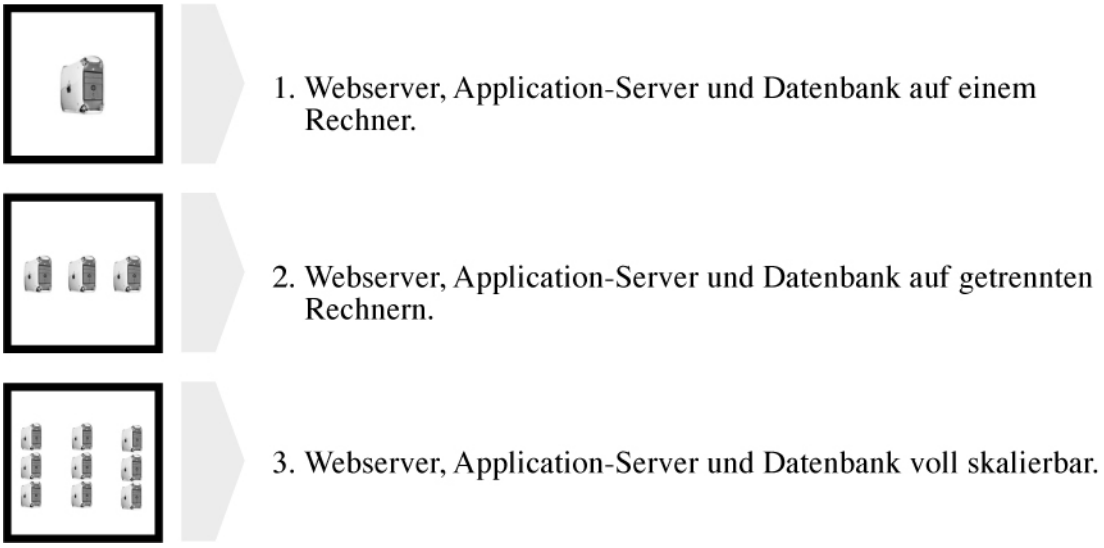
Obwohl sich bisher keines der Internet-Zahlungssysteme [70-76] durchsetzen konnte (nur weit unter ein Promille der getätigten Einkäufe werden zur Zeit per Internet-Zahlungsmethode abgerechnet), ist deren Unterstützung – wahrscheinlich aus Marketinggründen – relativ weit verbreitet. Die Untersuchungen ergaben folgende Ergebnisse:

System	Integrierte elektronische Bezahlsysteme
eXBO	**o** keine vorhanden, aber erweiterbar
ARIBA (B2B)	+ flexibel erweiterbar
Intershop	+ mehrere vorhanden
Openshop	+ mehrere vorhanden
Mercantec Softcart	+ mehrere vorhanden
iCat Commerce Suite	+ mehrere vorhanden
Microsoft Site Server Commerce Edition	+ mehrere vorhanden
Easymarket	- keine vorhanden
IBM Websphere Commerce Edition	**o** wenig vorhanden, aber erweiterbar

Abbildung 17: Integrierte elektronische Bezahlsysteme

3.1.14 Performance und Skalierung

Im Rahmen der durchgeführten Untersuchungen konnten leider keine Performanz-Tests durchgeführt werden. Auch in der Literatur sind kaum Hinweise zu solchen Untersuchungen zu finden. Eine der wenigen Untersuchungen mit ernstzunehmenden Performance-Messungen, die Untersuchungen des Französischen Instituts SQLI [153-154] bezogen sich allgemein auf Applicationserver und nicht auf einzelne eCommerce-Systeme. Wie in Kapitel 2 angesprochen, ist die Performanz von vielen Faktoren abhängig: Applicationserver, Qualität der Programmierung, Geschwindigkeit der Hardware, Performanz aller beteiligten Einzelkomponenten, wie Datenbank-Server, Webserver, Scheduling-System, Automatic-Failover und Recovery-Systeme, oder der Möglichkeit der Verteilung auf verschiedene Rechner. Letzteres – die Möglichkeit die Last auf verschiedene Rechner zu verteilen – ist besonders bei sehr großen bzw. stark frequentierten Systemen von besonderer Wichtigkeit. Die nachfolgende Übersicht gibt einen Überblick über entsprechende Möglichkeiten der untersuchten Systeme:



System	Möglichkeiten der Lastverteilung auf verschiedene Rechner
eXBO	1, 2, 3
ARIBA (B2B)	1, 2, 3
Intershop	1, 2

System	Möglichkeiten der Lastverteilung auf verschiedene Rechner
Openshop	1, 2
Mercantec Softcart	1
iCat Commerce Suite	1
Microsoft Site Server Commerce Edition	1, 2
Easymarket	1
IBM Websphere Commerce Edition	1, 2, 3

Abbildung 18: Möglichkeiten der Lastverteilung auf verschiedene Rechner

3.1.15 Funktionsumfang und Benutzerfreundlichkeit

Sehr zum Erstaunen aller beteiligten, kristallisierte sich im Laufe der Untersuchungen heraus, daß nicht einmal alle „scheinbar“ marktführenden eCommerce-Shop-Anwendungen über alle als grundlegend anzunehmenden Funktionen verfügen.

So ist es bei einigen Anwendungen z.B. nicht oder nur bedingt möglich, Warengruppen in Unterwarengruppen zu gliedern (vgl. z.B. [77-78]). Weitere Ergebnisse sind z.B.:

- eindimensionale Unterkategorien bei Microsoft Site Server (vgl. z.B. [77-78], [48]),
- zweidimensionale Kategorien bei Easymarket (vgl. [77-78]) und
- eine Ebene bei Mercantec-Softcard (vgl. [77-78], [88-89]).
- „Ein weiterer Schnitzer ist die Tatsache, daß Artikel nicht in mehreren verschiedenen Warengruppen gleichzeitig gelistet werden können“ (aus der Untersuchung zu Softcard (vgl. [77-78])).

Außerdem scheint es bei vielen Systemen sehr zweifelhaft zu sein, ob die Benutzerfreundlichkeit – vor allem im Bereich des Backoffice – ausreicht, um die tägliche, während des Betriebs anfallende Arbeit effizient bewältigen zu können. Funktionen wie Batch-Processing („Weiterblättern“ von Blöcken, von z.B. jeweils 50 Artikeln) oder Such-

und Sortierfunktionen im Backoffice um einen bestimmten Artikel anwählen zu können, sind eher die Seltenheit. Bei einigen Produkten (z.B. Microsoft-Site-Server [77-78], [48]) muß sogar die Datenbank-ID der entsprechenden Warengruppe vorgemerkt werden, um einen Artikel später in diese Warengruppe einordnen zu können.

Bedenkt man, daß die Artikelanzahl in einem Online-Shop schnell die 10.000 überschreiten kann, so ist eine z.B. fest nach Datenbank-ID sortierte Artikelliste mit 10.000 oder mehr Einträgen ohne jegliche Such- oder Sortierfunktion ein klares Defizit.

In der nachfolgenden Abbildung 20 wird stellvertretend für viele andere Systeme die Auflistung der Warengruppen im Microsoft-Site-Server dargestellt. Sie weist keine Sortiermöglichkeit auf, keine Suchmöglichkeit und kein Batch-Processing. Auch eine grafische Strukturierung von Unter- und Oberwarengruppen ist nicht vorhanden.

Die nachfolgende Abbildung 19 zeigt ein weiteres Mal den Backoffice des Microsoft-Site-Servers (Produktpreis ca. 5000 Euro). Diesmal die Übersichtsseite der Artikel. Ebenfalls ist keine Sortiermöglichkeit und keine Suchmöglichkeit vorhanden. Dafür aber ein Batch-Processing (weiterblättern in 15er Blöcken). Dieses Batch-Processing ist aber nicht änderbar (um bei 10.000 Artikeln den Artikel mit der ID 9500 auswählen zu können, müßte man vorher also 633 mal den „Weiter“-Knopf betätigen). Abbildung 21 stellt diesem die entsprechenden User-Interface-Komponenten des eCommerce-Systems „TXT“ (siehe Kapitel 8 „Realisierte Anwendungen“) gegenüber (mit (1) freiwählbarem Batch-Processing, (3) beliebiger Sortierung durch „Klick“ auf die Spaltenüberschrift und (2) integrierter Suchfunktionalität (vergleiche ebenfalls Kapitel 5 und 8). Eine ausführliche Gegenüberstellung (inklusive dem Vergleich mit den eCommerce-Systemen, die mit den vorgestellten Framework umgesetzt wurden, kann [177] entnommen werden.

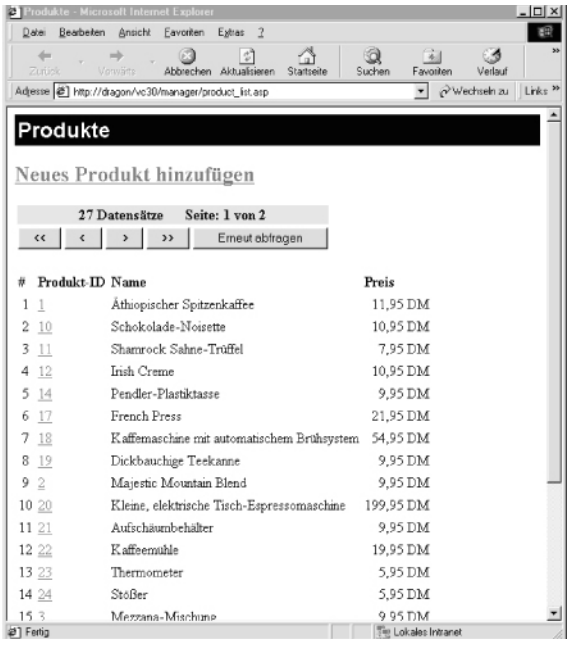


Abbildung 19: Bearbeitung von Artikeln in der Anwendung Microsoft Site Server Commerce Edition

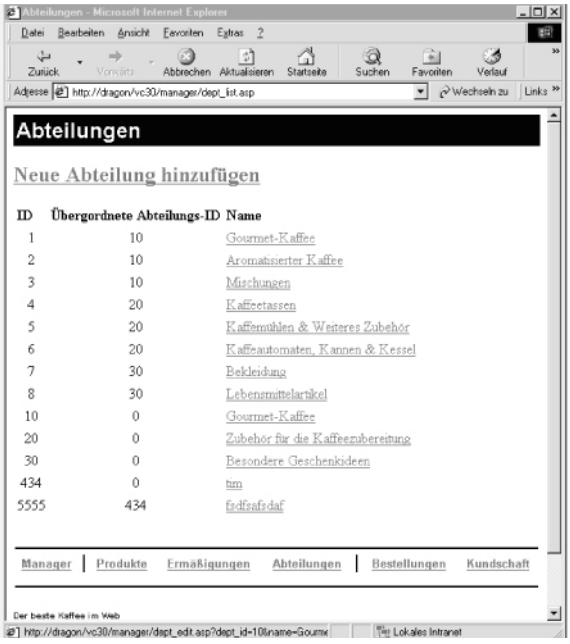


Abbildung 20: Bearbeitung von Warengruppen in der Anwendung Microsoft Site Server Commerce Edition

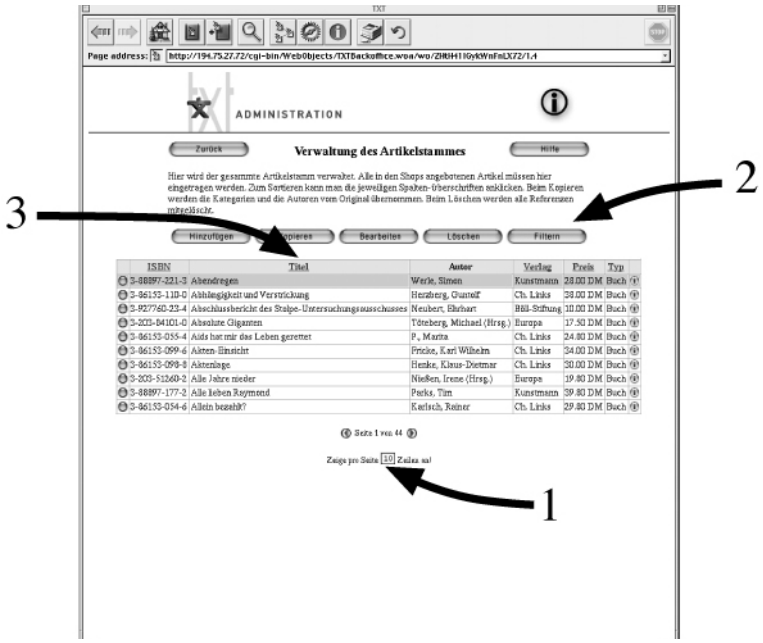


Abbildung 21: Bearbeitung von Artikeln in der Anwendung TXT

Die verschiedenen vorhandenen bzw. nicht vorhandenen Funktionen und die Benutzerfreundlichkeit bzw. Bedienbarkeit einzelner Systeme gegeneinander abzuwägen und zu beurteilen fällt schwer und führt unweigerlich zu subjektiven Einschätzungen. Um die Eindrücke der Untersuchungen dennoch vermitteln zu können, wird in der nachfolgenden Grafik versucht, einen groben Überblick bezüglich „Funktionsumfang und Benutzerfreundlichkeit“ zu geben:

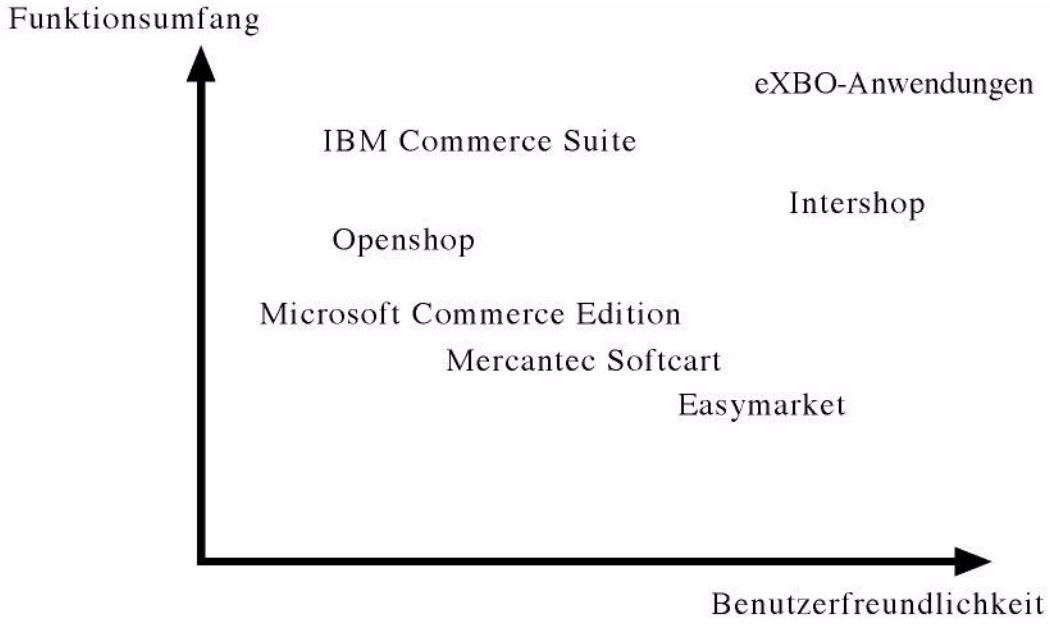


Abbildung 22: Abschätzung von Benutzerfreundlichkeit und Funktionsumfang

Der Grafik ist zu entnehmen, daß die Anwendung Intershop den Konkurrenzprodukten im Shop-Bereich noch immer um einiges voraus ist. Wenngleich eCommerce-Systeme, wie z.B. der Prototyp TXT, der mit dem in dieser Arbeit vorgestellten Framework konstruiert wurde, an vielen Stellen mehr Funktionalität aufweisen (z.B. bei den Such- und Filterfunktionen im Backoffice) wurde die Benutzerfreundlichkeit des Intershop-Systems gleich hoch eingestuft. Dies hat seine Ursache in der angerechneten Erfahrung aus dem mehrjährigen Betrieb, den Intershop sicherlich in die Anwendung hat einbringen können. Ausführlichere Betrachtungen und Vergleiche zwischen den einzelnen Systemen sind [77-78] und [177] zu entnehmen.

3.1.16 Wiederverwendbarkeit und Objektorientierung

Im Bereich der untersuchten B2C-Shop-Systeme konnte nur ein System (vgl. IBM-Websphere) gefunden werden, welches objektorientierte Grundlagen aufweist und ebenso erstellt wurde. Einige Systeme, wie z.B. Microsoft Site Server Commerce Edition versuchen mit nachkaufbaren Komponenten, die mit Hilfe eines „Process-Pipeline-Editors“ eingefügt werden können, objektorientierte bzw. komponentenorientierte Konzepte nachzuahmen (vgl. [77-78]).

Zusammenfassend läßt sich beurteilen, daß die Vorzüge der objektorientierten Programmierung im Bereich der B2C-Shop-Anwendungen (im Gegensatz zum B2B-Bereich, vgl. [1-2]) nur selten genutzt werden. Die Mehrzahl der existierenden Systeme sind einfachste, scriptbasierte Anwendungen, die entsprechend unflexibel und schlecht erweiterbar sind. Die Untersuchungsergebnisse im einzelnen sind in der nachfolgenden Abbildung wiedergegeben:

System	Einsatz von objektorientierten Konzepten
eXBO	+ durchgängig objektorientiert
ARIBA (B2B)	+ durchgängig objektorientiert
Intershop	-
Openshop	-
Mercantec Softcart	-
iCat Commerce Suite	-
Microsoft Site Server Commerce Edition	-
Easymarket	-
IBM Websphere Commerce Edition	+ objektorientiert

Abbildung 23: Einsatz von objektorientierten Konzepten

3.1.17 Erweiterbarkeit und Flexibilität

Wie in den vorangegangenen Kapiteln mehrfach erläutert, sind Erweiterbarkeit und Flexibilität besonders wichtige Faktoren für heutige und zukünftige eCommerce-Systeme. In Kapitel 2 „Grundlegende Überlegungen beim Aufbau von eCommerce-Systemen“ wurde aufgezeigt, daß die Möglichkeit ein eCommerce-System beliebig erweitern oder neuen Gegebenheiten flexibel anpassen zu können, von vielen Faktoren abhängt. Besonders hervorzuheben seien an dieser Stelle noch einmal die architekturellen Grundentscheidungen (drei Ebenen mit konsequenter Trennung der Schichten und der Einsatz von objektorientierten

und komponentenbasierten Konzepten). In der nachstehenden Tabelle wird versucht, eine zusammenfassende Bewertung bezüglich „Erweiterbarkeit und Flexibilität“ der untersuchten eCommerce-Systeme vorzunehmen:

System	Einschätzungen bezüglich Erweiterbarkeit und Flexibilität
eXBO	+ objektorientiert und Trennung der Ebenen durch Adapter-Schichten
ARIBA (B2B)	+ objektorientiert und Trennung der Ebenen durch Adapter-Schichten
Intershop	**o** eigene „Tag-Language“ ermöglicht kleine Variationen
Openshop	**o** eigene „Tag-Language“ ermöglicht kleine Variationen
Mercantec Softcart	**o** eigene „Tag-Language“ ermöglicht kleine Variationen
iCat Commerce Suite	- wenig Erweiterungsmöglichkeiten
Microsoft Site Server Commerce Edition	**o** Pipeline Editor und nachkaufbare Module
Easymarket	- abgeschlossenes System
IBM Websphere Commerce Edition	+ objektorientierter Ansatz

Abbildung 24: Einschätzungen bezüglich Erweiterbarkeit und Flexibilität

3.1.18 Zusammenfassende Darstellung

In der nachfolgenden Abbildung werden die gefundenen Einzelergebnisse noch einmal zusammenfassend dargestellt:

+ gut o mittel - weniger gut	eXBO	Ariba	Intershop	Opemshop	Mercantec	iCat	Microsoft	Easymarket	IBM Commerce
3.1.1 Unabhängigkeit von Datenbanken	+	+	-	o	-	o	o	-	-
3.1.2 Allgemeine Unabhängigkeit von Datenquellen	+	+	-	-	-	-	-	-	-
3.1.3 Unabhängigkeit von WebServern	+	+	o	o	o	o	-	o	o
3.1.4 Getrennte Entwicklung von Design und Logik	+	+	+	o	+	+	-	-	o
3.1.5 Unabhängigkeit von Endgeraäten	+	-	-	-	-	-	-	-	-
3.1.6 System-Architektur	+	+	o	o	o	o	o	o	+
3.1.7 Schnittstellen	+	+	o	+	o	o	o	o	+
3.1.8 Profiling	o	o	o	o	o	o	o	o	o
3.1.9 Session-Management	+	+	+	+	+	-	+	+	+
3.1.10 Applets	+	+	-	-	-	-	o	o	+
3.1.11 Hardware- und Betriebssystem Unterstützung	+	+	+	+	+	+	-	-	+
3.1.12 Sicherheit	+	+	+	+	+	+	+	-	+
3.1.13 Elektronisches Bezahlen	o	+	+	+	+	+	+	-	o
3.1.14 Performance und Skalierung	+	+	o	o	-	-	o	-	+

+ gut o mittel - weniger gut	eXBO	Ariba	Intershop	Opemshop	Mercantec	iCat	Microsoft	Easymarket	IBM Commerce
3.1.15 Funktionsumfang und Benutzer-freundlichkeit	+	+	+	+	o	o	o	-	+
3.1.16 Wiederverwendbarkeit und Objektorientierung	+	+	-	-	-	-	-	-	+
3.1.17 Erweiterbarkeit und Flexibilität	+	+	o	o	o	-	o	-	+

Abbildung 25: Zusammenfassende Darstellung der Untersuchungsergebnisse

3.2 Standardisierungsbemühungen, Frameworks und Entwicklungsumgebungen

In diesem zweiten Teil des dritten Kapitels werden Standardisierungsbemühungen, Frameworks und Entwicklungsumgebungen in Auszügen vorgestellt, die sich im Bereich der eCommerce-Lösungen hervorgetan haben. Bedingt durch die große Anzahl von Standardisierungsbemühungen, und der großen Anzahl von Frameworks und Entwicklungsumgebungen kann im Rahmen dieser Arbeit kein umfassesnder Überblick über alle Systeme gegeben werden. Im nachfolgenden werden, diejenigen Standardisierungsbemühungen, Frameworks und Entwicklungsumgebungen vorgestellt, die die Enstehung dieser Arbeit am stärksten beeinflusst haben.

3.2.1 OMG-Geschäftsobjekte

Die Object Management Group (OMG) hofft durch ihre Bemühungen im Bereich der Business-Objects, die Entwicklung von Geschäftsanwendungen wesentlich zu verbessern. Diese Verbesserungen beruhen hauptsächlich auf zwei Punkten.

Erstens verspricht man sich durch die Definition der Business-Objects eine Annäherung zwischen der betriebswirtschaftlich-anwendungsbezogenen Sichtweise und der technisch-implementierungsbezogenen Sicht der Entwickler. Die schnelle und effiziente Entwicklung von spezialisierten Geschäftsanwendungen erfordert eine enge Zusammenarbeit von Software-Ingenieuren und Experten aus den jeweiligen Geschäftsbereichen. Für die schnelle und korrekte Umsetzung von Anforderungen aus der Geschäftswelt in einsatzfähige Softwaresysteme benötigt man einen Weg, um geschäftliche Konzepte, Prozesse, Elemente und Zusammenhänge so zu beschreiben, daß sie von beiden Welten gleich gut und eindeutig verstanden werden. Dies ist eine der Hauptintentionen der Business-Objects.

Der zweite Punkt ist die Bereitstellung von wiederverwendbaren, getesteten, erprobten und standardisierten Komponenten, mit denen es möglich ist, Geschäftsanwendungen mit wesentlich geringeren Kosten und mit wesentlich kürzeren Entwicklungszyklen zu erstellen. Zukünftige Geschäftsanwendungen sollen außerdem die Fähigkeit besitzen, auch komplexere

Aufgaben, wie zum Beispiel die firmenübergreifende Kooperation durch Business-Objects, zu bewältigen.

Was aber sind Business-Objects?

Unter Business-Objects versteht man Objekte, die reale Dinge aus dem Geschäftsleben repräsentieren, zum Beispiel einen Kunden, eine Rechnung oder einen Auftrag.
Die OMG definiert ein Business-Object sinngemäß als "die Repräsentation eines Objektes, das in einem Unternehmensbereich aktiv ist. Es umfaßt seinen Namen und seine Definition innerhalb des Unternehmens, Attribute, Verhalten, Beziehungen, Regeln, Vorgehensweisen und Randbedingungen. Ein Business-Object kann zum Beispiel eine Person, einen Ort, ein Ereignis, ein Geschäftsprozeß oder ein Konzept repräsentieren“.

Diese Definition läßt erkennen, daß der Begriff der „Business-Objects“ sehr allgemein gehalten wird. Um eine bessere Kategorisierung von den verschiedenen Business-Object-Typen zu erreichen wird von Mitgliedern der BODTF (Business Object Domain Task Force) der OMG eine Klassifizierung der Business-Objects vorgeschlagen. Diese Klassifizierung, soll das Verständnis unter den Anwendungsentwicklern verbessern, die in Zukunft per Katalogauswahl auf vorhandene Komponenten zugreifen sollen. Die OMG Business-Objects werden dabei nach drei Typen unterschieden: Entity-Business-Object (Entität bzw. Gegenständlichkeit), Process-Business-Object (Ablauf) und Event-Business-Object (Ereignis).

Beispielhafte Vertreter der ersten Kategorie von Business-Objects (Entity Business Objects) sind reale oder abstrakte Phänomene, die in Fachspezifikationen meist als Substantive auftauchen: Kunde, Bestellung, Adresse oder Vertrag. Eine weitere Hilfe für das Aufspüren von Entity-Business-Objects ist die Tatsache, daß ihre Attribute in herkömmlichen Anwendungen oft und gerne in Datenbanken gespeichert werden.

Process-Business-Objects sind im Gegensatz dazu stets zeitbehaftete Geschehnisse bzw. Geschäftsabläufe, wie zum Beispiel Verkaufen, Beschaffen, Unterstützen, Abrechnen oder Liefern.

Event-Business-Objects sind die Abbildung von Ereignissen. Bei diesen kann es sich um Ereignisse handeln, die zeitabhängig sind (wie zum Beispiel „Steuererklärung anfertigen, weil Kalendermonat vorbei ist“), um Ereignisse die vom System selbst herbeigeführt werden (wie zum Beispiel „Mindestlagermenge unterschritten“ oder „Kontobuchung ausgeführt“) oder auch um Ereignisse die vollständig unabhängig vom System sind (wie zum Beispiel „Tank leer“ oder „Apple-Aktie > 100 $“).

Die Vorgehensweise der OMG ist dabei so geregelt, daß man versucht zunächst domain-spezifische Business-Objects zu finden. Durch die Etablierung derartiger Objekte mit denen grundlegende und typische Aufgabenstellungen in den jeweiligen Branchen abgedeckt werden können, soll vor allem die Kooperation verschiedener Firmen einer Branche gefördert werden. Die domain-spezifischen Business-Objects können ihrerseits noch in Unternehmens- bzw. Enterprise-Spezifische-Business-Objects verfeinert werden. Für einige Business-Objects ist es notwendig, regional bzw. geopolitisch bedingte Variationen einzuführen. Beispiele für diese Variationen sind Business-Objects, die Steuersätze oder Währungen repräsentieren. Nachdem domain-spezifische Business-Objects identifiziert werden können, wird versucht, durch den Gedankenaustausch mit den anderen OMG Domain Task Forces eine größere Übereinstimmung zu finden, um so ein allgemeingültiges und somit am besten wiederverwendbares „Common Business Object“ (CBO) zu erhalten.

Da der OMG Interface Definition Language (IDL) die Ausdrucksmittel fehlen, um eine ausreichende semantische Beschreibung der Schnittstellen und Eigenschaften von komplexen Business-Objects zu beschreiben, wird innerhalb der Business Object Domain Task Force (BODTF) der OMG versucht, die Entwicklung und Standardisierung der sogenannten Component Definition Language (CDL) voranzutreiben. Die Component Definition Language ermöglicht eine semantische Spezifikation von Objekten, deren Beziehungen zu anderen Komponenten und eine formale Beschreibung des Verhaltens, inklusive etwaiger Vor- und Nachbedingungen. Die CDL ist eine Erweiterung der IDL.

In den Diskussionen der BODTF kann man zwei verschiedene Stadien von Business-Objects unterscheiden. In einem ersten Stadium wird versucht Business-Objects aus einem Business-Modell abzuleiten. Die dabei identifizierten Business-Objects werden textuell (zum Beispiel mit der Component Definition Language) oder graphisch (zum Beispiel mit der Unified Modelling Language (UML)) beschrieben. In diesem Stadium existieren die Business-Objects unabhängig von Informationsverarbeitungssystemen, Anwendungen, Software-Design oder Programmcode. Außerdem wird nicht davon ausgegangen, daß jedes Business-Object in diesem Stadium in einer späteren Anwendung als solches implementiert werden muß.

Business-Objects im zweiten Stadium sind „reale“ Software-Objekte mit allen „normalen“ Objekteigenschaften. Zusätzlich können die Business-Objects Eigenschaften haben, die über die Beschreibung des OMG-Objekt-Modells hinausgehen. Diese Eigenschaften können zum Beispiel durch die CDL oder die UML beschrieben werden:

- Behavior: Verhaltensbeschreibungen, wie zum Beispiel: „immer wenn ein Jahreswechsel vollzogen wird, wird veranlaßt, daß...“,

- Business rules: exakte Beschreibung von Geschäftsregeln die vom Business-Object eingehalten werden müssen,
- Business identity: eindeutige Identifikation jeder Instanz des Business-Objectss, um es der entsprechenden Entität in der realen Welt zuordnen zu können,
- Integrity of instances and inter-instance relationships: um die Vollständigkeit einer Instanz zu garantieren; im Falle eines Vertrages darf zum Beispiel nicht die Hälfte fehlen,
- Persistence: Business-Objects sollen solange existieren, wie die „realen“ Phänomene, die sie abbilden,
- Security: ein Business-Object „Rechnung“ darf zum Beispiel nicht unautorisiert verändert werden,
- Interoperability: Business-Objects verschiedener Hersteller sollen sich untereinander verständigen können,
- Transactability: die Vollständigkeit von Transaktionen soll garantiert werden können; im Falle eines Fehlers oder Abbruchs soll ein Rollback durchführbar sein.

Die ursprüngliche Vision der OMG ging davon aus, daß Entitätstypen wie Kunde, Artikel, Verkauf oder Rechnung relativ schnell als Common Business Objects (CBOs) definierbar sind. Inzwischen ist man erheblich zurückhaltender, zeigte sich doch in der Diskussion, daß selbst bei diesen – auf den ersten Blick – einfach erscheinenden Business-Objects große Verständigungsschwierigkeiten bestehen. Mit erheblich reduziertem Anspruch wird nunmehr in einem Bottom-Up-Ansatz versucht, nach und nach wenigstens eine Normierung elementarer Objekttypen (wie zum Beispiel Decimal oder Time) zu erreichen. Aktuelle Arbeiten der BODTF betreffen die Umrechnung, Verwaltung und Darstellung von Währungs- und Datumseinheiten, Objekte zur Repräsentation von Geschäftsadressen sowie Kalenderfunktionen.

Das Konzept der Business-Objects ist nach Einschätzung der Experten ein Schritt in die richtige Richtung. Sollte es gelingen, auch nur einen Teil der Konzepte und grundlegenden Ideen umzusetzen, könnte dies einen evolutionären Schritt für die gesamte Software-Entwicklung, vor allem aber für den Bereich der Entwicklung von Geschäftsanwendungen bedeuten.

Zusammenfassend läßt sich also sagen, daß die Konzepte der OMG Business-Objects sehr vielversprechend sind. Der gegenwärtige Stand der Entwicklung im Bereich der OMG Business-Objects erlaubt es jedoch nicht, auf eine ausreichend große Anzahl von ausgereiften, standardisierten Business-Objects zurückzugreifen, mit denen man schon heute konkrete eCommerce-Anwendungen erstellen könnte. Es ist angebracht die Entwicklung der OMG Business-Objects weiterhin zu verfolgen um schon jetzt von den aufgezeigten Lösungsvorschlägen und Ideen zu profitieren und sie in eigene Ansätze einfließen zu lassen. Weitere Informationen zum Thema Business-Objects ist [102-119] und [122-126] zu entnehmen.

Einer der Ursprünge der vorliegenden Arbeit liegt in der Beobachtung der Business Object Domain Task Force der OMG und der Beschäftigung mit den dort vorgeschlagenen Ideen und Lösungsansätzen.

3.2.2 IBM San Francisco

Als einer der großen Akteure innerhalb der Business Object Domain Task Force der OMG hat IBM parallel zu den Standardisierungsbemühungen ein eigenes Business-Object-Framework entwickelt. Dieses Java-basierte Framework wird von IBM unter dem Namen „San Francisco“ vertrieben.

Das San Francisco Framework ist in mehreren Ebenen aufgebaut, die anhand der nachfolgenden Abbildung erläutert werden sollen.

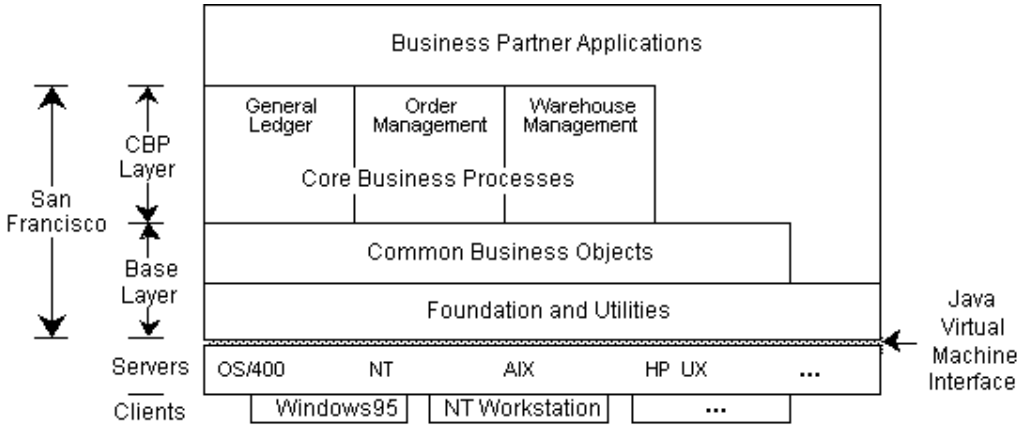


Abbildung 26: Der Aufbau des San Francisco Frameworks

Die beiden untersten Schichten (Server, Clients) gehören nicht zum eigentllichen Bestand des San Francisco Frameworks. Sie gehören zum Standard-Umfang einer jeden Java-Installation.

i) Base Layer

Die untere Ebene des San Francisco Frameworks wird als Base Layer bezeichnet. Sie besteht aus zwei Teilen: den „Foundation and Utilities“ und den „Common Business Objects“.

ii) Foundation and Utilities

Der Foundation and Utilities Layer bildet die unterste Schicht des San Francisco Frameworks. Sie enthält grundlegende Klassen, die zum Beispiel für die Kommunikation mit dem darunterliegenden Computer-Betriebssystem verantwortlich sind, die Basis-Funktionalitäten, wie das Instanziieren von Objekten ermöglichen oder auch Collections und Iteratoren, mit denen das Arbeiten mit mehreren Objekten ermöglicht wird.

Neben diesen Klassen, die typischer Weise die Basis für die Klassenhierarchie eines Frameworks darstellen, existieren im Foundation and Utilities Layer einige Hilsmittel für Entwickler. Zu ihnen gehören Werkzeuge, wie Administrations- und Konfigurations-Tools, sowie Werkzeuge zur Kontrolle von Konflikten oder zur Beobachtung des Laufzeitverhaltens von Objekten. Weitere Informationen zum Foundation and Utilities Layer sind [130-131] zu entnehmen.

iii) Common Business Objects Layer

Der Common Business Objects Layer beinhaltet die eigentlichen Business-Objects. Beispiele für die im San Francisco Projekt umgesetzten Business-Objects sind:

- Adress
- BusinessPartner
- Bank
- BankAccount
- Kalender
- Company

- Currency
- PaymentMethod

Zu den Business-Objects werden jeweils passende Darstellungs- und Bearbeitungs-komponenten angeboten. Die User-Interface-Elemente des San Francisco Frameworls sind allerdings (zumindest bisher) nicht für web-basierte Anwendungen geeignet. Alle Oberflächen existieren ausschließlich für „normale Fensteranwendungen“ und dies auch nur für Computer mit Microsoft-Windows- oder IBM-AIX-Betriebssystemen. Weitere Informationen zum Common Business Objects Layer sind [128-133] zu entnehmen.

iv) Core Business Processes Layer

Die Komponenten des Core Business Processes Layer bilden den obersten Teil des San Francisco Frameworks. Diese Ebene enthält „Rohgerüste“ für eCommerce-Anwendungen, die dann weiter verfeinert und zu individuellen Lösungen ausgebaut werden können. In den jeweiligen Konstrukten sind alle grundlegenden Funktionen enthalten, die für die jeweilige Geschäftsanwendung typisch sind. Bereits mit dem ersten Release wurde auch eine erste Core Business Process Komponente („General Ledger“ bzw. „Hauptbuchhaltung“) ausgeliefert. Weitere Core Business Process Komponenten, wie zum Beispiel „Auftragsabwicklung“ oder „Warenwirtschaft“ sollen folgen. Weitere Informationen zum Core Business Processes Layer sind [128-133] zu entnehmen.

v) Zusammenfassung

Mit San Francisco liegt erstmals ein Produkt vor, welches als komponenten-basiertes Framework für Geschäftsanwendungen bezeichnet werden kann. Der erwartete Erfolg blieb aber bisher aus. Dies hat verschiedene Ursachen, über deren Gewichtung nur spekuliert werden kann. Zu den Punkten, die diesbezüglich diskutiert werden, gehört die unverständliche Einschränkung auf fenster-basierte Oberflächen, die der allgemeinen Entwicklung hin zur computer-, standort-, und betriebssystemunabhängigen Web-Anwendung entgegenläuft. Die Begrenzung auf die Windows-Plattform, die bei vielen Unternehmen (zumindest im Bereich der „Mission Critical Applications“) nicht eingesetzt wird, trägt ebenfalls ihren Teil dazu bei. Weitere Probleme sind die unterstützenden Tools, die sich zum größten Teil noch im Beta-Status befinden und den Entwickler eher behindern als unterstützen.

Eines der größten Probleme jedoch ist eindeutig die erheblich zu langsame Ausführungsgeschwindigkeit von Anwendungen. Dies wird zum Teil auf die rein Java-basierte Implementierung geschoben, zum Teil der mangelhaften Optimierung großer Teile

des Frameworks angelastet.

Nachteilig wirkt sich sicherlich auch die Tatsache aus, daß nur zwei verschiedene Datenspeicher unterstützt werden: die hauseigene, relationale Datenbank IBM DB2 und Oracle. Neben der technischen Einschränkung auf diese beiden Systeme schließt die Tatsache, daß beide Datenbanken im oberen Preissegment liegen, einen preisgünstigen Einstieg aus.

Zusammenfassend läßt sich also feststellen, daß das bisherige Ausbleiben des Erfolges nicht unbegründet ist, und schon alleine durch die Mißachtung vieler Kriterien (die in Kapitel 2 vorgestellt wurden) erklärbar wäre:

Mißachtung des Kriteriums „Skalierbarkeit“ in beiden Gesichtspunkten: Geschwindigkeit und Preis. Die Ausführungsgeschwindigkeit von Anwendungen ist zu langsam, und durch die Einschränkung auf teure Datenbanksysteme ist ein preiswerter Einstieg nicht möglich. Weitere Krierien, die mißachtet wurden, sind: „Unabhängigkeit von Datenbankherstellern“, „Allgemeine Unabhängigkeit von Datenquellen“, „Unabhängigkeit von User-Interface-Technologien und Endgeräten“, „Unterstützung verschiedener Hardware- und Betriebssystemplattformen“ und vor allem die unverständliche Nicht-Unterstützung von Web-Anwendungen im Internet-Zeitalter. Weitere Informationen zum Thema IBM San Francisco ist [127-133] zu entnehmen.

3.2.3 WebObjects

Während einer tiefergehenden Recherche im eCommerce-Umfeld, besonders bei der Untersuchung von anspruchsvollen und erfolgreichen eCommerce-Anwendungen im World Wide Web stößt man zwangsläufig auf den Begriff WebObjects. Die eCommerce-Anwendungen von DELL, Apple, Deutsche Bank 24, Consors, das Portal der Deutschen Telekom, von Reiseveranstalter TUI, the American Stock Exchance, eTrade, dem Schweizerischen Bankenverein, mehrere B2B-Portale, oder die führenden B2B-Anwendungen (wie z.B. Ariba), um nur einige zu nennen - sie alle wurden mit Hilfe von WebObjects entwickelt.

WebObjects ist eine der bewährtesten und industriell erprobtesten objektorientierten Anwendungs- und Entwicklungsumgebungen überhaupt. Die Ursprünge der WebObjects-Technologie liegen bei der Firma Next, die sich jahrelang als technologischer Vorreiter und Pionier auf dem Gebiet der objektorientierten Programmierung auszeichnete. Nach dem Zusammenschluß der Firmen Apple und Next wurde aus der ständig weiterentwickelten NextStep-Entwicklungsumgebung schließlich WebObjects.

Die WebObjects Technologie ist zudem auf ursprünglichste Weise mit dem World Wide Web verbunden. Das World Wide Web selbst, d.h. die ersten Webserver und Webbrowser wurden von Tim Bernes Lee mit dieser Entwicklungsumgebung erstellt (siehe auch [178-179]).

„The NeXT Development Tools beneath the NeXT Frameworks was it that make it all possible“ (Zitat von Tim Bernes Lee, dem „Erfinder“ des Web [178]).

3.2.3.1 Was ist WebObjects?

WebObjects besteht aus mehreren Teilen:

- aus einem Applicationserver,
- aus leistungsfähigen Entwicklungswerkzeugen
- und aus verschiedenen Frameworks für die Entwicklung von Internet- und anderen Anwendungen.

Die vollständig objektorientierte Entwicklungsumgebung unterstützt dabei hundertprozentig eine ebensolche Anwendungsentwicklung. Als Programmiersprachen können dabei ObjectiveC, WebScript, Java oder C++ zum Einsatz kommen. Sämtliche Punkte aus Kapitel 2

150wie z.B. die strikte Trennung der Datenhaltungs-, Anwendungs-Logik- und User-Interface-Ebenen durch Adapter-Schichten, optimale Skalierbarkeit oder die Wiederverwendbarkeit von Objekten und Komponenten werden von WebObjects optimal unterstützt. Funktionalitäten, wie das für Internetanwendungen besonders wichtige Session-Management oder auch die Anbindung von relationalen Datenbanken an objektorientierte Anwendungen (einschließlich der automatischen Erzeugung von Objekten aus Datenbankinhalten), werden durch fertige, höchst leistungsfähige Framework-Komponenten zur Verfügung gestellt (siehe z.B. [136], [138-140], [149-150], [153-154]).

3.2.3.2 Frameworks

WebObjects beinhaltet mehrere verschiedene Teilframeworks. Jeder von ihnen ist für einen bestimmten Einsatzbereich spezialisiert:

i) Enterprise Objects Framework (EOF)

Die Objekte und Komponenten dieses Frameworks sind darauf spezialisiert, aus beliebigen Datenquellen, meist relationalen Datenbanken, automatisiert Objekte („Enterprise-Objects“, wie sie im „WebObjects-Sprachgebrauch“ genannt werden) oder Business-Objects zu erzeugen bzw. die erzeugten Objekte persistent zu halten. Die Objektebene und die Datenbankebene werden dabei durch Adapter-Objekte sauber voneinander getrennt. Genauere Informationen sind den diesbezüglichen Untersuchungen z.B. [150] oder [154] zu entnehmen. Weitere Informationen über das Enterprise-Objekts-Framework sind in [140] und [146-148] zu finden.

ii) Portable-Distributed-Objects (PDO)

Die Portable-Distributed-Objects (PDO) sind bei einer Verteilung von Objekten an beliebiger Stelle eines Netzwerks für eine transparente Kommunikation zuständig. PDO ist eine plattformunabhängige Technologie, die dank ihres dynamischen Objektmodells höchste Flexibilität auf Objektebene ermöglicht. PDO stellt auch die Interoperatibilität zu anderen Objektmodellen, wie z.B. RMI, DCom oder CORBA her. Weitere Informationen sind z.B. [143-147] zu entnehmen.

iii) WebObjects-Framework

Der WebObjects-Framework ist spezialisiert für den Aufbau von Internetanwendungen. Der

WebObjects-Framework ist der jüngste Framework, der natürlich noch nicht existierte als Tim Bernes Lee und sein Team am CERN „das Web erfanden“. Eine der wesentlichsten Komponenten in diesem Framework ist die flexible und höchst leistungsfähige Session- bzw. Zustandsverwaltung. Außerdem existieren zahlreiche Objekte und Komponenten für die Konstruktion von dynamischen Internetseiten. Die Wiederverwendung von Objekten und Komponenten beim Aufbau von dynamischen Internetseiten wird massiv unterstützt. Mehr Informationen zum WebObjects Framework sind in [138-142] zu finden.

iv) Foundation Framework

Der Foundation Framework beinhaltet Basisklassen für die Programmierung, wie z.B. Arrays, Timer usw. Mehr Informationen zum Foundation Framework sind in [134-137] zu finden.

v) ApplicationKit (AppKit)

Das ApplicationKit-Framework beinhaltet hauptsächlich Objekte und Komponenten für den Aufbau von „normalen“ Anwendungen. Die entwickelten Programme können mit dem entsprechenden Compiler für verschiedenste Plattformen (wie z.B. Apple, Windows, Sun oder HP) übersetzt werden. Außerdem können diese Anwendungen, (wenn in Java geschrieben), zu Applets werden, die z.B. über das Netz verschickt werden können. Weitere Informationen über das ApplicationKit-Framework, die entsprechenden Entwicklungswerkzeuge und über Java-Applets findet man in: [134-144].

3.2.3.3 Entwicklungswerkzeuge

WebObjects beinhaltet eine Reihe von höchst leistungsfähigen Entwicklungswerkzeugen, die im folgenden kurz vorgestellt werden sollen.

i) Project Builder

Der Project Builder ist das zentrale Entwicklungswerkzeug, von dem aus alle anderen Werkzeuge gestartet werden können. Er enthält einen Projekt-Browser, Quelltext-Editor, Objekt-Browser, Build- und Debugging-Unterstützung und vieles mehr.

ii) WebObjects Builder

Der WebObjects Builder ist ein Werkzeug, das den Entwickler bei der Erstellung von dynamischen Internetseiten unterstützt.

iii) EOModeler

Der EOModeler unterstützt den Entwickler bei der Verbindung von relationalen Datenbanken und objektorientierten Systemen. Er ermöglicht sowohl das automatisierte Erzeugen von Datenbanken, Tabellen und Relationen aus graphischen Objektbeschreibungen, als auch das automatische Erzeugen von Objekten aus Datenbankinhalten.

Mehr Informationen zu den höchst leistungsfähigen Entwicklungswerkzeugen, die jahrelang als Vorbild für die meisten der heutigen Java-Entwicklungswerkzeuge dienten, sind am besten der technischen Dokumentation für Entwickler [136-142] zu entnehmen.

3.2.3.4 Applicationserver

Erfolgreichste Internetanwendungen, mit denen Tagesumsätze in Höhe von mehreren Millionen US-Dollar generiert wurden, wie z.B. die Anwendung von DELL-Computer, liefen bereits mit dem WebObjects-Applicationserver lange bevor es den Begriff „Applicationserver“ oder vergleichbare Konkurrenzprodukte überhaupt gab.

In jüngster Vergangenheit traten zahlreiche Applicationserver von verschiedensten Herstellern auf den Markt. Eine der wenigen tiefgreifenden Vergleichsstudien bezüglich Applicationservern, die quantitative und qualitative Analysen beinhaltet, wurde vom Französischen Institut SQLI Engenerie angefertigt. Diese Studie beinhaltet sowohl qualitative als auch quantitative Untersuchungen, wie z.B. die Messung der Antwortzeiten unter Last. In allen Untersuchungsergebnissen spiegelte sich die führende Stellung des WebObjects-Applicationservers wieder. Weiterführendes Material, wie z.B. die SQLI-Studie [154] oder eine ähnliche Untersuchung der NASA [150] sind der Literatur zu entnehmen.

3.2.3.5 Abgrenzung zur vorliegenden Arbeit

Wie oben beschrieben, kann dem WebObjects-Framework eine vorzügliche Eignung für den Aufbau von internetbasierten Anwendungen zugeschrieben werden. Was dem Framework jedoch eindeutig fehlt sind spezialisierte Objekte und Komponenten, wie Business-Objects und Business-Object-Facilities für den gezielten Aufbau von eCommerce-Anwendungen.

Aufgrund der beschriebenen Eigenschaften sollte WebObjects aber keinesfalls als Konkurrenzprodukt zu dem in dieser Arbeit vorgestellten Framework angesehen werden. Es sollte, vor allem wegen der konsequenten Einhaltung und Unterstützung der „Grundlegenden Überlegungen beim Aufbau von internetbasierten eCommerce-Anwendungen“ (vergleiche Kapitel 2), als eine der zu bevorzugenden Entwicklungsumgebungen für den Aufbau und für die Umsetzung der Konzepte in reale Anwendungen angesehen werden.

Bei der Implementierung der Prototypen (vergleiche Kapitel 8) wurden beste Erfahrungen beim Arbeiten mit WebObjects gemacht. Insbesondere konnte festgestellt werden, daß bei der Umsetzung der in dieser Arbeit vorgestellten Konzepte mit Hilfe der WebObjects-Entwicklungsumgebung ideale Synergieeffekte genutzt werden können. Dies ermöglichte es einem Kleinstteam von Entwicklern in kürzester Zeit konkurrenzfähige, zum Teil preisgekrönte eCommerce-Systeme zu erschaffen.

Kapitel 4

Entwurfsmuster (Design Patterns) für eCommerce-Systeme

Zu den jüngsten Entwicklungen im Bereich der Objektorientierung gehört die methodische Suche nach Entwurfsmustern (englisch: „Design-Patterns“). Die Suche nach Entwurfsmustern umfaßt mehr als die Suche nach möglichst „vollkommenen“, wiederverwendbaren Klassen bzw. Komponenten. Entwurfsmuster zielen vielmehr darauf ab, gemeinsame Verhaltens- und Interaktionsmuster zu erkennen, die über die Ebene von Klassen hinausgehen.

Die Untersuchung von Mustern ist nicht auf dem Bereich der Software-Technik beschränkt. Die Bedeutung von Mustern ist schon viel früher von anderen technischen Wissenschaften erkannt worden, wie etwa der Biologie, Chemie, Physik oder der Architektur. Die Bedeutung

für die Software-Entwicklung, insbesondere bei der Konstruktion großer und komplexer Software-Systeme wurde erst zu Beginn der letzten Dekade dieses Jahrhunderts erkannt.

Zu den wesentlichen Veröffentlichungen auf dem Gebiet der Entwurfsmuster (Design-Patterns) gehören die Arbeiten von Grady Booch, Bruce Anderson, Erich Gamma, Kent Beck oder J. Vlissides [162-173], [180] um nur einige zu nennen. Grundlage vieler Arbeiten über Entwurfsmuster sind die Aufzeichnungen von Christopher Alexander, einem Gebäude-architekten aus Berkeley, der schon in den 70er Jahren eine „neue“ auf Entwurfsmustern basierende Architektur propagierte. Insbesondere sein Buch „The Timeless Way of Building“ fand großen Anklang bei vielen „Software-Architekten“ von heute und wird entsprechend häufig zitiert [162].

In seinen Untersuchungen „der Komplexität“ beobachtet er beispielsweise, daß „komplexe Systeme in den meisten Fällen aus einer kleinen Zahl von Teilsystemen bestehen, die in unterschiedlichen Verbindungen und Anordnungen organisiert sind“. Mit anderen Worten: komplexe Systeme haben gemeinsame Muster. Diese Muster können die Wiederverwendung kleiner Komponenten einschließen, wie etwa die Zellen, die sowohl in Pflanzen als auch in Tieren zu finden sind. Ebenso können auch größere Strukturen, wie z.B. das Gefäßsystem das Pflanzen und Tieren gemeinsam ist, wiederverwendet werden.

In den oben genannten Arbeiten wird beschrieben, wie ähnliche „Muster“ in gut strukturierten Software-Systemen wiederzufinden sind. Systeme, die auf den ersten Blick komplex und undurchdringbar erscheinen, werden durch das Suchen bzw. Auffinden von Mustern plötzlich verständlich und überschaubar. Hat man einmal diese grundlegenden Muster erkannt, so können auch komplexeste Systeme als bloße Anordnung von verschiedenen, aber immer wiederkehrenden Mustern betrachtet werden. Die Anzahl der dabei voneinander verschiedenen Grundmuster ist oft erstaunlich gering.

Folgendes Zitat, welches von Grady Booch in [162] zitiert wird, gibt die Einschätzung der Experten zu Entwurfsmustern wieder: „... ferner wird veranschaulicht, wie diese Muster die eingewurzelte Komplexität von Systemen per se zu vereinfachen vermögen.“

In den nächsten Abschnitten dieses Kapitels sollen diese Eigenschaften genutzt werden und anhand eines konkreten Beispiels gezeigt werden wie zunächst komplex anmutende eCommerce-Software-Systeme durch die Identifikation von Mustern zu übersichtlichen bzw. überschaubaren Systemen werden.

Zuvor wird im folgenden Abschnitt auf die unterschiedlichen Kategorien von Entwurfsmustern und ihren Bezug zur vorliegenden Arbeit eingegangen.

4.1 Kategorien von Entwurfsmustern

Entwurfsmuster für Software existieren in unterschiedlichen Größenordnungen und Abstraktionsniveaus. Sie werden im allgemeinen in die folgenden Kategorien eingeordnet:

- Architekturmuster: Architekturmuster drücken fundamentale Strukturkriterien und Architekturkonzepte eines Software-Systems aus. Sie beschreiben eine Menge von vordefinierten Subsystemen, deren Verantwortlichkeiten und Regeln sowie Hinweise, die zwischen ihnen bestehenden Beziehungen zu realisieren.

- Entwurfsmuster: Entwurfsmuster bieten ein Schema zur Verfeinerung von Subsystemen oder Komponenten. Ein Entwurfsmuster beschreibt häufig vorkommende Strukturen kommunizierender Komponenten.

- Idiome: Idiome sind in der Regel spezifisch für eine Programmiersprache. Sie beschreiben wie man eine bestimmte Komponente, ihre Funktionalität oder Beziehungen zu anderen Komponenten realisieren kann.

Die Grenzen zwischen den Kategorien sind fließend und vom konkreten Anwendungsfall abhängig. Die Software-Konstruktion mit Entwurfsmustern hat eine professionelle Unterstützung der Erstellung und Pflege großer und komplexer Systeme zum Ziel. Zur Erreichung dieses Ziels werden sowohl die funktionalen als auch die nicht-funktionalen Eigenschaften explizit betrachtet.

Die erste Kategorie von Entwurfsmustern, *die Architekturmuster*, gehen häufig auf die nicht-funktionalen Eigenschaften eines Systems ein. Zu diesen, in der Software-Entwicklung immer wichtiger werdenden (und für eCommerce-Systeme besonders wichtigen) „nicht funktionalen“-Eigenschaften, gehören z.B.: Wiederverwendbarkeit, flexible Änderbarkeit, Erweiterbarkeit, Anpassbarkeit, Wartbarkeit, Portierbarkeit, Zuverlässigkeit, Robustheit, Fehlertoleranz, Verfügbarkeit, Entwicklungsgeschwindigkeit, Testbarkeit oder die Integrationsfähigkeit in vorhandene IT-Systeme. Die nicht-funktionalen Eigenschaften sind wichtige Qualitätskriterien für ein Software-System. Auf diese nicht-funktionalen Eigenschaften, die im großen Maße von den gewählten Architekturmustern abhängen, wurde in Kapitel 2 ausführlich eingegangen. In Kapitel 3 wurde zudem gezeigt, daß viele der existierenden Standard-Systeme unzulängliche bzw. stark einschränkende Architekturmuster aufweisen. Durch Gegenüberstellung konnte in Kapitel 3 dementsprechend gezeigt werden,

daß die mit dem vorgestellten Framework erstellten eCommerce-Systeme den heutigen Standard-Systemen in fast allen „nicht-funktionalen“ Eigenschaften überlegen sind.

Das vorliegende Kapitel beschäftigt sich mit den funktionalen Eigenschaften von eCommerce-Systemen. Die funktionalen Eigenschaften von Systemen sind stark von dem konkreten Anwendungsfall abhängig, wenngleich, wie in Kapitel 1 erläutert wurde, in unterschiedlichsten eCommerce-Systemen die gleichen bzw. ähnliche funktionale Eigenschaften wiederzufinden sind. Nachfolgend soll anhand des konkreten Beispieles eines eCommerce-Shop-Systems ein Entwurfsmuster für ein eCommerce-System vorgestellt werden.

In den Kapiteln 5 und 6 wird dann auf „Idiome“ eingegangen, d.h. es wird beschrieben wie man die einzelnen Komponenten, ihre Funktionalität und ihre Beziehungen zu anderen Komponenten realisieren kann. Abweichend zum „Normalfall“ wird in diesen Kapiteln aber nicht auf eine bestimmte Programmiersprache eingegangen, sondern eine allgemeingültige Beschreibung mit den Elementen der UML (Unified Modelling Language) [60-66] angestrebt.

4.2 Entwurfsmuster für eCommerce-Shop-Systeme

4.2.1 Rollenverteilung

Analog zu den real existierenden Rollen (Käufer und Betreiber bzw. Shop-Inhaber) können eCommerce-Shop-Systeme in zwei verschiedene Bereiche aufgeteilt werden:

- die für den Kunden zugängliche „Storefront“

- und den, nur für den Shop-Betreiber (und eventuell seine Mitarbeiter) zugänglichen „Backoffice“.

Gemäß der Aufgabenverteilung in größeren Unternehmen kann der Backoffice in weitere Subrollen aufgesplittet werden, wie z.B. „Kundenmanager“, „Produktmanager“, „Lagerverwaltung“, „Bestellabwicklung“ usw.
In Abbildung 27 wird diese Rollenverteilung dargestellt:

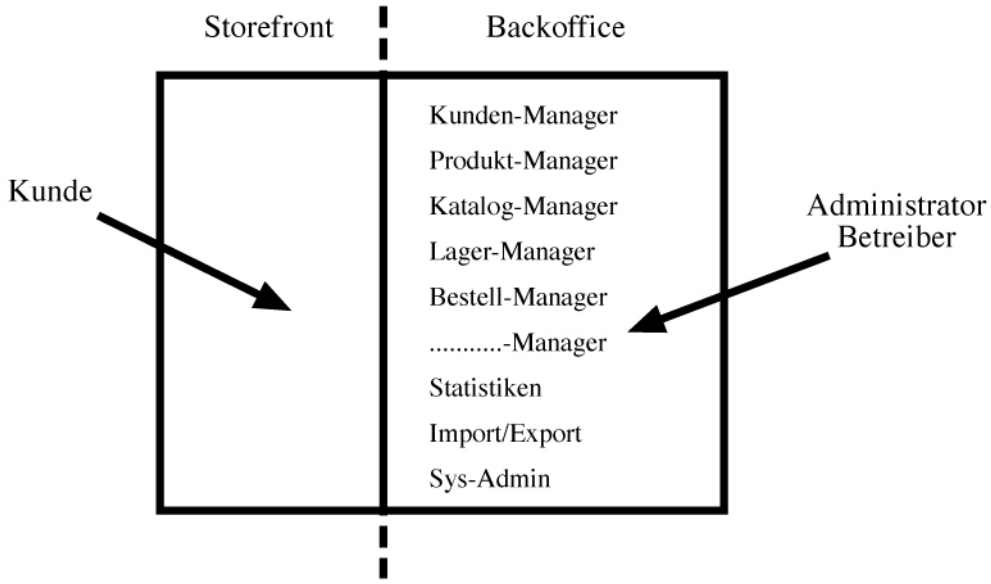


Abbildung 27: Rollenverteilung in eCommerce-Shop-Systemen

Die Betrachtung unterschiedlicher Rollen vereinfacht die Darstellung von eCommerce-Shop-Systemen zunächst in der Art, daß das System in zwei funktional voneinander getrennte Teile zerlegt werden kann. Diese Trennung kann, wenn konsequent fortgeführt, sogar dazu führen, eine größere Anwendung in mehrere überschaubare Teilanwendungen zu zerlegen. In Kapitel 8 wird z.B. ein eCommerce-System vorgestellt, bei dem Storefront und Backoffice durch zwei verschiedene, unabhängige Anwendungen realisiert wurden, die auf einem gemeinsamen Datenbestand (einer relationalen Datenbank) operieren. Im folgenden werden die Bereiche Backoffice und Storefront isoliert voneinander betrachtet.

4.2.2 Funktionen Backoffice

Der Backoffice eines eCommerce-Shop-Systems muß es ermöglichen, Warengruppen und Produkte einzupflegen und zu ändern, Kunden und Bestellungen zu verwalten, Zahlungssysteme und Lieferbestimmungen festzulegen, allgemeine Systemeinstellungen vorzunehmen und vieles mehr. Anhand eines „Produktmanagers“ sollen zunächst die notwendigen Funktionalitäten und Bedienoberflächen, die für das Anlegen, Ändern, Kopieren, Löschen und Suchen, also für das Arbeiten mit Produkten notwendig sind, erläutert werden. Ziel ist es, an diesem Beispiel ein „Muster“ aufzuzeigen, nach dessen Vorbild (fast) alle „Manager“ aufgebaut sind. Später (in Kapitel 5) wird dieses immer wiederkehrende Muster erneut aufgegriffen. Aus diesem Muster werden für alle Business-Objects wiederverwendbare Komponenten bzw. Business-Object-Facilities geschaffen. In den Abbildungen 28 bis 30 sind die beteiligten Komponenten dargestellt:

Abbildung 28: List-Page-Komponente

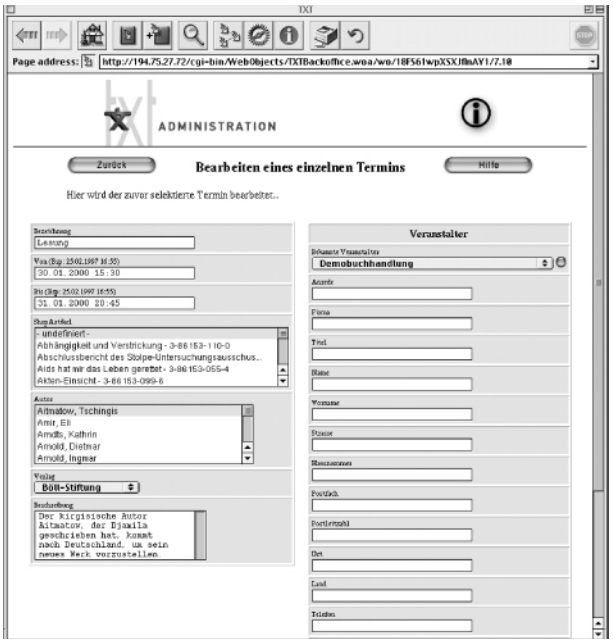


Abbildung 29: Edit-Page-Komponente

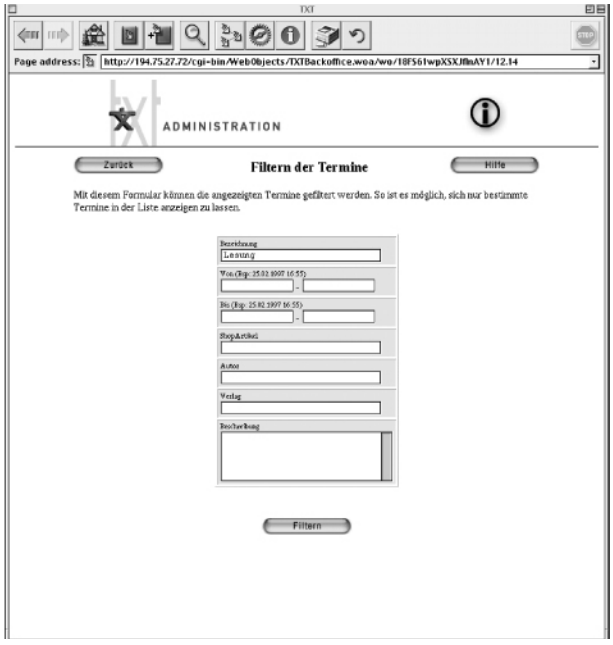


Abbildung 30: SearchPage-Komponente

Die List-Komponente listet die vorhandenen Produkte auf und ermöglicht das Selektieren einzelner Objekte. In ihr werden Funktionen wie Editieren, Neuanlegen oder Kopieren angeboten, die jeweils ein Wechsel zur Edit-Komponente nach sich ziehen. Mit Hilfe der Edit-Komponente kann auf alle Attribute eines Produktes zugegriffen werden. Das Anwählen der Speicherfunktionen in der Edit-Komponente führt zu der persistenten Speicherung des editierten Objektes.

Bei einer großen Anzahl von Produkten kann das Auffinden eines speziellen Produktes oder einer bestimmten Gruppe von Produkten zu einer anstrengenden, zeitraubenden Aufgabe werden. Deshalb bieten die meisten Systeme Such-. bzw. Filterfunktionen an, die es ermöglichen bestimmte Objekte schneller auffinden zu können. Die in Abbildung 31 dargestellte Suchkomponente bietet diese entsprechende Funktionalität einer Internet-Suchmaschine. Komfortable List-Komponenten ermöglichen darüber hinaus die Sortierung nach verschiedenen Attributen (beispielsweise durch Klick auf die Spaltenüberschrift), oder ein „Batch-Processing (vor- und zurückblättern einer zuvor festgelegten Anzahl von Objekten).

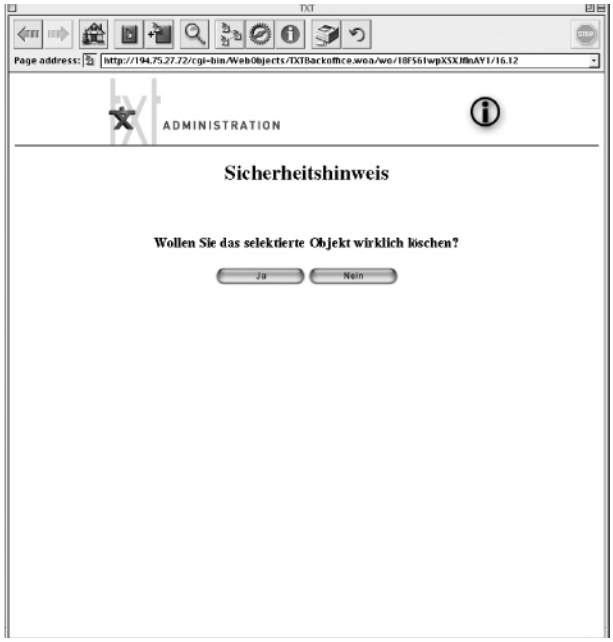


Abbildung 31: Hinweis-Dialog-Komponente

Abbildung 32: Info-Komponente

Die in den Abbildungen 31 und 32 dargestellten Komponenten können das „Muster“ weiter verfeinern. Eine Hinweis-Komponente ermöglicht einen zusätzlichen Dialog mit dem Benutzer, um ihn z.B. vor den Folgen bestimmter Handlungen zu warnen und ein versehentliches Ausführen von Funktionen (z.B. das Löschen von Objekten) zu vermeiden. Mit Hilfe einer Info-Komponente können komplexere Geschäftsobjekte, die beispielsweise viele Verbindungen bzw. Beziehungen zu anderen Objekten haben übersichtlich und zusammenfassend dargestellt werden. In Kapitel 8 wird eine Abwandlung dieser Info-Komponente, z.B. dafür genutzt, um eine druckfertige Rechnung für eine Bestellung zu generieren. In einem anderen Prototypen (TXT) wird ein Produkt-Objekt (ein Buch) mit allen Attributen den zugehörigen Autoren- und Multimedia-Daten, den zugewiesenen Warengruppen und den eventuell verliehenen Auszeichnungen in einer Info-Komponente übersichtlich zusammengefaßt.

Diese fünf Komponenten (List-, Edit-, Such-, Hinweis- und Info-Komponente) bieten bereits (fast) alle notwendigen Funktionen um Produkte innerhalb eines eCommerce-Shop-Systems zu managen. Gelingt es dieses „Muster“ fortzusetzen, bzw. gelingt es diese Komponenten so allgemein zu halten, daß sie für alle vorkommenden Geschäftsobjekte wiederverwendet werden können, so hat man bereits einen großen Teil der gesamten Backoffice-Funktionalität umgesetzt.

Die einzelnen Komponenten des „Manager-Musters“ können wiederum aus verschiedenen, wiederverwendbaren Einzel-Elementen bestehen. Zur Veranschaulichung werden in Abbildung 33 die verschiedenen Einzelkomponenten einer Edit-Komponente vorgestellt.

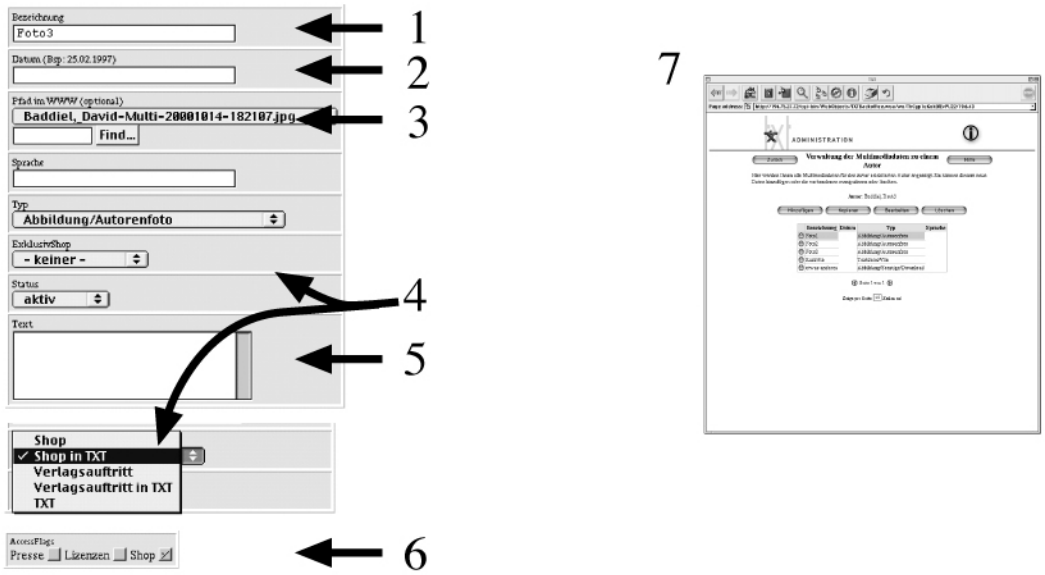


Abbildung 33: Elementarkomponenten einer Edit-Page

1. Normales Textfeld

2. Textfeld für die Eingabe eines Datums

3. Upload-Funktionalität, die es ermöglicht Bilder und andere Multimedia-Daten vom lokalen Rechner zum Server zu transportieren

4. Verschiedene „Choices“ (aufklappende Auswahlboxen, die vordefinierte Möglichkeiten bieten)

5. Ein größeres Textfeld für die Eingabe von längeren Beschreibungen

6. Schalter

7. Sub-List-Komponenten für die Bearbeitung von „Arrays“

In Kapitel 5 werden die fünf Hauptkomponenten und ihre Subkomponenten detailliert beschrieben. Neben dem Manager-Muster, daß das bestimmende Grundmuster im Backoffice ist, existieren weitere Komponenten, die in zahlreichen eCommerce-Anwendungen in ähnlicher Art vorkommen. Dazu gehört eine Log-In-Komponente, die verhindert, daß nicht autorisierte Personen in den Backoffice gelangen. Eine Import-Export-Komponente die es ermöglicht größere Mengen von z.B. Produkten mit einem „Schwung“ (z.B. aus einem ASCII-File oder XML-File) einzupflegen. Eine Statistik-Komponente in der alle relevanten Vorgänge im Shop dargestellt werden, oder z.B. eine Systemadministrator-Komponente, in der verschiedenste Systemparameter, wie beispielsweise die maximale Sessionlänge oder die Anzahl der Undo-Schritte eingestellt werden können. In Abbildung 34 ist ein Entwurfsbeispiel für einen eCommerce-Shop-Backoffice dargestellt:

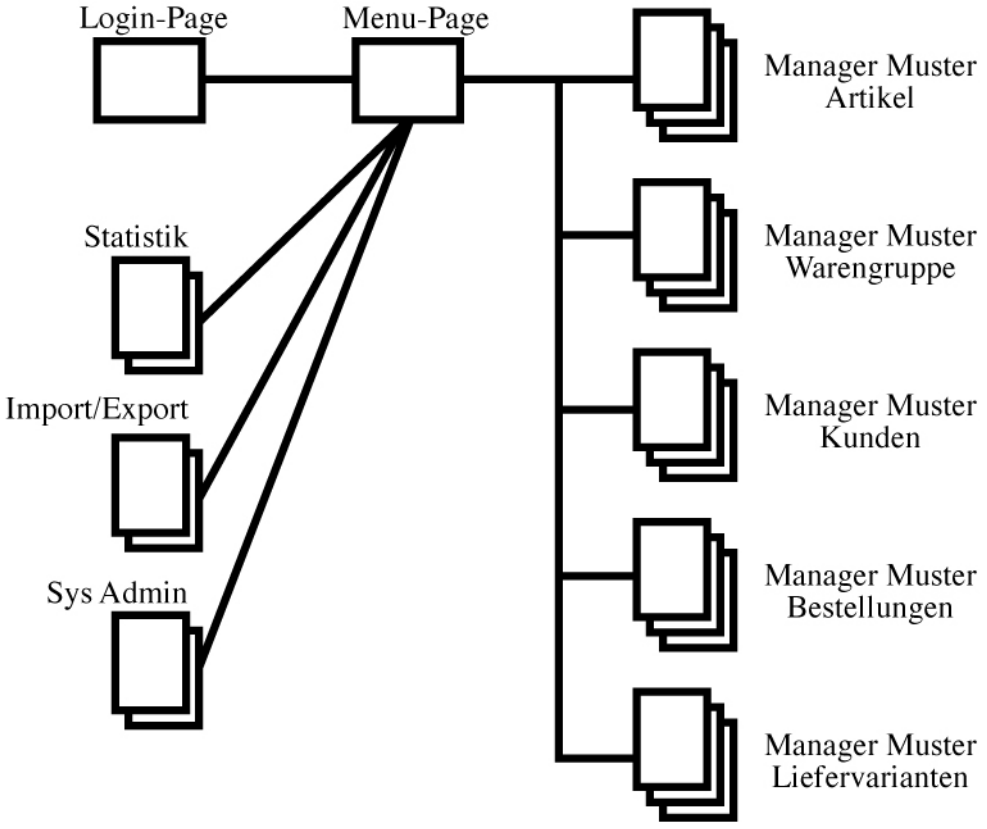


Abbildung 34: Entwurfsmuster für den „Backoffice“ eines eCommerce-Shop-Systems

4.2.3 Funktionen Storefront

Betrachtet man heutige eCommerce-Shop-Systeme, wie z.B. *www.dell.com*, *www.apple.com* oder *www.amazon.com* genauer, so stellt man fest, daß sich hinter den komplexen Storefronts immer der gleiche grundlegende Kern, das gleiche Grundmuster verbirgt. Dieses Muster besteht lediglich aus sechs grundlegenden Elementen: MainPage, List-Page, DetailPage, Basket, Bestelldatenaufnahme und Bestellbestätigung.

Die folgende Abbildung gibt einen Überblick über das grundlegende Muster eines e-commerce-Shop-Storefronts.

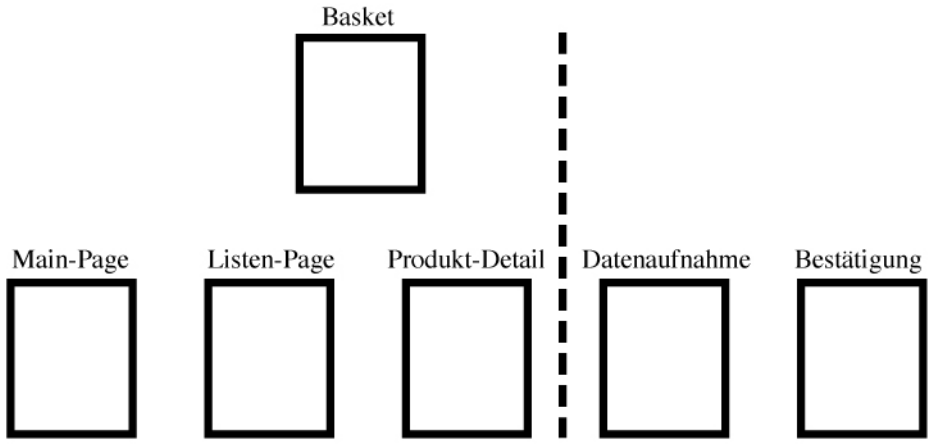


Abbildung 35: Grundmuster eines eCommerce-Storefronts

MainPage
Diese Seite ist die Start- bzw. Begrüßungsseite, von der aus in die verschiedenen Warengruppen bzw. Kategorien verzweigt wird.

List-Page
Auf dieser Art von Seite werden Ansammlungen von Produkten (z.B. alle Produkte einer Warengruppe) oder auch Unterwarengruppen dargestellt. Die gleiche Funktionalität kann auch für die Anzeige von Suchergebnissen genutzt werden.

DetailPage
Dienen die verschiedenen Formen von List Pages vorwiegend dazu, einen Überblick zu

geben, so wird durch die DetailPage-Funktionalität eine detaillierte Information über ein bestimmtes Produkt ermöglicht.

Basket
In den Basket oder den Warenkorb können die Produkte gelegt werden, die später bestellt werden sollen. Normalerweise ermöglicht eine typische Warenkorbseite auch das Zurückspringen auf die entsprechende DetailPage, das Herausnehmen von Produkten und die Veränderung der Stückzahl.

Aufnahme der Bestelldaten
Auf dieser Seite (die in der Praxis oft in Einzelschritte zerlegt wird) müssen vor der eigentlichen Bestellung die gewünschten Liefer- und Rechnungsanschriften, die Art der Bezahlung, die eventuell gleich notwendig werdenden Zahlungsinformationen, wie Bankverbindung oder Kreditkarten-Nummern und die gewünschte Liefervariante angegeben werden. Durch den Klick auf den Bestellknopf wird dann die eigentliche Bestellung ausgelöst.

BestätigungsPage
Auf dieser Seite bzw. mit dieser Funktionalität wird eine Bestellbestätigung (meist inklusive einer nochmaligen Auflistung der bestellten Produkte) realisiert.

Dieses grundlegende Muster wird in vielen Fällen soweit verfeinert und durch nützliche (und auch weniger nützliche) Zusatzfunktionen ergänzt, daß es häufig (vor allem ohne eine gewisse Übung) kaum noch zu erkennen ist. In der folgenden Abbildung sind einige der typischen Erweiterungen des grundlegenden Storefront-Musters dargestellt:

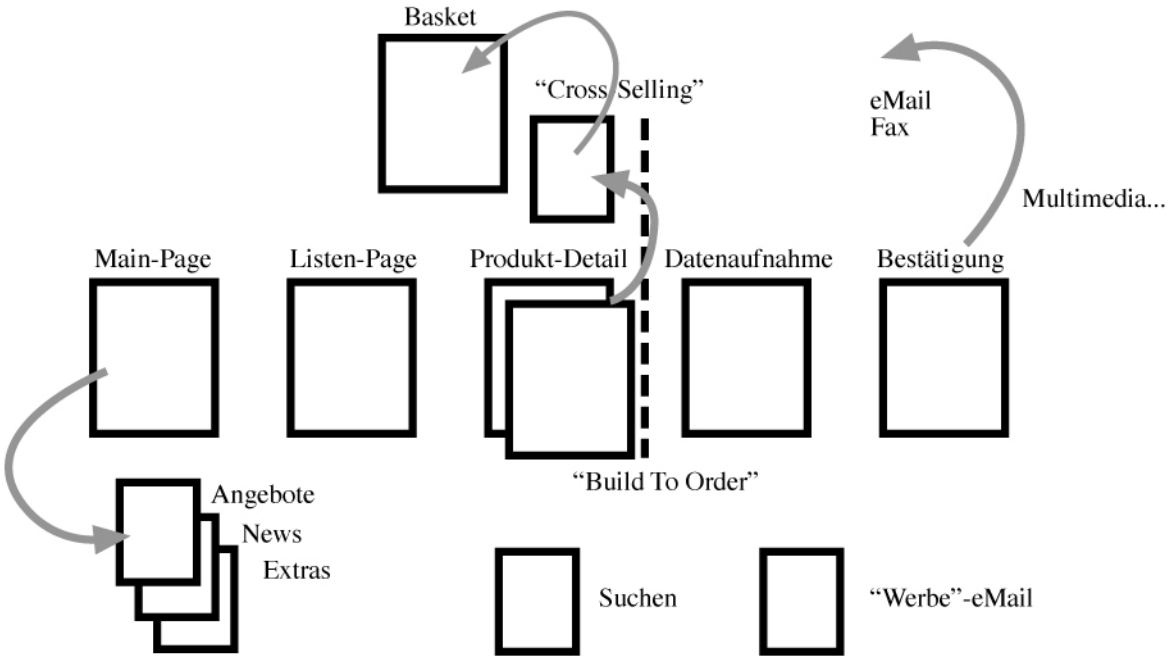


Abbildung 36: Erweiterungen des Storefront-Grundmusters

1. Wie schon oben angesprochen wird die Aufnahme der verschiedenen Daten, wie z.B. Lieferadresse, Rechnungsadresse, die Auswahl der gewünschten Liefervariante, die Auswahl der gewünschten Zahlungsvariante und die eventuell notwendig werdenden Angaben über Zahlungsinformationen in mehreren, besser überschaubaren Einzelschritten bzw. Einzelseiten zerlegt.

2. Bestellbestätigungen können auch auf anderen Wegen wie z.B. per E-Mail oder Fax zugestellt werden.

3. Die List-Page-Funktionalität muß nicht immer durch tabellenähnliche vertikale Reihen umgesetzt werden, sondern kann auch durch andere Elemente, wie z.B. durch dynamisch veränderbare und anklickbare Grafiken umgesetzt werden. In der nachfolgenden Abbildung ist ein solches Beispiel dargestellt. Es ist einem der Prototypen entnommen (vgl. Kapitel 8) in dem die drei Unterwarengruppen (Putter, Wood, Eisen) der „Ober-Warengruppe" Golfschläger grafisch umgesetzt wurden.

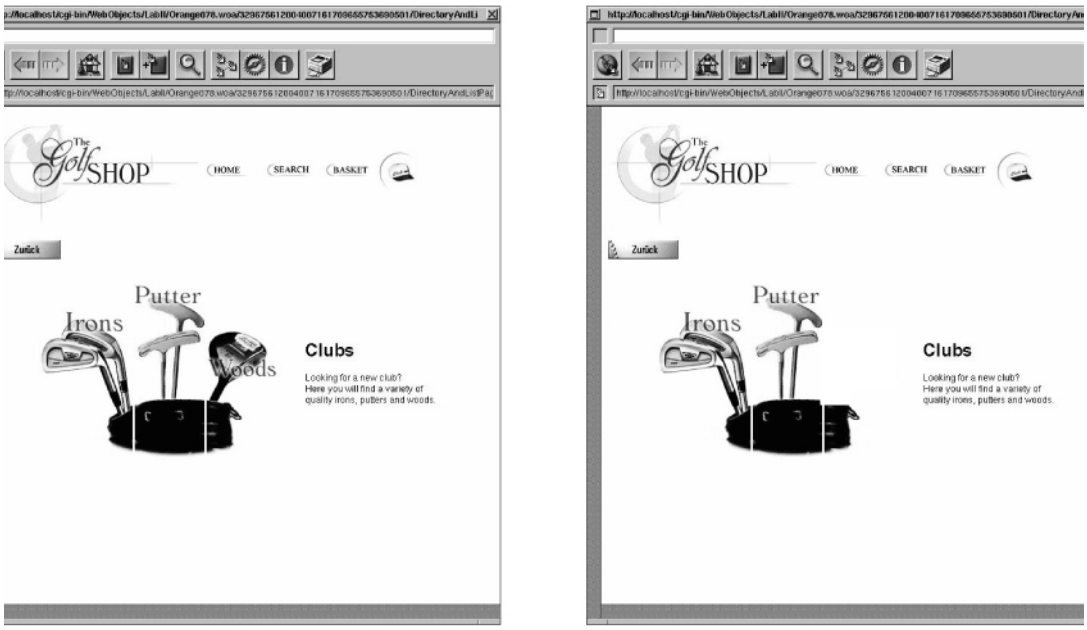


Abbildung 37: Umsetzung von List-Page-Funktionalität durch dynamische Graphiken

4. Die DetailPage-Funktionalität kann dahingehend erweitert werden, daß in einer Detail-Ansicht nicht nur feste Informationen dargestellt werden, sondern es zusätzlich möglich ist z.B. einen ausgewählten Artikel individuell abzuwandeln. Vorbildliche Beispiele für solch eine „Build To Order“-Funktionalität weisen z.B. die eCommerce-Anwendungen *www.dell.com* und *www.apple.com* auf.

5. Die Cross-Selling-Funktionalität soll den Kunden über mögliche, passende Zusatzprodukte informieren und so den Umsatz eines Shops erhöhen. Typischerweise werden die Cross-Selling-Informationen beim Verlassen einer Produkt-Detail-Page oder beim Hineinlegen von einem Produkt in den Warenkorb (Basket) eingeblendet. Einige Shop-Formen (z.B. www.amazon.com blenden diese Cross-Selling Informationen generell bei jeder Produktansicht ein („Kunden, die dieses Buch gekauft haben, haben sich auch für folgende Bücher interessiert: ...“).

6. Zusätzlich zum „Browsen“ im Katalog bieten die meisten Storefronts Suchfunktionen an, mit denen mehr oder minder spezialisiert nach Produkten oder bestimmten Produkteigenschaften gesucht werden kann.

7. Desweiteren existieren oft Zusatzfunktionen mit denen verschiedene Marketing-Strategien

umgesetzt werden können. Zu den häufigsten Zusatzfunktionen gehören Sonderangebote, Rabattfunktionen oder auch Grußkarten.

8. Besitzt ein Shopsystem erst einmal die Bestellinformation eines Kunden (zu denen in der Regel neben der Anschrift auch eine E-Mail-Adresse gehört), so wird dieses häufig ausgenützt um ihn mit „Werbe-Mails“ bzw. „Informations-Updates“ zu versorgen.

9. Zusätzliche Funktionalitäten wie Multimedia-Informationen oder Animationen kommen ebenfalls häufig zum Einsatz - sinnvoll angewandt (z.B. kann man bei einer Produkt-DetailPage verschiedene zusätzliche multimediale Informationen über ein Produkt anbieten) können diese Ergänzungen eine effiziente Bereicherung darstellen. Allzuoft aber werden die normalen Storefrontseiten mit Effekten und Animationen überladen, so daß die Übersicht verloren geht und sich zudem die Ladezeit der entsprechenden Seite unnötig verlängert. Durch die übermäßige Verwendung von nicht HTML-konformen Elementen wie z.B. „Javascript“ oder „Flash“-Animationen versperren sich zudem viele Storefronts eigenverschuldet einer großen Anzahl von potentiellen Kunden.

10. Kann ein bereits bekannter Kunde, z.B. durch eine Login-Funktion gleich zu Beginn (zu dem Zeitpunkt an dem er den Shop betritt) wiedererkannt werden, so können einige oder alle Funktionen des Shops individuell auf seine Bedürfnisse und Neigungen „personalisiert“ werden und dadurch eine größere Kundenzufriedenheit und Kundenbindung erreicht werden.

In den späteren Kapiteln wird gezeigt, wie für die grundlegenden Elemente des Storefront-Musters (wie z.B. der List-Page-Funktionalität, die sowohl für die Auflistung von Warengruppen, von Produkten und auch für die Darstellung von Suchergebnissen verwandt werden kann) wiederverwendbare Komponenten zur Verfügung gestellt werden können.

Kapitel 5

eXBO-Facilities (eXBOF)

5.1 Einleitung

Wie im vorangegangenen Kapitel Entwurfsmuster gezeigt, können beim Aufbau von eCommerce-Systemen grundlegende Muster, wie z.B. das Manager-Muster identifiziert werden. Die einzelnen Elemente des Musters (z.B. die List-Page-Komponente oder die Edit-Page-Komponente) bilden dabei die sogenannten „Facilities“-Elemente bzw. Komponenten für die Bearbeitung, Darstellung und Verarbeitung von Wissensobjekten. In Abbildung 34 ist wiedergegeben, wie ein und daßelbe Entwurfsmuster (in diesem Falle das Manager-Muster) für eine Vielzahl von Wissensobjekten genutzt wird, sich also ständig wiederholt. In der Praxis bedeutet dies zunächst, daß für jedes Business-Object jede einzelne „Facility“ implementiert werden muß. Für ein Objekt „Produkt“ eine Produkt-List-Page, eine Produkt-Edit-Page, eine Produkt-SearchPage und eine Produkt-InfoPage. Für ein Objekt „Warengruppe“ eine Warengruppen-List-Page, eine Warengruppen-Edit-Page, eine Warengruppen-SearchPage und eine Warengruppen-InfoPage usw. Angelehnt an die Grundsätze der objektorientierten Programmierung wird ein erfahrener Programmierer zunächst versuchen, gemeinsame

Funktionalitäten dieser verschiedenen (und doch im wesentlichen gleichen) Facilities zu erkennen und in einer Oberklasse auszulagern. Durch diese Maßnahmen wird die Wartbarkeit erhöht (spätere „Verbesserungen“ müssen z.B. nur an einer einzigen Stelle vorgenommen werden und gelten dann sofort für Dutzende von Subklassen) und vor allem der Zeit- und Arbeitsaufwand stark verringert. Trotzdem bleibt ein gewisser Aufwand vorhanden, um die spezialisierten Subklassen zu implementieren und zu instantiieren. Geht man vom Manager-Muster aus, das lediglich vier Facilities (List, Edit, Search und Info) pro Business-Object vorsieht, sind für eine eCommerce-Anwendung die zwanzig verschiedenen Business-Objects benötigt, insgesamt 80 Facility-Komponenten zu implementieren. Bei einer eCommerce-Anwendung mit 100 verschiedenen Business-Objects würden 400 verschiedene Facility-Komponenten benötigt usw.

Die Idee der „eXBO-Facilities ist es, eine einzige universelle Instanz einer Facility (z.B. der List-Page) zu besitzen, die von allen Business-Objects genutzt werden kann. Sollte dies gelingen, so verringert sich der Arbeitsaufwand um ein Vielfaches. Unabhängig davon ob 20, 40 oder 400 Business-Objects, die Anzahl der benötigten Facilities (z.B. für das Manager-Muster) bleibt konstant. Im Beispiel des angeführten Manager-Musters sind es konstant immer vier.
In Abbildung 38 sind die Einsparungspotentiale zur Verdeutlichung noch einmal graphisch aufbereitet. Abbildung 38 stellt die Anzahl der verschiedenen Business-Objects in Relation zur Anzahl der benötigten Facilities, bzw. in Relation zum Arbeitsaufwand für die Facilities.
Dabei wird versucht, eine Abschätzung des benötigten Arbeitsumfangs (ausgedrückt durch Programmzeilen) bei nicht-objektorientierter Programmierung, bei objektorientierter Programmierung und bei objektorientierter Programmierung und zusätzlichem Einsatz von eXBO-Komponenten zu geben. Im Falle der objektorientierten Programmierung wird die Annahme gemacht, daß ca. 50 Prozent der Programmzeilen in die entsprechenden Oberklassen ausgelagert werden können

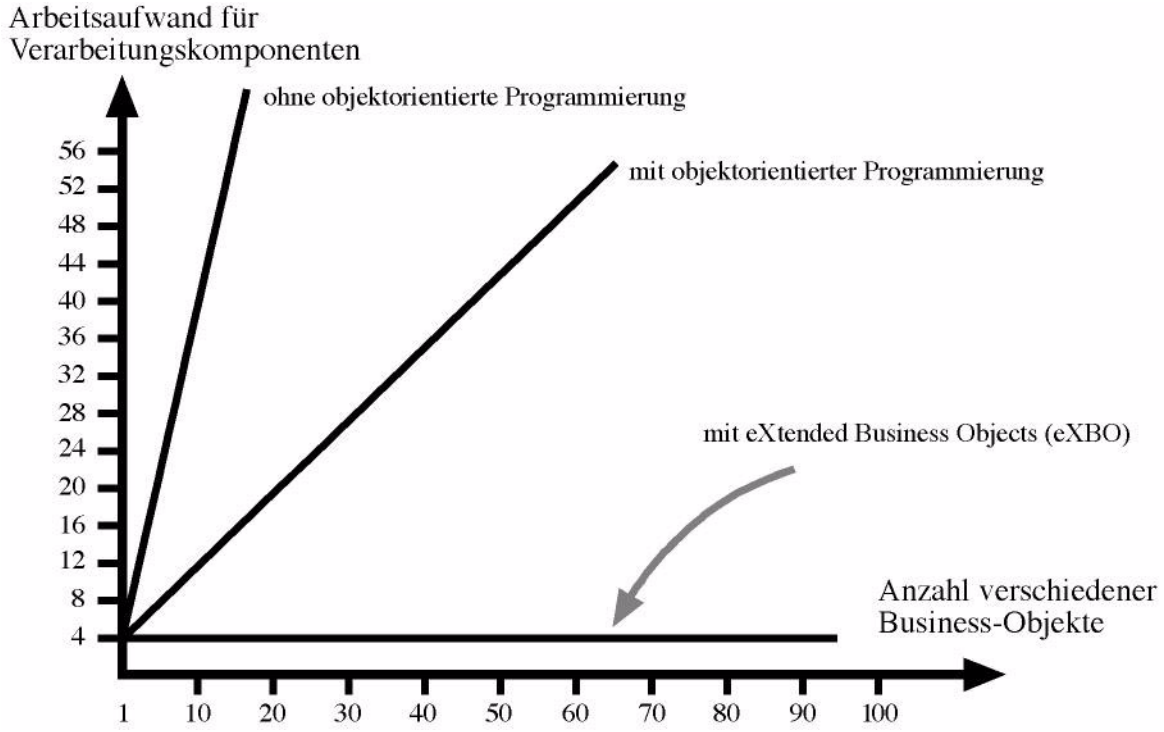


Abbildung 38: Einsparungspotentiale durch OO-Programmierung und durch eXBO-Komponenten

Abbildung 38 verdeutlicht am Beispiel der Facilities für das „Manager-Muster“ die Einsparungspotentiale, die durch den Einsatz von OO-Programmierung und im zweiten Schritt durch eXBO-Komponenten erzielt werden können. Dies äußert sich in entsprechend wesentlich verkürzten Entwicklungszeiten und der deutlichen Senkung der Herstellungskosten. In den folgenden Abschnitten dieses Kapitels werden eXBO-Facilities für das Manager-Muster vorgestellt. Die Anforderung, die die eXBO-Facilities dabei an die Business-Objects stellen, werden dabei in diesem Kapitel angesprochen und im nächsten Kapitel vertieft.

5.2 eXBO-Facility List

Bevor auf technische Details eingegangen wird, soll zunächst noch einmal die Funktionalität dieses Elements vorgestellt werden. Nachfolgend wird die eXBO-Facility List verkürzt als „eXBO-Listkomponente“ bezeichnet. Obwohl in der Abbildung 39 eine konkrete Implementierung vorgestellt ist (siehe Kapitel 8), sei an dieser Stelle angemerkt, daß die optische Erscheinung von HTML-Designern jederzeit verändert bzw. angepaßt werden kann und an dieser Stelle unerheblich ist. Allein die Funktionalität soll an dieser Stelle betrachtet werden:

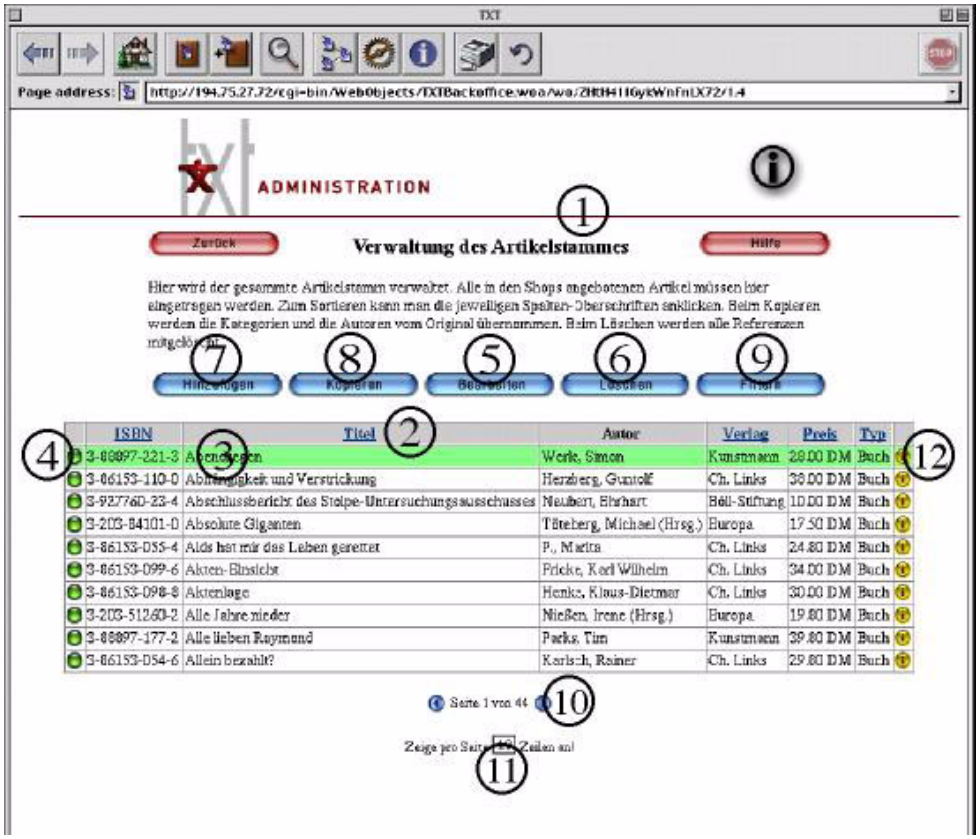


Abbildung 39: eXBO-Facility List

1. An dieser Stelle wird in der Listkomponente angegeben, welcher Typ von Objekt in Bearbeitung ist. In dem angegebenen Beispiel werden Artikel angezeigt.

2. Die Spaltenüberschriften, in dem die Attributnamen dargestellt werden. Die Anzahl und die Auswahl der Reihenfolge der Spalten ist frei definierbar. Die tatsächliche Bezeichnung des Attributs innerhalb des Business-Objects muß dabei nicht mit der Spaltenüberschrift übereinstimmen. Aus verschiedensten Gründen werden während der Implementierung von Programmierern häufig Attribut-Bezeichnungen gewählt, die für Endnutzer nicht immer eindeutig genug zu verstehen sind. Auf diesem Weg ist die Bezeichnung der Objekt-Attribute und die der Spaltenüberschriften voneinander getrennt und den jeweiligen Experten überlassen. Durch Klick auf die Spaltenüberschriften kann der Tabelleninhalt nach den Spalten-Attributen sortiert werden. Bei wiederholtem Klick erfolgt die Sortierung in umgekehrter Reihenfolge. Die „bold“-Darstellung gibt an, nach welchem Spalten-Attribut gerade sortiert wurde (per Default ist es immer die erste Spalte).

3. In diesen Zeilen werden die Attributwerte der Objekte dargestellt. Jede Zeile entspricht dabei einem Objekt.

4. Durch Anwahl dieses Auswahlknopfes kann ein Objekt selektiert werden. Die entsprechende Zeile wird dann farblich hervorgehoben.

5. Ist ein Objekt selektiert, kann es mit dem Bearbeiten-Link für die Bearbeitung auf der sogenannten Edit-Page ausgewählt werden (siehe eXBO-Facility Edit im nächsten Abschnitt). Das selektierte Objekt wird dann der Edit-Komponente übergeben und kommt in dieser zur Bearbeitung.

6. Durch die Anwahl der Funktion „Löschen“ wird das selektierte Objekt aus der Tabelle und aus dem Speicher gelöscht. Um eine versehentliche Löschung zu vermeiden, wird das Objekt nicht unmittelbar entfernt sondern zuvor die Sicherheitsabfrage-Komponente aufgerufen (vergleiche Kapitel 7).

7. Durch die Anwahl der Funktion „Hinzufügen“ wird ein neues Objekt des entsprechenden Typs erzeugt und zur Bearbeitung an die Edit-Komponente (siehe nächsten Abschnitt) übergeben, wo das neue, leere Objekt bearbeitet werden kann.

8. Wurde zuvor ein Objekt selektiert, so kann dieses auch kopiert werden. Das Kopieren eines Objektes entspricht dem Hinzufügen eines neuen, leeren Objektes und dem anschließenden Eingeben derselben Attributwerte für das neue Objekt. Die Kopierfunktionalität erleichtert das Eingeben von neuen Objekten die anderen sehr ähnlich sind. Die in Abbildung 39 dargestellte Komponente ist dem Kapitel 8 „Realisierte Anwendungen“ entnommen. In dem dort vorgestellten Prototypen TXT (eine Portalplattform für Buchverlage) erleichtert die Kopierfunktionalität beispielsweise das Eingeben von mehreren Büchern einer Reihe, die sich

häufig nur in wenigen Attributwerten wie z.B. Seitenzahl und Autor unterscheiden.

9. Durch Anwahl der Filter-Funktionalität gelangt man zur eXBO-Facility Search, die eine detaillierte Suche ermöglicht.

10. Die Batch-Funktionalität erlaubt das „portionierte“ Darstellen von Objekten bei größeren Mengen. Im angegebenen Beispiel werden pro Batch zehn Objekte dargestellt. Durch Anwählen der Blätterknöpfe kann in den einzelnen Blöcken vor und zurückgeblättert werden. Durch die Doppel-Pfeil-Knöpfe kann direkt auf den Anfang und auf das Ende gesprungen werden.

11. In dem Eingabefeld „Zeige pro Seite ... Zeilen an“ kann ein User die Anzahl der dargestellten Zeilen bzw. die Anzahl der Objekte per Batch selbständig seinem Bedarf anpassen.

12. Durch Betätigen des Info-Knopfes gelangt man zur eXBO-Facility Info, die einen zusammenfassenden Uberblick über alle Attribute des selektierten Objektes bietet.

Das Array von Objekten, das in der Tabelle bzw. in der eXBO-List-Komponente dargestellt wird, muß dieser zuvor übergeben werden. Meist geschieht dies durch die sogenannte Menu-Page. Dies ist eine Seite, in der der Benutzer auswählen kann, welche Objekte er als nächstes bearbeiten möchte. In Abbildung 40 ist diese Menu-Page dargestellt. Durch Anwahl des Menüpunktes „Artikel“ wird das Array mit den vorhandenen Artikel-Objekten an die List-Komponente übergeben. Durch Anwahl des Menüpunktes „News“ wird das Array mit den vorhandenen News-Objekten übergeben usw.

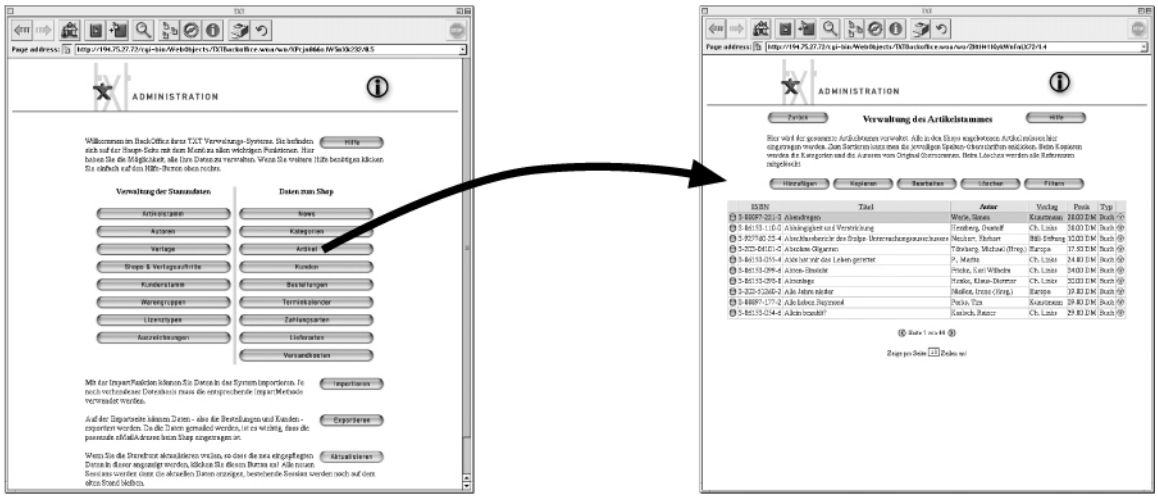


Abbildung 40: Menu-Page und List-Page

Um die vorgestellten Eigenschaften erfüllen zu können (vor allem um sie für jedes beliebige Objekt erfüllen zu können), benötigt die eXBO-Komponente „List“ eine Reihe von Attributen und Methoden. In Abbildung 41 ist das UML-Klassendiagramm für diese Listkomponente dargestellt.

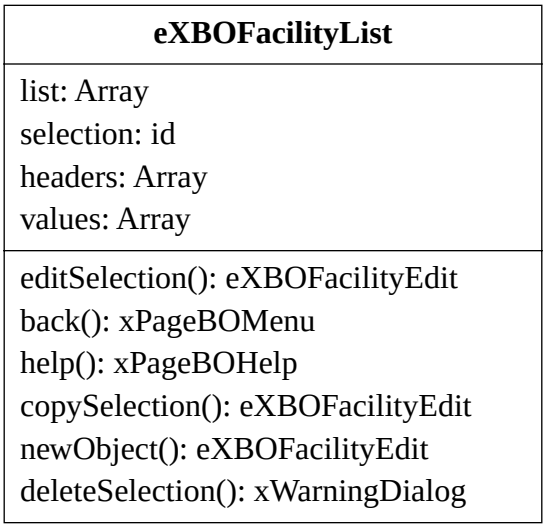


Abbildung 41: UML-Diagramm für die eXBO-Facility List

1. list: In diesem Array werden alle darzustellenden Objekte gehalten, die wie zuvor beschrieben von der Menu-Page übergeben werden.

2. selection: Mit diesem Attribut, dessen Objekttyp erst zu Laufzeit bestimmt werden kann, wird das in der Liste selektierte Objekt festgehalten. Für die Bezeichnung des Attribut-Typs wurde in Anlehnung an die Programmiersprache ObjectiveC die Bezeichnung ID gewählt. Attribute, deren Typ nicht vorhersehbar ist, können dort mit dem allgemeinen Typ ID bezeichnet werden [137].

3. headers: Dieses Array beinhaltet String-Objekte. Der erste String ist dabei der String für die Überschrift der ersten Spalte. Der zweite String ist der String für die Überschrift der zweiten Spalte usw. Diese Informationen können nicht fest in der Listkomponente gehalten werden. Bei der Übergabe des List-Arrays an die List-Komponente „befragt“ die List-Komponente das erste Business-Object in der Liste nach den darzustellenden Spalten in dem sie eine Methode aufruft, die das Headers-Array zurückgibt.

4. values: Wie oben angesprochen, müssen die Attributnamen der Business-Objects nicht mit den Spaltenüberschriften übereinstimmen. Deshalb sind die Spaltennamen auch nicht dazu

geeignet (um mit Hilfe der daraus abgeleiteten Zugriffsmethoden), auf die entsprechenden Attribute zugreifen zu können. In dem Values-Array befinden sich demzufolge eine Reihe von String-Objekten aus denen die zugehörigen Zugriffsmethoden für die Attribute abgeleitet werden können. Das Values-Array wird ebenfalls von dem ersten darzustellenden Business-Object „erfragt“ (durch den Aufruf der entsprechenden Methode des Business-Objects). Die Anzahl und die Reihenfolge der String-Objekte im Values-Array muß dabei exakt mit der des Header-Array übereinstimmen, sonst werden womöglich Attributwerte in einer Spalte dargestellt, die nicht mit der Spaltenüberschrift übereinstimmen. Auf die genauere Funktionsweise der eXBO-List-Komponente wird nachfolgend (bei der Vorstellung der dynamischen HTML-Oberfläche der eXBO-List-Komponente) noch einmal vertiefter eingegangen.

5. editSelection: Diese Methode soll ausgeführt werden, wenn der Benutzer den Bearbeiten-Knopf auswählt (vergleiche Abb. 39). Das selektierte Objekt wird an die eXBO-Edit-Komponente übergeben. Anschließend wird die eXBO-Edit-Komponente aufgerufen. (Die eXBO-Edit-Komponente wird im nächsten Abschnitt vorgestellt.)

6. deleteSelection: Diese Methode wird ausgeführt, wenn der Benutzer den Knopf „Löschen“ betätigt (vgl. Abb. 39). Bevor das selektierte Objekt tatsächlich gelöscht wird, wird mit Hilfe der Sicherheitsabfragekomponente (vgl. Abb. 31) noch eine Sicherheitsabfrage durchgeführt.

7. copySelection: Diese Methode wird ausgeführt, wenn der Benutzer die Kopieren-Funktionalität auswählt (vgl. Abb. 39). Von der eXBO-List-Komponente wird im ersten Schritt ein neues Objekt (des gerade in Bearbeitung befindlichen Business-Objects-Typs) erzeugt. Im zweiten Schritt werden die Attributwerte des neuen Business-Objects mit den Attributwerten des gerade selektierten Objektes gleichgesetzt.

8. newObject: Diese Methode wird ausgeführt, wenn der Benutzer die Funktion „Hinzufügen“ auswählt (vgl. Abb. 39). Wie bei der „copySelection“-Methode sorgt die eXBO-List Komponente dafür, daß ein neues Objekt (des gerade in Bearbeitung befindlichen Objekttyps) erzeugt wird. Das neue Objekt wird dann der eXBO-Edit-Komponente übergeben, wo es bearbeitet werden kann.

Ein weiterer Bestandteil der eXBO-List-Komponente ist deren dynamische HTML-Oberfläche, auf deren Aufbau und Funktionsweise im folgenden eingegangen werden soll.

Nachdem bereits die zur Verfügung stehenden Methoden, Attribute und deren Herkunft erläutert wurden, soll mit der folgenden Abb. 42 auf die Funktionsweise der dynamischen HTML-Oberfläche der List-Komponente eingegangen werden:

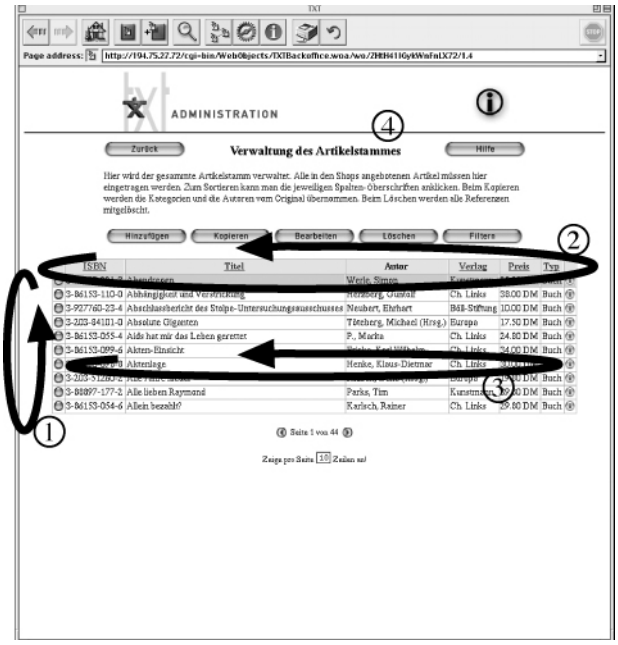


Abbildung 42: Dynamische HTML-Oberfläche der eXBO-List-Komponente

1. Für jedes Element im „listArray“ (genauer gesagt im gerade angezeigten Batch) wird in der Tabelle eine neue Zeile eingefügt.

2. Das erste Objekt im gerade angezeigten Batch des „listArray“ (es könnte auch jedes andere sein) wurde nach dem „headerArray“ gefragt. Durch das „headerArray“ ist die Repetition-Loop für das dynamische Element „headerArray“ bestimmt und damit auch die Anzahl der Spalten und die Spalten-Überschriften.

3. Nun müssen die Spalten noch mit den richtigen Werten gefüllt werden. Wie zuvor beim „headerArray“ wird das erste Elemente des gerade dargestellten Batches nach dem „valuesArray“ gefragt und damit die Repetition-Loop-„values“ gesetzt. Wie zuvor beschrieben können aus den enthaltenen String-Objekten die Zugriffsmethoden für die einzelnen Attribute abgeleitet werden. Jeder Spaltenwert in einer Zeile wird durch den Aufruf der entsprechenden Zugriffsmethode des in der Zeile gerade in Bearbeitung befindlichen Business-Objects ermittelt.

4. Das erste vorkommende Business-Objects im „listArray“ wird nach seinem Namen gefragt, um den dynamischen String „objectName“ aufzulösen.

An dieser Stelle soll angemerkt werden, daß alle hier vorgestellten eXBO-Facility-Komponenten voll skalierbar sind. Die durchgeführten Testläufe mit den Komponenten des Manager-Musters (mit 10, 100, 1000, 100.000 und 500.000 Business-Objects) ergaben nur unwesentliche Zeitdifferenzen (kleiner als 0,8 Sekunden). Auf die Beziehungen zwischen den einzelnen eXBO-Facilities wird am Ende dieses Kapitels noch einmal gesondert eingegangen.

5.3 eXBO-Facility Edit

Wie zuvor bei der eXBO-List-Komponente soll zunächst auf die Funktionalität der Komponente eingegangen werden. Nachfolgend wird die eXBO-Facility Edit verkürzt eXBO-Edit-Komponente genannt. Die eXBO-Edit-Komponente dient dazu neue oder bereits vorhandene Business-Objects bearbeiten zu können. Als Bestandteil des Manager-Musters (s. Kapitel 4) wird die eXBO-Edit-Komponente typischerweise nach der eXBO-List-Komponente aufgerufen (nachdem dort ein Objekt selektiert wurde und anschließend die Funktion „Bearbeiten“ ausgewählt wurde, entsprechendes gilt auch für die Funktionen „Kopieren“ und „Hinzufügen“). Das in der eXBO-List-Komponente selektierte Objekt wird der eXBO-Edit-Komponente übergeben. Alle Attribute des entsprechenden Objektes werden in der eXBO-Edit-Komponente dargestellt und können mit den jeweiligen User-Interface-Elementen bearbeitet werden. In Abbildung 43 ist ein Beispiel für eine Erscheinungsform der eXBO-Edit-Komponente dargestellt.

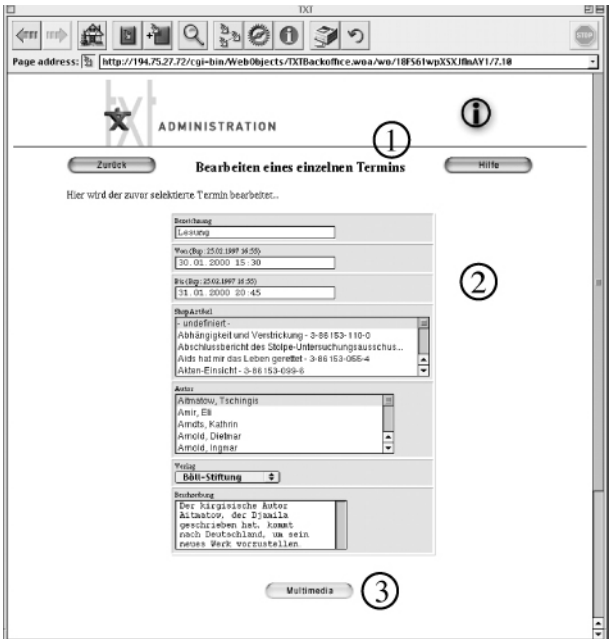


Abbildung 43: eXBO-Edit-Komponente

1. Wie zuvor bei der eXBO-List-Komponente wird die Information, welcher Typ von Objekt gerade in der Bearbeitung ist, angegeben. Diese Information wird von dem, zuvor in der eXBO-List-Komponente selektierten und an die eXBO-Edit-Komponente übergebenen Objekt abgefragt. Hierfür wird die gleiche Methode des eXBO in Anspruch genommen, wie zuvor in der eXBO-List-Komponente.

2. In diesem Bereich können die Attribute des eXBO bearbeitet werden. Jedes Attribut besitzt dabei sein eigenes, für sich passendes User-Interface-Element. Die verschiedenen zur Auswahl stehenden elementaren User-Interface-Elemente wurden bereits in Kapitel 4 vorgestellt. In dem gegebenen Beispiel kommt ein großer Teil von ihnen zur Anwendung. Für das Attribut "Bezeichnung" wird zum Beispiel ein kurzes einzeiliges Textfeldelement gewählt. Für das Attribut "Beschreibung", das typischerweise einen längeren Text enthalten kann, kommt ein mehrzeiliges Textfeld zur Anwendung. Ein Attribut „Land“ (nicht in Abbildung 43 enthalten) würde durch die Auswahl von vordefinierten Elementen aus einer Choice ausgewählt werden. Für das Datum steht ein „Dateformat-Textelement“ zur Verfügung, welches selbständig auf die Eingabe eines korrekten Datum-Formates achtet (und gegebenenfalls zur Korrektur auffordert). Für einzelne Multimedia-Daten, wie z.B. „detailBild“, steht das User-Interface-Element "Upload" zur Verfügung (nicht in Abbildung 43 enthalten), das das Übertragen und Speichern von Multimediadaten zum Server ermöglicht (siehe Kapitel 4).

3. Für Attribute, wie in diesem Beispiel Multimedia-Daten, zu denen beliebig viele Objekte abgelegt werden können, wird ein Hyperlink angeboten. Die Anwahl des Hyperlinks führt zum Aufruf der eXBO-List-Komponente, in der alle entsprechenden Objekte (in diesem Beispiel die zu diesem Termin vorhandenen Multimedia-Daten) aufgelistet werden. Die Elemente können dann mit der eXBO-List-Komponente ganz normal, wie im vorangegangenen Abschnitt beschrieben, bearbeitet werden.

Die „normale“ Implementierung einer Edit-Komponente, wie sie in Abb. 43 dargestellt ist, erfordert schon alleine wegen der Anzahl der zu editierenden Attribute und der verschiedenartigen User-Interface-Elemente einen nicht zu vernachlässigenden Zeit- und Arbeitsaufwand. Das Interessante an der eXBO-Edit-Komponente ist, daß sie sich ohne jeglichen Programmieraufwand vollautomatisch selbst erzeugt – individuell für jedes Business-Objects. Und das ohne vorher wissen zu können, welches Business-Objects als nächstes bearbeitet werden soll. Ohne vorher zu wissen, wie viele Attribute dieses Objekt besitzen wird und mit welchen User-Interface-Elementen diese Attribute bearbeitet werden sollen. Um diese Funktionalität erbringen zu können wird, ähnlich wie zuvor bei der eXBO-List-Komponente, ein wenig „Hilfe“ von den eXBO benötigt. Für die zuvor besprochene eXBO-List-Komponente mußten die eXBO im wesentlichen die Methoden „header“ und „values“ besitzen, die die entsprechenden Arrays für die Spalten der Tabellen zurückgaben.

Für die eXBO-Edit-Komponente muß das zu bearbeitende Objekt lediglich eine weitere Methode besitzen: Die Methode „attributesToEdit“, die ein entsprechendes Array zurückgibt, in dem alle zu bearbeitenden Attribute vorhanden sind.

5.3.1 XAttributes

Würde das Attribut-Array (der Rückgabewert der „attributesToEdit“-Methode) lediglich die „normalen“ Attribute beinhalten, so würde die eXBO-Edit-Komponente nicht in der Lage sein, für jedes Attribut ein individuelles User-Interface-Element anzubieten. Die eXBO-Edit-Komponente benötigt für jedes einzelne Attribut gesonderte Informationen, um den Aufbau des entsprechenden Interface-Elements zu ermöglichen. Für die Übertragung dieser Information (die neben dem gewünschten User-Interface-Element auch weitere Informationen, wie z.B. die anzubietenden Auswahl-möglichkeiten auf einer Choice usw.

beinhalten müssen) werden sogenannte X-Attribut-Objekte eingeführt. Das wichtigste Element bzw. das wichtigste Attribut eines X-Attributs ist der Wert des zu bearbeitenden Attributs selbst. Dieser Wert wird nachfolgend mit „value“ bezeichnet. Zu jedem „value“ gehören innerhalb eines X-Attributes mindestens die drei nachfolgenden zusätzlichen Attribute:

- „label“: Ein String-Objekt das die Bezeichnung des User-Interface-Elements ermöglicht (vgl. Abb. 43). An dieser Stelle einfach den Attribut-Namen einzublenden würde, z.B. schon bei Verwendung einer anderen Landessprache im User-Interface zu Engpässen führen.

- „userInterface“: Dieses Attribut gibt den Typ des gewünschten User-Interface-Elements an. Zu den möglichen User-Interface-Elementen gehören: Text, Textfeld, Choice, DateField, Passwortfield, Sub-Array (für zusätzliche Arrays, wie z.B. Multimedia-Daten), Upload usw.

- „items“: Mit diesem Array können weitere Zusatzinformationen, wie z.B. die String-Objekte, die in einer Choice zur Auswahl stehen sollen, übertragen werden.

Für zukünftige User-Interface-Elemente, die weitere Attribute und eventuell verschiedenartige Zusatzinformationen benötigen, können zu einem späteren Zeitpunkt Subklassen der X-Attribut-Klasse eingeführt werden.

Die folgende Abbildung stellt das UML-Klassendiagramm für die Klasse „XAttribute“ vor.

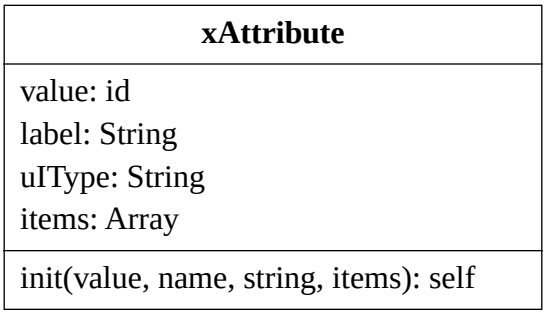


Abbildung 44: UML-Diagramm für die XAttribute-Klasse

1. Neben den angegebenen Attributen sollte das X-Attribut-Objekt über eine „init“-Methode verfügen mit der es „in einem Schritt“ erzeugt werden kann. Auf diese Weise kann die

entsprechende Methode im eXBO („attributesToEdit“), die als Rückgabewert das XAttribute-Array aufweist, in wenigen Zeilen bzw. sogar als „Einzeiler“ implementiert werden. Zu den wichtigsten Eigenschaften des XAttribute-Objektes, das (statt des ursprünglichen Attributes des eXBO) eigentlich in der eXBO-Edit-Komponente bearbeitet wird, gehört, daß es in der Lage ist, den ursprünglichen Attributwert im eXBO ständig aktuell zu halten. In Programmiersprachen, die eine entsprechende Objektreferenzierung ermöglichen, kann dies durch entsprechend durchgeführte „Zeigerverschiebungen“ gelöst werden. Für Programmiersprachen, die diese Möglichkeit nicht aufweisen, muß die Zugriffsmethode mit der des „value“-Attributes im XAttribute verändert wird, implizit die entsprechende Zugriffsmethode im eXBO aufrufen. Detaillierte Informationen über entsprechende technische Details einzelner Programmiersprachen können der Fachliteratur, wie z.B. [137] entnommen werden.

In der folgenden Abbildung 45, dem UML-Klassendiagramm für die eXBO-Edit-Komponente sind die Attribute und Methoden aufgeführt, die mindestens benötigt werden, um die beschriebenen Funktionen der eXBO-Facility Edit erfüllen zu können.

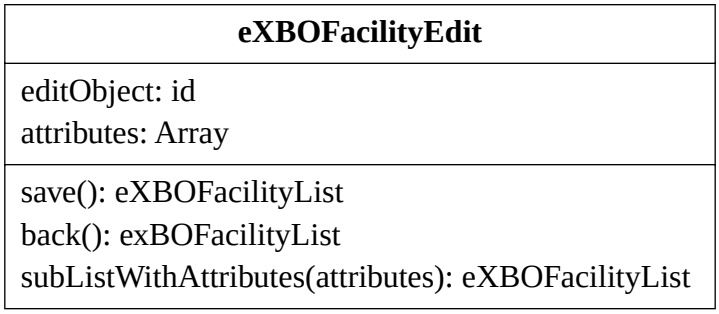


Abbildung 45: UML-Klassendiagramm für die eXBO-Facility Edit

1. editObject: Das zu editierende Objekt. Da die Klasse des Objektes nicht im voraus bestimmt werden kann, wird an dieser Stelle wieder die allgemeine Typbezeichnung „ID“ gewählt (siehe auch Abschnitt eXBO-Facility List).

2. attributes: Das Array, das die XAttributes enthält. Dieses Array wird durch den Aufruf der entsprechenden Methode des Edit-Objekts zurückgegeben („attributesToEdit“).

3. save: Das Anwählen des Speichernknopfes führt zur Ausführung der Safe-Methode, die dafür Sorge zu tragen hat, daß das Edit-Object, in welchem angeschlossenen Datenspeicher auch immer, gespeichert wird. Nach dem Anwählen der Safe-Funktion kann davon ausgegangen werden, daß der Benutzer seine Arbeit an diesem Edit-Object abgeschlossen hat. Dementsprechend wird die eXBO-List-Komponente zurückgegeben bzw. als Nächstes

angezeigt.

4. back: Diese Funktion wird ausgeführt, wenn der Benutzer den Zurückknopf anwählt. Bereits vorgenommene Änderungen am Attribute des Edit-Objects werden nicht gespeichert und es wird sofort zur eXBO-List-Komponente zurückgekehrt.

5. subListWithAttributes: Werden, wie im Beispiel von Abbildung 43, zusätzliche Arrays für ein Edit-Object verwaltet, wie im Beispiel die Multimedia-Daten bzw. Multimedia-Objekte für den Termin, so wird durch das Anwählen des entsprechenden Hyperlinks eine eXBO-List-Komponente aufgerufen und ihr die darzustellenden Objekte übergeben.

Mit der nachfolgenden Abbildung 46 soll die Funktionsweise der dynamischen HTML-Oberfläche der eXBO-Edit-Komponente erläutert werden.

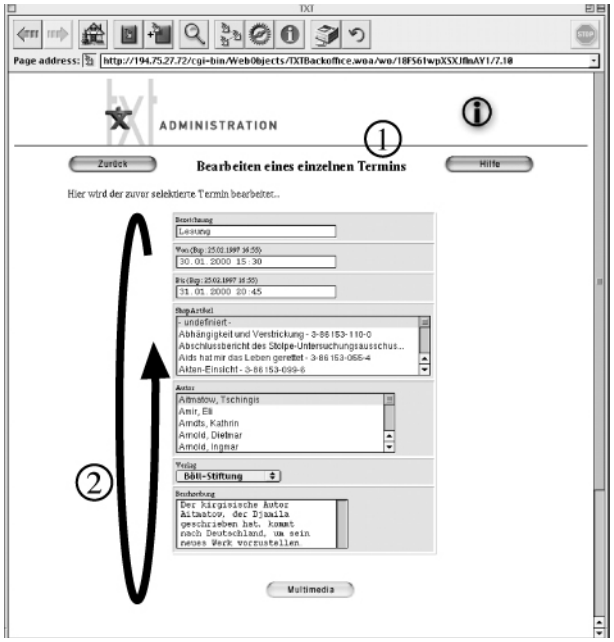


Abbildung 46: Dynamische HTML-Oberfläche der eXBO-Edit-Komponente

1. Wie zuvor bei der eXBO-List-Komponente wird dieser dynamische String durch die Abfrage der entsprechenden „name“-Methode des „editObjects“ aufgelöst.

2. Die Anzahl der Einzelelemente des Repetition-Attributs ergibt sich aus dem „attributesArray“, das von der Methode „attributesToEdit“ des jeweiligen eXBOs zurückgegeben wird. Jedes einzelne Element des Attributs-Array wird der Reihe nach

abgearbeitet. Durch die Abfrage des User-Interface-Typs des jeweiligen XAttributes (das „attributesArray“ ist wie weiter oben beschrieben mit XAttribute-Objekten gefüllt) kann das jeweilige User-Interface-Element ermittelt werden. Für jedes User-Interface-Element existiert innerhalb des HTMLs ein sogenanntes „WOConditional“, das mit einem IF-Statement vergleichbar ist.

Wird ein bestimmtes User-Interface-Element erkannt, so werden die benötigten Parameter vom XAttribute abgefragt (z.B. die „items“, bzw. die Angaben die in einer Choice zur Verfügung gestellt werden sollen) und die entsprechenden, dynamischen User-Interface-Elemente angezeigt.

5.4 eXBO-Facility Search

Die eXBO-Facility Search, die nachfolgend verkürzt als eXBO-Search-Komponente bezeichnet wird, weist eine HTML-Oberfläche auf, die der eXBO-Edit-Komponente auf dem ersten Blick sehr ähnlich ist. In der nachfolgenden Abbildung 47 ist eine Instanz einer eXBO-Search-Komponente abgebildet:

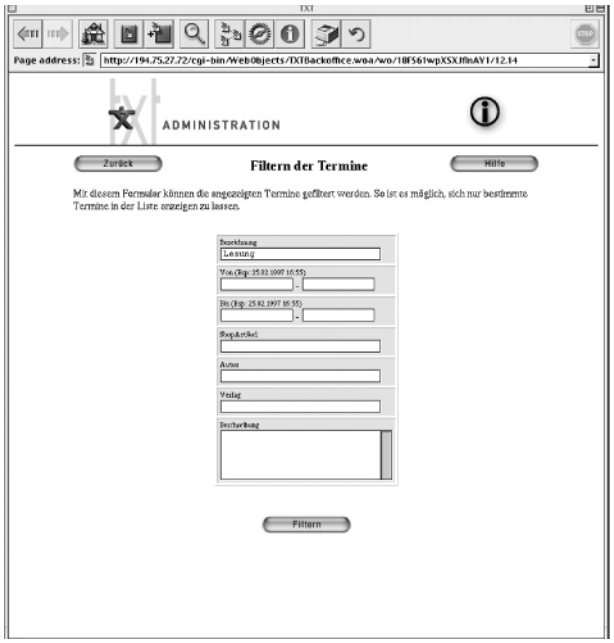


Abbildung 47: eXBO-Facility Search

Die eXBO-Such-Komponente weist für jedes Attribut ein Suchfeld auf, das ein gezieltes Suchen bzw. Filtern nach jedem einzelnen Attribut ermöglicht. Für Attribute, die numerische Größen oder Zeitpunkte enthalten, werden jeweils zwei Suchfelder eingeblendet, die eine „von-bis“-Suche ermöglichen. Die eXBO-Such-Komponente kommt meist dann zum Einsatz, wenn der Benutzer in der eXBO-List-Komponente ein gesuchtes Objekt nicht auf Anhieb findet oder wenn in der eXBO-List-Komponente nur Objekte mit bestimmten Eigenschaften angezeigt werden sollen. Durch das Anwählen der Funktion „Filtern“ bzw. „Suchen" wird das gesamte, zuvor von der eXBO-List-Komponente übergebene „listArray“ (siehe Abschnitt eXBO-Facility List), nach den entsprechenden Attributen gefiltert. Dabei werden "*" und sonstige Regular Expressions erkannt. Im Falle von mehreren ausgefüllten Suchfeldern werden diese mit einer AND-Funktion verknüpft. Nach dem Abschluß der Suche wird das gefilterte Array an die eXBO-List-Komponente übergeben und dort dargestellt. In der Abbildung 48 wird das UML-Klassendiagramm der eXBO-Search-Komponente vorgestellt.

eXBOFacilitySearch
list: Array
search(searchParameter): eXBOFacilityList

Abbildung 48: UML-Klassendiagramm für die eXBO-Facility Search

1. list: Das „listArray“ wird vor dem Aufruf der eXBO-Such-Komponente von der eXBO-List-Komponente übertragen. Die späteren Such- bzw. Filteraufgaben werden auf den Objekten, die in diesem „listArray“ enthalten sind, durchgeführt.

2. search: Das Anwählen der Funktion „Filtern“ (vgl. Abb. 39) ruft diese Methode auf, die den eigentlichen Filtervorgang anstößt, das gefilterte „listArray“ an die eXBO-List-Komponente übergibt und diese zur Anzeige in der User-Oberfläche zurückgibt. Der Aufbau und die Funktionsweise der dynamischen HTML-Oberflächen-Komponente der eXBO-Such-Komponente ist der der eXBO-Edit-Komponente sehr ähnlich, so daß an dieser Stelle auf eine nähere Betrachtung verzichtet werden kann.

5.5 eXBO-Facility Info

Die eXtended Business-Objects-Facility Info, die nachfolgend verkürzt eXBO-Info-Komponente genannt wird, hat die Aufgabe einen kompakten zusammenfassenden Überblick über ein Business-Object und seine Attribute zu geben. Wie im Abschnitt eXBO Facility Edit gesehen, wird in der eXBO-Edit-Komponente z.B. bei Bildern lediglich der Pfadname angegeben und bei Array-Attributen (wie z.B. bei den zur Verfügung stehenden Multimedia-Daten) auf eine weitere eXBO-List-Komponente verwiesen. Im Gegensatz zur eXBO-Info-Komponente eignet sich die eXBO-Edit-Komponente nicht dazu, einen Sicherheits-Check bzw. ein „Preview“ für ein Business-Object durchzuführen, bevor es zum Beispiel im Storefront eines Shop-Systems angezeigt wird. In der Praxis (siehe z.B. das Portal-System für Buchverlage „TXT“ in Kapitel 8) wird die Info-Page häufig auch als Vorlage für einen Papierausdruck eingesetzt, der dann bei entsprechenden Gesprächen als Diskussions-

grundlage dienen kann. In anderen Fällen, wie z.B. bei dem Prototypen Factory-Circus (vergleiche Kapitel 8), wird eine Sub-Klasse der eXBO-Info-Komponente genutzt, um ausdruckbare Lieferbestätigungen und Rechnungen zu erzeugen. In Abbildung 49 ist eine mögliche Instanz der eXBO-Info-Komponente dargestellt.

Abbildung 49: eXBO-Facility Info

Wie in Abbildung 49 zu erkennen, werden die Attribute eines eXBOs in der eXBO-Info-Komponente nur dargestellt. Es existiert keine Möglichkeit diese zu bearbeiten. In Abbildung 50 ist das UML-Klassendiagramm der eXBO-Info-Komponente dargestellt.

eXBOFacilityInfo
object: id
back(): eXBOFacilityList

Abbildung 50: UML-Klassendiagramm für die eXBO-Facility Info

1. object: Das Attribut „object“ wird vor dem Aufruf der eXBO-Info-Komponente von der eXBO-List-Komponente gesetzt, und zwar mit dem entsprechenden „selectedObject“-Attribut aus der eXBO-List-Komponente. Wie zuvor wird auch an dieser Stelle die Typbezeichnung

„ID“ für zuvor nicht definierbare Objekt-Typen gewählt.

2. back: Die einzige Methode die eine eXBO-Info-Komponente aufweisen muß, ist die „back“-Methode mit der wieder zur List-Komponente zurückgekehrt werden kann. Funktionen, wie z.B. das Ausdrucken der eXBO-Info-Komponente werden von dem jeweiligen Web-Browser übernommen. Das gleiche gilt für das Abspielen von Videos, das Anzeigen von Bildern usw.

Aufbau und Funktionsweisen der dynamischen HTML-Oberfläche der eXBO-Info-Komponente sind der der eXBO-Edit-Komponente sehr ähnlich, so daß auf eine nähere Betrachtung an dieser Stelle verzichtet wird. Die wesentlichen Unterschiede, wie z.B. die Darstellung eines Bildes statt der bloßen Anzeige des Pfades und die zusammenfassende Darstellung auch von Sub-Arrays, wie z.B. Multimedia-Daten, wurden bereits eingangs erwähnt.

5.6 Zusammenfassung eXtended Business-Object-Facilities

In der Einleitung zu diesem Kapitel wurde bereits auf die Einsparungspotentiale durch die Verwendung von eXBO-Facilities eingegangen. An dieser Stelle sollen die Anforderungen an bzw. die nötigen Erweiterungen von Business-Objects noch einmal zusammenfassend dargestellt werden.

Damit „normale“ Business-Objekte von eXBO-Facilities bearbeitet werden können, benötigen sie lediglich drei zusätzliche Methoden:

- „`headers`“: Die Methode „headers“, die ein Array mit String-Objekten zurückgibt, in denen die gewünschten Spaltenüberschriften in der eXBO-List-Komponente enthalten sind.

- „`values`“: Die Methode „values“, die ein Array mit String-Objekten zurückgibt, die den Zugriffsmethoden für die Attribute, die in den Spalten der eXBO-List-Komponente dargestellt werden, entsprechen.

- „`attributes`“: Die Methode „attributes“ wird von der eXBO-List-Komponente aufgerufen. Sie soll ein Array mit den zu bearbeitenden Attributen enthalten. Wie in Abschnitt eXBO-Facility Edit erläutert, wird dazu jedes Attribut des Business-Objects zuvor in ein XAttribute „verpackt”, womit die Übertragung der gewünschten User-Interface-Elemente und der eventuell benötigten Parameter ermöglicht wird.

Um den verhältnismäßig geringen Mehraufwand bei der Implementierung von eXBO zu verdeutlichen, sind die angesprochenen Methoden nachfolgend einmal in einer konkreten Programmiersprache angegeben. Der Source-Code wurde der real existierenden eCommerce-Anwendung „MetaShop“ entnommen (vergleiche Kapitel 8). Diese Anwendung wurde mit der Entwicklungsumgebung WebObjects von Apple (vormals NeXT) unter Einsatz der Programmiersprache ObjectiveC entwickelt. Der abgebildete Source-Code ist dem eXBO „Termin“ entnommen.

```
- (NSArray *) attributes
{
return [NSArray arrayWithObjects: xAttributeName, xAttributeDa-
   teStart, xAttributeDateEnd, xAttributeBeschreibung, nil];
}

- (NSArray *) headers
{
return [NSArray arrayWithObjects: Bezeichnung, Veranstaltungsbeginn,
   nil];
}

- (NSArray *) values
{
return [NSArray arrayWithObjects: name, dateStart, nil];
}
```

Abschließend werden in Abbildung 51 die verschiedenen eXBO-Facility-Komponenten, die zusammen das „Manager-Muster“ ergeben, noch einmal zusammenfassend dargestellt. Die Menu-Page-Komponente, die keine eXBO-Facility-Komponente ist, wurde zur Verdeutlichung mit in das Diagramm genommen, da sie der Ausgangspunkt der meisten Aktionen im „Manager-Muster“ ist.

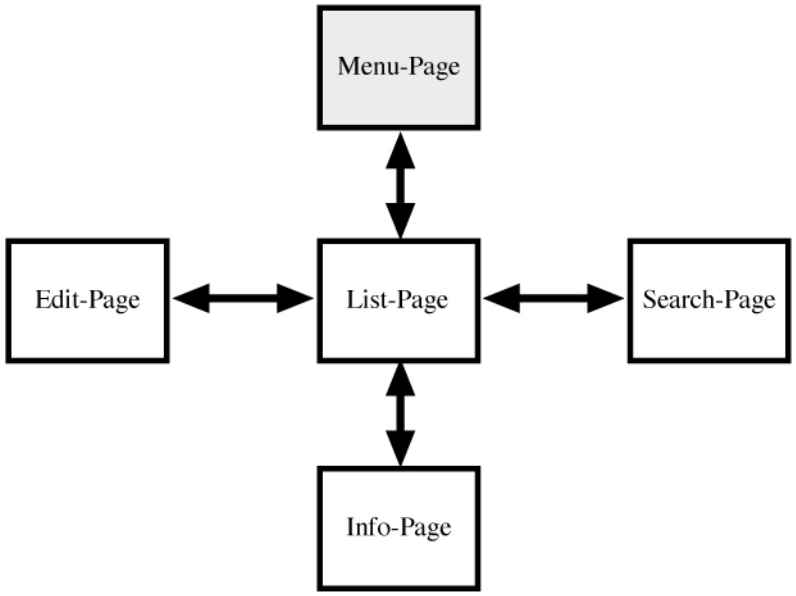


Abbildung 51: eXBO-Facilities des Manager-Musters

Kapitel 6

eXtended Business-Objects

Im ersten Kapitel wurden die beiden wesentlichen konzeptionellen Erweiterungen vorgestellt, die eXBO von normalen Business-Objects unterscheiden. Die erste Erweiterung „Wissen über die Verarbeitung“, die es den eXBO ermöglicht, mit eXBO-Facility-Komponenten verarbeitet zu werden, wurde im vorangegangenen Kapitel ausführlich behandelt. Die wenigen Methoden, um die ein „normales“ Business-Objects erweitert werden muß, um die entsprechenden Eigenschaften aufzuweisen, wurden im Detail erläutert. Ebenso wurde auf die Einsparungspotentiale durch die Verwendung von eXBO-Facilities eingegangen.

In diesem Kapitel soll nun zunächst auf die zweite konzeptionelle Erweiterung eingegangen werden, bevor anschließend eine Reihe von konkreten eXBO vorgeschlagen wird. Die zweite konzeptionelle Erweiterung die in der Einleitung mit dem Stichwort „Wissen über die Darstellung“ bezeichnet wurde, ermöglicht es einem eXBO in jedem bzw. mit jedem beliebigen Endgerät optimal präsentiert zu werden. Die meisten der heutigen eCommerce-Anwendungen und der entsprechenden Überlegungen bezüglich Business-Objects und Frameworks gehen implizit davon aus, daß die jeweiligen Anwendungen bzw. Objekte mit Web-Browsern auf PC-Monitoren dargestellt werden. Implizit werden dabei vorausgehende Annahmen bezüglich der Übertragungskapazitäten, Bildschirmauflösungen und Bildschirm-größen gemacht und versucht die Darstellung für diese Art von Endgerät zu optimieren. Die Mehrheit diesbezüglicher Untersuchungen von führenden Marktforschungsinstituten (siehe z.B. [1-4]), sagen voraus, daß diese „Monokultur“ bezüglich der eingesetzten Endgeräte bei

der Verwendung von eCommerce-Anwendungen schon in naher Zukunft nicht mehr in dieser Form vorhanden sein wird. Schon heute existieren weltweit mehr mobile Endgeräte (z.B. in Form von Handys) als PCs mit Internet-Anschluß [1-4], [67-69]. Die ständig wachsende Verbreitung von mobilen Computern (Laptops), von miniaturisierten mobilen Computern (PDAs), die aktuellen Entwicklungen im UMTS-Umfeld, die Einführung eines digitalen, internetfähigen TV-Standards und vieles mehr lassen diese Prognosen nachvollziehbar erscheinen.

Als Reaktionen auf den prognostizierten (und dann doch verpufften) WAP-Hype wurden bereits eCommerce-Shop-Systeme geschaffen, die für die Bedienung auf einem WAP-Handy optimiert sind [15]. Diese Entwicklungen zeigen in die prognostizierte Richtung, unterscheiden sich aber von den in dieser Arbeit vorgestellten Konzepten in dem fehlenden ganzheitlichen Ansatz.

Mit Hilfe der in dieser Arbeit vorgestellten eXBO soll es möglich werden eCommerce-Anwendungen zu konstruieren, die gleichzeitig für verschiedenste heutige und zukünftige Endgeräte optimiert sind.

Mit den vorgestellten eXBO kann darüber hinaus der Grundstein für medienübergreifende eCommerce-Anwendungen gelegt werden, die ebenfalls die Herstellung von hochqualifizierten (gedruckten) Katalogen ermöglichen. Dabei werden die Anwendungen bzw. „Endgeräte“ der Druckindustrie einfach als ein weiteres mögliches „Endgerät“ behandelt. Selbst in Ländern wie der Bundesrepublik Deutschland, in denen die neuen elektronischen Kommunikationswege, wie Mobiltelefon und Internet, bereits eine sehr große Verbreitung besitzen, haben gedruckte (Papier)-Kataloge noch immer eine enorm große Bedeutung für Handel und Industrie. In vielen Bereichen sind gedruckte Kataloge noch immer weit wichtiger als eCommerce-Anwendungen. Um so erstaunlicher ist es, daß es bisher nur sehr wenige Entwicklungen, gibt, die danach streben, bereits vorhandene elektronische Kataloge (z.B. in einem eCommerce-Shop-System) automatisiert in professionelle Vorlagen für die Druckvorstufe umzusetzen.

Wie zuvor bei den eXBO-Facility-Komponenten stand es auch bei der Erarbeitung der Konzepte, die diese „endgeräteunabhängigen, universellen eCommerce-Anwendungen“ ermöglichen, im Vordergrund, möglichst einfache, überschaubare Ideen zu entwerfen, die mit geringstem Mehraufwand und einer kleinstmöglichen Anzahl von zusätzlichen Objekten umzusetzen sind. Das in dieser Arbeit vorgeschlagene Konzept, das wie zuvor im wesentlichen nur drei Erweiterungen vorsieht, wird im nachfolgend erläutert.

6.1 Description-, Picture- und Multimedia-"Manager"

In diesem Abschnitt soll anhand eines beispielhaften eXBO „Warengruppe“, das zweite Konzept „Wissen über die Darstellung“ erläutert werden (vgl. hierzu auch Kapitel 1). In diesem Zusammenhang werden ebenfalls die unterstützenden Objekte, die Description-, Picture- und Multimedia-Manager vorgestellt.

6.1.1 eXBO Warengruppe

Das eXBO Warengruppe wird genutzt, um Online-Kataloge aufzubauen, bei denen verschiedene Artikel in die jeweiligen Warengruppen bzw. Unterwarengruppen bzw. Rubriken einsortiert werden können. Zu diesem Zweck benötigt ein eXBO Warengruppe verschiedene Attribute, wie z.B. ein „Artikel-Array“, in das Artikel einsortiert werden können, eine Attributbezeichnung, um der Warengruppe einen Namen geben zu können und ein Attribut „Oberwarengruppe“, mit dessen Hilfe eine hierarchische Struktur aufgebaut werden kann. In diesem Abschnitt jedoch sollen vorrangig diejenigen, möglichen Attribute eines eXBO betrachtet werden, die bei der Darstellung in unterschiedlichen Endgeräten zu Problemen führen können, und deshalb gesondert behandelt werden sollten.

Als Paradebeispiel für ein solches Attribut kann das Attribut „Foto“ bzw. „detailPicture“ bzw. „Abbildung“ angeführt werden. Ein entsprechendes „detailPicture“ (eine größere, detailliertere Darstellung oder eine symbolische Abbildung zur Visualisierung der Warengruppe) wird in „normalen“ eCommerce-Systemen für die Darstellung in einem Web-Browser auf einem gängigen PC-Monitor optimiert. Wobei Bildgröße und -schärfe auf die entsprechende Bildschirmauflösung und die entsprechende Bildschirmgröße abgestimmt sind. Zusätzlich wurde die Bildqualität unter Rücksichtnahme auf typische Internet-Übertragungsgeschwindigkeiten („Download-Zeiten“) auf das geringste vertretbare Maß reduziert. Ein derart „spezialisiertes“ Bild ist für eine optimale Präsentation in anderen Medien bzw. Endgeräten ungeeignet. Für die Herstellung von gedruckten Katalogen genügt beispielsweise die Bildqualität nicht. Für die Darstellung in einem mobilen Kleincomputer

(z.B. einem heutigen PDA) ist das Bild viel zu groß. Für die Darstellung in einem WAP-Handy, das allenfalls minimierte Symbole darstellen könnte, ist es vollkommen ungeeignet, ganz abgesehen davon, daß die Übertragungszeiten unangemessen lang wären. Für die Darstellung auf einem TV-Monitor, mit normalerweise weit geringerer Auflösung, ist es ebenfalls „zu groß“.

Um eine „optimale“ Darstellung eines solchen „detailPicture“ in verschiedenen Endgeräten zu ermöglichen, existieren generell zwei verschiedene Ansätze:

- Das vorhandene „detailPicture“ wird von entsprechenden Umwandlungs- bzw. Verarbeitungskomponenten „modelliert“ und so an das jeweilige Endgerät angepaßt.

- Für jedes Endgerät wird ein eigenes, spezialisiertes „detailPicture“ zur Verfügung gestellt.

Beide Methoden haben ihre Vor- und Nachteile. Die Umwandlung bzw. Umrechnung von vorhandenen Bilddaten kann z.B. derart zeit- und rechenintensiv werden, daß die Gesamtperformance eines Systems gefährdet wird. Für die Datenreduktion bzw. Verkleinerung eines Bildes stehen gängige Verfahren zur Verfügung, der umgekehrte Weg jedoch, aus einem qualitativ schlechten Bild ein hochwertiges, hochauflösendes Bild zu produzieren, ist bisher ungleich seltener begangen worden. Dies hat bei einem solchen Ansatz zur Folge, daß eigentlich immer das Bild mit der größten Datenmenge eingesetzt werden muß.

Andererseits ist das zur Verfügung stellen von zahlreichen, verschiedenen, spezialisierten Bildern eine zeitaufwendige Angelegenheit, die zudem ein gewisses technisches Know-How erfordert, das nicht von allen „Business-Anwendern“ erwartet werden kann. In der Praxis führt dies beispielsweise oft dazu, daß allein aus reinem Zeitmangel die gegebenen Möglichkeiten nicht voll genutzt werden und es zu Darstellungslücken für weniger beachtete Endgeräte kommt.

Die Diskussion über die verschiedenen Vor- und Nachteile soll an dieser Stelle nicht weitergeführt werden, da das in dieser Arbeit vorgestellte Konzept beide Ansätze miteinander vereint. Die konkrete Implementierung, der Umsetzung des vorgestellten Konzeptes kann individuell variiert werden und somit im Einzelfall entschieden werden, in welchem Verhältnis die Eigenschaften der jeweiligen Ansätze genutzt werden sollen. Um die endgeräteunabhängige Darstellung von Attributen, wie „detailPicture“ zu ermöglichen, werden im folgenden ein weiteres mal drei Erweiterungen vorgeschlagen, um die ein Business-Object ergänzt werden sollte (vergleiche Kapitel 5). Diesmal um die endgeräteunabhängige Darstellung zu ermöglichen:

1. Statt eines einzelnen Attributes „detailPicture“ wird für jedes zu erwartende Endgerät ein spezialisiertes Attribut eingeführt (z.B. „detailPictureForWeb“, „detailPictureForPDA“, „detailPictureForDigitalTV“, „detailPictureForPrint“ usw.).

2. Jedes spezialisierte Attribut soll, neben den normalen get- und set-Zugriffsmethoden, eine zugehörige best-Zugriffsmethode besitzen. Ein Attribut „detailPictureForPDA“ besitzt dann beispielsweise die Zugriffsmethode „bestDetailPictureForPDA“, das Attribut „detailPictureForWeb“ besitzt die zusätzliche Methode „bestDetailPictureForWeb“ usw. Die Rückgabewerte dieser Methoden sind nicht, wie bei den „normalen“ get-Zugriffsmethoden „fest verdrahtet“, sondern werden dynamisch von einem „PictureManager“ ermittelt (siehe unten).

3. Das neu eingeführte Objekt „PictureManager“ versucht zunächst das entsprechende spezialisierte „detailPicture“ des Objektes zurückzugeben. Ist dieses nicht vorhanden, kann es regelbasiert das nächstbeste „detailPicture“ laden, dieses Umrechnen bzw. Anpassen und zurückgeben. Ist das nächstbeste „detailPicture“ ebenfalls nicht vorhanden, versucht es das darauffolgende zu laden usw. Ist es überhaupt nicht möglich, ein entsprechendes Bild zu finden (wenn z.B. gar keines vorhanden ist), ist es möglich, durch den „PictureManager“ wenigstens ein, für das entsprechende Endgerät, spezialisiertes „defaultPicture“ zurückzugeben.

Diese drei Erweiterungen lassen sich auch auf andere problematische Attribute übertragen. Zu solchen „problematischen Attributen“ gehören alle Sorten von Multimedia-Daten, wie Animationen, Musik, Videos usw. und ebenfalls längere Beschreibungen von Business Objekten (z.B. eine längere Beschreibung der angegebenen Warengruppe). Ein entsprechender „DescriptionManager“ würde z.B. in einer Methode „bestDescriptionForPDA“ zunächst versuchen die „descriptionForPDA“ zu laden und zurückzugeben. Ist diese nicht vorhanden, versucht er nach seinem internen Regelwerk eine andere „description“ auszuwählen und diese z.B. entsprechend zu kürzen. In einem anderen Szenario, z.B. bei der Herstellung von gedruckten Katalogen, könnte der „DescriptionManager“ beispielsweise eine Warnung erzeugen, wenn der gewünschte Text „descriptionForPrint“ nicht vorhanden ist und die ausgewählte Ersatz-Description (z.B. „descriptionForWeb“) zu lang ist und es so zu einem ungewollten Seitenumbruch im Katalog kommen würde.

Im Rahmen dieser Arbeit soll die Vorstellung des Konzeptes und die Einführung der drei Manager: PictureManager, MultimediaManager und DescriptionManager genügen. Je nach Einsatzgebiet und Anwendungsfall können diese Manager beliebig komplexe Regelwerke und Rechenvorschriften enthalten, auf die aber im Rahmen dieser Arbeit nicht eingegangen werden soll. Bei der Implementierung der Prototypen (siehe Kapitel 8) wurden ebenfalls

zunächst nur einfache Manager implementiert. Sie sind erst einmal nur in der Lage, das jeweils nächstbeste Attribut auszuwählen und geringfügige Anpassungen vorzunehmen (z.B. ist der DescriptionManager in der Lage, zu lange Texte zu kürzen). In der folgenden Abbildung wird abschließend das UML-Klassendiagramm für einen PictureManager dargestellt. Die anderen Manager weisen einen entsprechenden Aufbau auf und werden im folgenden daher nicht näher erläutert:

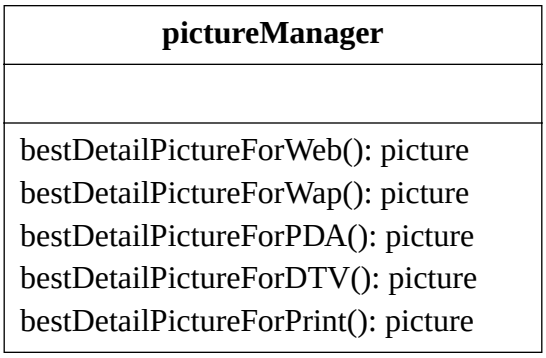


Abbildung 52: UML-Klassendiagramm für den PictureManager

Diese "bestFor"-Methoden werden innerhalb der gleichnamigen „bestDetailFor..."-Methoden des eXBO aufgerufen und fordern den PictureManager auf, ein entsprechendes spezialisiertes Bild zu ermitteln, oder zu generieren und zurückzugeben. Als Parameter übergibt dabei das aufrufende eXBO eine Referenz auf sich selbst. Erst dadurch kann der allgemeingültige und von allen eXBO nutzbare PictureManager gezielt die Attribute des richtigen (des aufrufenden) Objektes abfragen.

Bevor nun auf den folgenden Seiten mit der Vorstellung einiger ausgewählter eXBO begonnen wird, soll an dieser Stelle darauf hingewiesen werden, daß die vorgeschlagenen Business-Objects nicht den Anspruch auf Vollständigkeit oder universellster Verwendbarkeit in verschiedensten Geschäftsanwendungen erheben. Dies ist eine der logischen Schlussfolgerungen aus der Beobachtung der „Business Object Domain Task Force“ der OMG, die genau an diesem selbstgesteckten Ziel scheiterte. Universelle Business-Objects zu definieren, die allen Geschäftsbereichen und Industrien gerecht werden, stellte sich als eine nur selten lösbare Aufgabe heraus. Für mehr Informationen zu OMG-Business Objects siehe [102-115]. Besonders zu empfehlen ist das praxisnahe Buch von Peter Eeles und Oliver Sims "Building Business Objects" [110].

Innerhalb dieser Arbeit soll deshalb nicht der gleiche Fehler begangen werden und der Anspruch auf universell definierte Business-Objects erhoben werden. Durch das in Kapitel 5

vorgestellte Konzept, das es ermöglicht mit geringstem Aufwand neue oder veränderte Business-Objects zum Einsatz zu bringen, wurde vielmehr auf die entsprechenden Erkenntnisse reagiert, nämlich das Business-Objects eben meist nicht allgemein gültig definierbar sind und von Geschäftsanwendung zu Geschäftsanwendung Variationen aufweisen und jeweils neu beziehungsweise leicht verändert definiert werden müssen.

Die vorgestellten eXBO sind vorwiegend aus den in Kapitel 8 vorgestellten Prototypen entnommen. Sie kommen dort im Rahmen von eCommerce-Portal- und Shop-Systemen zum Einsatz. Natürlich existieren neben dem hier vorgestellten eXBO noch viele hundert andere Business-Objects für die verschiedensten Anwendungsbereiche. Diese ausfindig zu machen und zu definieren ist nicht Inhalt dieser Arbeit. Wie schon in der Definition der Business-Objects angegeben sollte dies in interdisziplinärer Zusammenarbeit während der Entwicklung von Geschäftsanwendungen mit den Experten aus den jeweiligen Geschäftsbereichen durchgeführt werden (vergleiche Kapitel 1 und Kapitel 3).

6.2 eXBO Termin

Das eXBO Termin kann, wie im Falle des Portal-Systems für Buchverlage TXT (siehe Kapitel 8) genutzt werden, um eine Veranstaltung, wie z.B. eine Buchlesung oder eine Preisverleihung bekannt zu geben. Zu den wichtigsten Attributen gehören Veranstaltungsort, Datum von, Datum bis, die Bezeichnung der Veranstaltung (nachfolgend „descriptionShort“) und ein beschreibender Text (nachfolgend „descriptionLong“). Zusätzlich können Informationen über den Veranstalter, ein Foto vom Veranstaltungsort und diverse Multimedia-Daten (z.B. ein Fotoalbum oder Videomitschnitte der Veranstaltung), die Informationen abrunden. Für die Attribute „descriptionLong“, die Multimedia-Daten und das Veranstaltungsfoto, die für verschiedene Endgeräte optimiert sein sollten, werden die entsprechenden „Manager“ (DescriptionManager, PictureManager und MultimediaManager) vorgesehen und entsprechende Attribute und Zugriffsmethoden definiert. In der nachfolgenden Abbildung ist das UML-Klassendiagramm für das eXBO Termin angegeben:

Abbildung 53: UML-Klassendiagramm für das eXBO Termin

1. anmerkungen bis veranstalter: In diesem Bereich sind die „normalen“ Attribute angegeben.

2. longDescriptionForWeb bis pictureForPrint: Dies sind die spezialisierten Attribute die für die Optimierung der Darstellung in unterschiedlichen Endgeräten genutzt werden.

3. decriptionManager bis multimediaManager: Dies sind die entsprechenden „Manager“, die das eXBO bei der Optimierung unterstützen.

4. bestDescriptionForWeb bis bestPictureForPrint: Die Zugriffsmethoden für die optimierbaren Attribute. Die anderen Zugriffsmethoden für die „normalen“ Attribute wurden zugunsten einer besseren Übersicht nicht abgebildet.

5. headers, values, attributes: Hier sind die drei Methoden angegeben, die eine Verarbeitung mit den eXBO-Facility-Komponenten ermöglichen (vergleiche dazu Kapitel 5).

Wie alle eXBO kann auch das eXBO Termin mit den eXBO-Facilities List, Edit, Search und Info bearbeitet werden. Gemäß der Rollenverteilung (siehe Kapitel 4) werden deshalb zunächst keinerlei Komponenten für die Bearbeitung des Business-Objects Termin im Backoffice eines eCommerce-Systems benötigt (Spezialanforderungen ausgenommen). Für den Storefront werden die nachfolgenden dynamischen HTML-Oberflächen vorgeschlagen, die in den Anwendungen TXT bzw. MetaShop (siehe auch Kapitel 8) für web-basierte Oberflächen umgesetzt wurden:

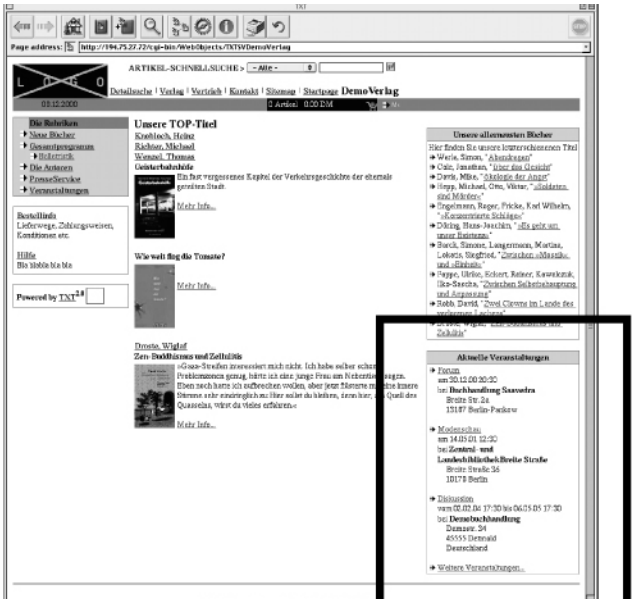


Abbildung 54: Termin-Box

Die wiederverwendbare Komponente Termin-Box eignet sich für die Plazierung auf einer Start- oder Übersichtsseite. In der Box wird eine variierbare Anzahl (im Beispiel 3), von Kurzbeschreibungen von Terminen aufgelistet. Die aktuellste Veranstaltung (mit der kürzesten Zeitspanne zum Veranstaltungsbeginn) wird als oberste dargestellt. Durch das Klicken auf den Veranstaltungs-Link springt der Benutzer zur „Termin-Detail-Ansicht“ (siehe weiter unten). Durch das Klicken des Links „weitere Termine hier“, springt der Benutzer zur Termin-List-Komponente. Die Termin-List-Komponente ist in Abbildung 55 dargestellt.

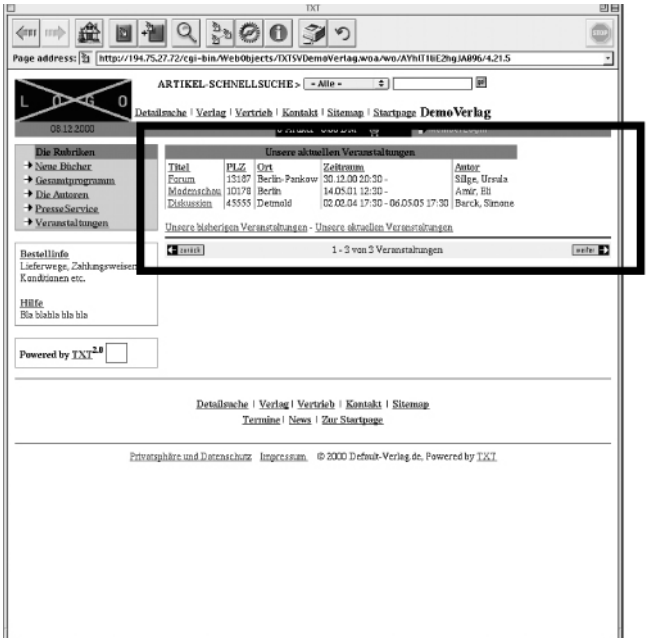


Abbildung 55: Termin-List-Komponente

In der Termin-List-Komponente werden alle Termine mit Bezeichnung, Ort und Zeitraum aufgelistet. Die Termine werden in dieser Beispielseite in 10er-Batches angezeigt (siehe auch Kapitel 5). Mit den Knöpfen „Zurück“ und „Weiter“ kann der Benutzer zum nachfolgenden 10er-Block weiterblättern. Mit Hilfe der Links „bisherige Veranstaltungen“und „aktuelle Veranstaltungen“ kann der Benutzer auch auf Informationen über bereits stattgefundene Veranstaltungen zugreifen.

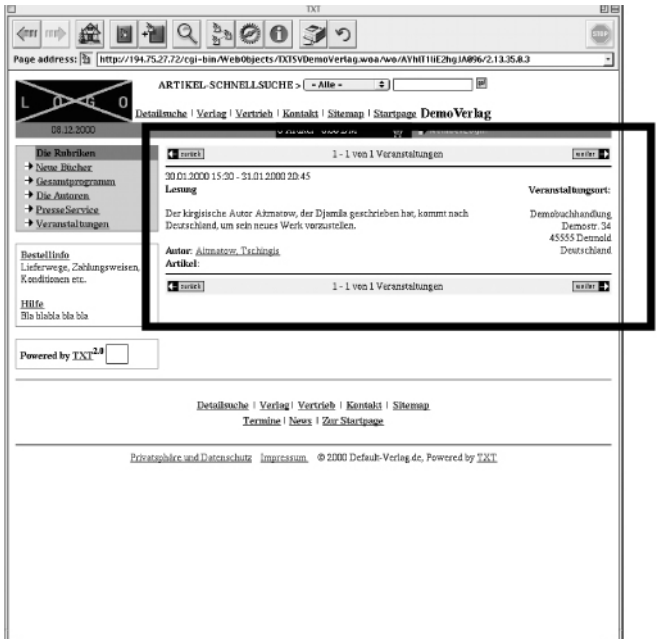


Abbildung 56: Termin-Detail-Komponente

In der Termin-Detail-Komponente wird der Termin mit all seinen Attributen im Detail angezeigt. (Ein Veranstaltungsfoto vom Veranstaltungsort und andere, zusätzliche Multimedia-Daten sind in diesem Beispiel nicht vorhanden und werden dementsprechend nicht dargestellt.)

Auf den internen Aufbau und die Funktionsweise der angegebenen dynamischen HTML-Komponenten, die durch einfache „Tags“ an jede beliebige Stelle eines HTML-Codes eingebracht werden können, wird an dieser Stelle nicht näher eingegangen. Die generelle Funktionsweise wurde bereits in den Kapiteln 2 und 5 erläutert. Die Komponenten Termin-Box, Termin-List und Termin-Detail sind in ihren Funktionsweisen dabei der eXBO-List-Komponente nachempfunden. Beim Bau der Prototypen wurden die Termin-Box-, Termin-List- und auch die Termin-Detail-Komponente aus der eXBO-List-Komponente abgeleitet bzw. sie sind sogar direkte Subklassen der eXBO-List-Komponente. Dem aufmerksamen Betrachter sind sicherlich die „Weiter“- und „Zurück“-Knöpfe in der Termin-Detail-Komponente aufgefallen, die auf die Verwendung der List-Komponente (mit Batch-Count gleich eins) schließen lassen. Für die „Verwandlung“ in die Termin-Detail-Komponente mußte lediglich das Design der HTML-Oberfläche angepaßt werden. Die Verwandtschaft der Termin-Box- und der Termin-List-Komponenten zur eXBO-List-Komponente ist auf den ersten Blick zu erkennen, obgleich in diesen Komponenten die „Weiter“- und „Zurück“-Knöpfe ausgeblendet wurden.

6.3 eXBO News

Das eXBO News kann für die Umsetzung der News-Funktionalität, wie sie aus den meisten Online-Anwendungen bekannt ist, genutzt werden. Die wichtigsten Attribute des eXBO News sind der News-Text selbst, die Überschrift, das Datum und der Autor. In der nachfolgenden Abbildung 57 ist das UML-Klassendiagramm für das eXBO News angegeben.

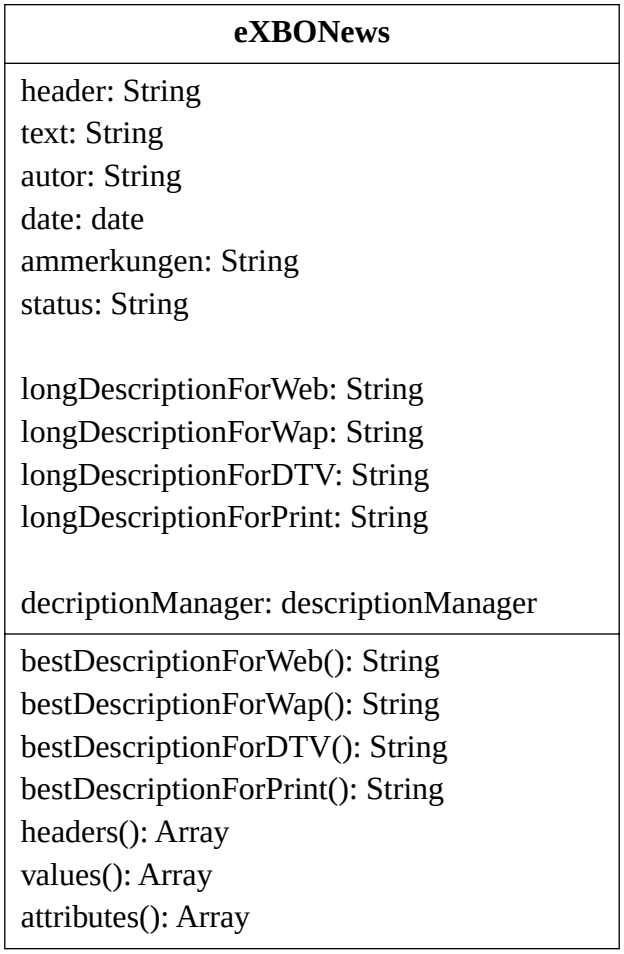


Abbildung 57: UML-Klassendiagramm für das eXBO News

1. header bis status: Zusätzlich zu den oben angegebenen gebräuchlichsten Attributen empfiehlt es sich, ein Statusfeld einzuführen, mit dem bei Bedarf z.B. veraltete News

ausgeblendet werden können, und ein Anmerkungsfeld für interne Anmerkungen vorzusehen.

2. longDescriptionForWeb bis longDescriptionForPrint: Lediglich der lange Text benötigt, wie die lange Beschreibung im Beispiel des eXBO Warengruppe weiter oben, eine Sonderbehandlung durch einen entsprechenden Manager.

3. bestDescriptionForWeb bis attributes: Die zusätzliche Methode für das „text“-Attribut und wie immer die drei Methoden für die eXBO-Facilities.

Wie bei allen eXBO werden für den Backoffice bzw. für das Content-Management-System keine weiteren Komponenten bzw. Facilities benötigt. Für den Storefrontbereich einer normalen Web-Anwendung, die für gängige PC-Monitore optimiert ist, werden die nachfolgenden dynamischen HTML-Oberflächen vorgeschlagen, die in den Anwendungen TXT bzw. MetaShop umgesetzt wurden:

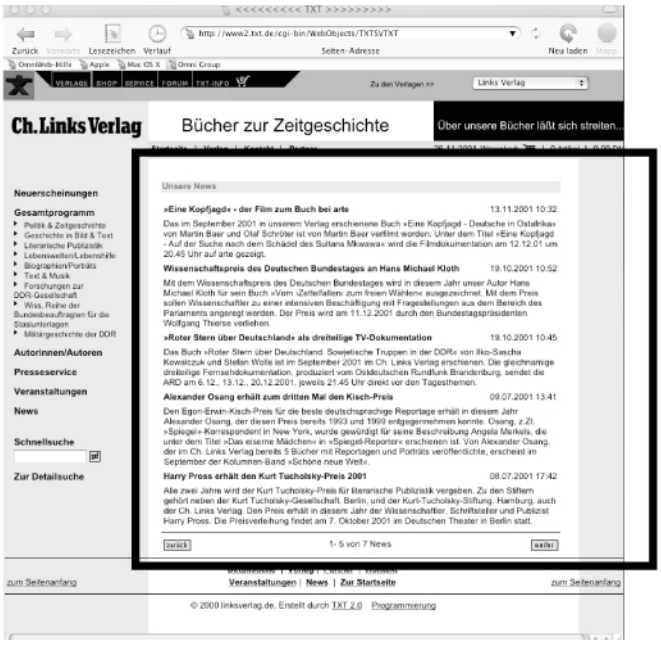


Abbildung 58: News-List-Komponente

In der News-List-Komponente werden alle aktuellen News, nach Datum geordnet, dargestellt. Wie leicht zu erkennen, wurde die entsprechende Funktionalität von der dynamischen HTML-Oberfläche der eXBO-List-Komponente geerbt.

Abbildung 59: News-Box

Die News-Box, in der lediglich die Überschriften und einige wenige News dargestellt sind, eignet sich für die Plazierung auf einer Startseite bzw. Begrüßungsseite. Mit dem Klick auf die zugehörige Nachricht gelangt der Benutzer zu der richtigen Stelle in der eXBO-List-Komponente (siehe oben).

6.4 eXBO Forum

Das eXBO Forum kann dazu genutzt werden, Kunden bzw. allgemein Nutzeranmerkungen zu sammeln und zur Diskussion zu stellen. Die oft unterschätzte Bedeutung solcher Foren, deren Funktionalität an die der „News-Groups“ angelehnt ist, wird in [4] eindrucksvoll beschrieben. Insbesondere wird dort auf die unternehmerische Relevanz gut organisierter und betreuter Foren als ein wichtiges Instrument der „After Sales“-Phase eingegangen, durch die langjährige Kundenzufriedenheit und Kundentreue erreicht werden kann.

Zu den wichtigsten Attributen des eXBO Forum gehören der Name des Autors, seine E-Mail, ein Datum und der eigentliche Text. Zusätzlich empfehlen sich die Attribute Status, mit dem eine entsprechende Moderation bzw. Filter-Funktion unterstützt werden kann und wie immer ein Attribut Anmerkungen für interne Zusatzinformationen. Ein Attribut „subject“ kann bei größeren Foren dazu benutzt werden nach Themenkreisen zu sortieren. In der nachfolgenden Abbildung 60 ist das UML-Klassendiagramm für das eXBO Forum abgebildet:

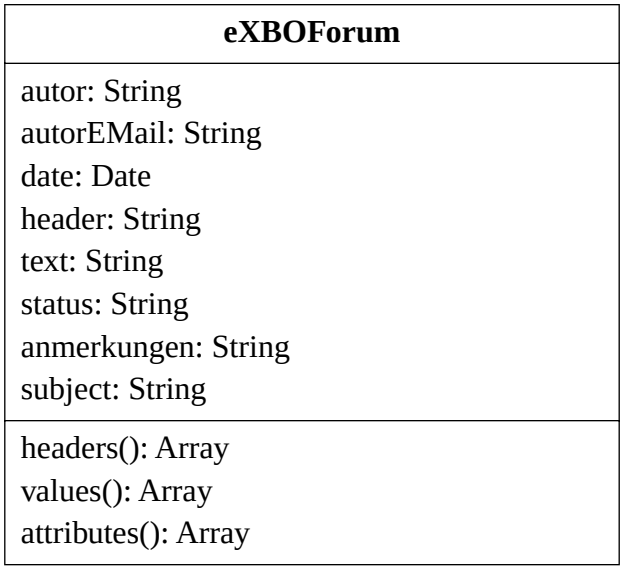


Abbildung 60: UML-Klassendiagramm für das eXBO Forum

Da es sich bei den entsprechenden Texten um Meinungen und Aussagen von Benutzern handelt, wird an dieser Stelle die Verwendung des Manager-Konzeptes nicht vorgeschlagen,

da dies eventuell zu ungewollten Veränderungen der Aussagen führen könnte. In Anwendungsfällen in denen auch das Forum endgeräteunabhängig gestaltet werden soll, sollte eine spezialisierte Subklasse des DescriptionManagers implementiert werden, die entsprechend „vorsichtig“ agiert und beispielsweise in der Lage ist, Texte die in einigen Endgeräten eventuell zu lang sind und gekürzt werden müssen, mit entsprechenden Hinweisen zu versehen und außerdem auf die Quelle des Originaltextes verweisen kann.

6.5 eXBO Artikel

Die Beobachtung der Diskussionen der Business Object Domain Task Force der OMG machten deutlich, daß gerade für das Business-Object-Artikel die Definition von allgemeingültigen Attributen und Eigenschaften nahezu unmöglich ist. Auch die eigenen Erfahrungen, z.B. bei der Implementierung der Artikel-Klasse für das Portal-System für Buchverlage TXT machten deutlich, daß dies nicht allein durch Entwickler, sondern in enger Zusammenarbeit mit den jeweiligen Domain-Experten durchgeführt werden sollte. Dementsprechend kann – und soll – die nachfolgende Beschreibung des eXBO-Artikel nur ein Vorschlag für eine Basis-Implementierung sein, die später den individuellen Bedürfnissen angepaßt wird.

Zu den Attributen einer Basis-Implementierung des eXBO-Artikel gehört ein kurzer Text für die Bezeichnung des Artikels, ein längerer Text für eine genauere Beschreibung, ein Detail-Foto für die entsprechende Artikel-Detail-Ansicht, ein Thumbnail-Photo für eine eventuelle Voransicht des Artikels in einer Liste, die z.B. alle Artikel einer Warengruppe darstellt, die Mehrwertsteuerklasse des Artikels und mindestens ein Preis. Oft werden neben dem Standardpreis zusätzliche Preise, wie Sonderangebotspreis, unverbindliche Preisempfehlung usw. benötigt. Bei international agierenden eCommerce-Systemen müssen zudem alle

Preisattribute in den verschiedenen Landeswährungen eingeführt werden. Außerdem sollte, ganz besonders für das Business-Object Artikel, eine umfassende Informationsversorgung durch multimediale Zusatzinformationen ermöglicht werden. Bei international agierenden eCommerce-Anwendungen, die Artikel in verschiedensten Währungen anbieten und eventuell zusätzlich unterschiedliche Preise wie die oben angesprochenen unverbindlichen Preisempfehlungen, Sonderangebotspreise oder darüber hinaus sogar Mengenrabattstaffeln einführen, empfiehlt sich die Einführung von entsprechenden Preismanagern, die die Verwaltung und Darstellung der Preise unterstützen.

In der nachfolgenden Abbildung ist das UML-Klassendiagramm für das eXBO Artikel abgebildet:

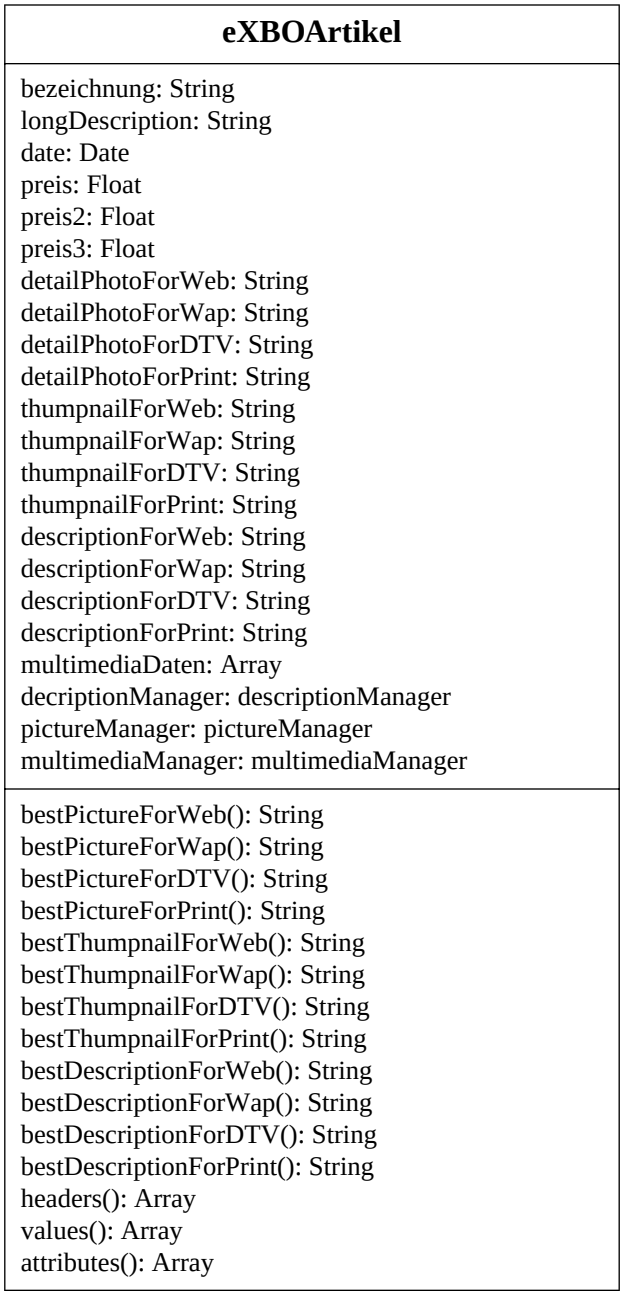


Abbildung 61: UML-Klassendiagramm für das eXBO Artikel

1. bezeichnung bis multimediaDaten: Die normalen Attribute der Artikelklasse und die durch Manager zu behandelnden Elemente, wie „detailPhoto“, „thumbnailPhoto“ und „multimediaDaten“ mit ihren dafür vorgesehenen Attributen.

2. descriptionManager bis multimediaManger: Die Manager, die das eXBO bei der Optimierung für verschiedene Endgeräte unterstützen.

3. bestPictureForWeb bis attributes: Die Zugriffsmethoden für die zu optimierenden Attribute und die Standard-Methoden „headers“, „values“ und „attributes“. Die Zugriffsmethoden für die „normalen“ Attribute wurden zugunsten einer besseren Übersicht nicht abgebildet.

Wie alle eXBO kann auch das eXBO Artikel mit den eXBO-Facilities (vergleiche Kapitel 5) bearbeitet werden, so daß sämtliche Backoffice- bzw. Content-Management-System-Komponenten bereits vorhanden sind. Für den Storefront werden die nachfolgenden dynamischen HTML-Oberflächen vorgeschlagen, die in den Anwendungen TXT und MetaShop umgesetzt wurden:

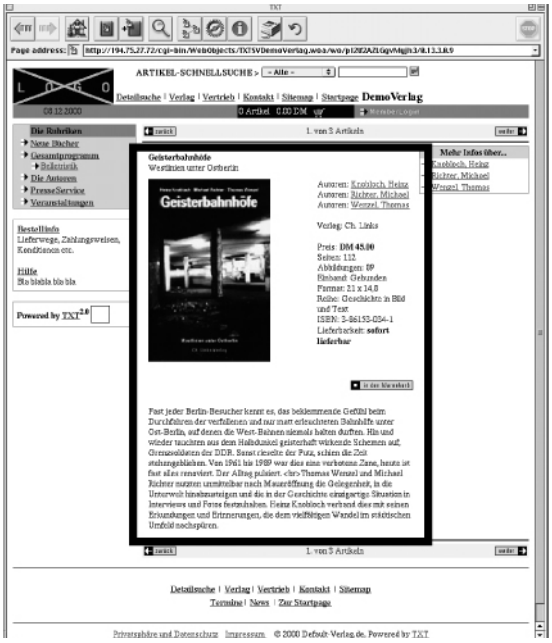


Abbildung 62: Artikel-Detail-Komponente

In der Artikel-Detail-Komponente werden alle vorhandenen Informationen inklusive der Links zu den Multimedia-Daten angeboten.

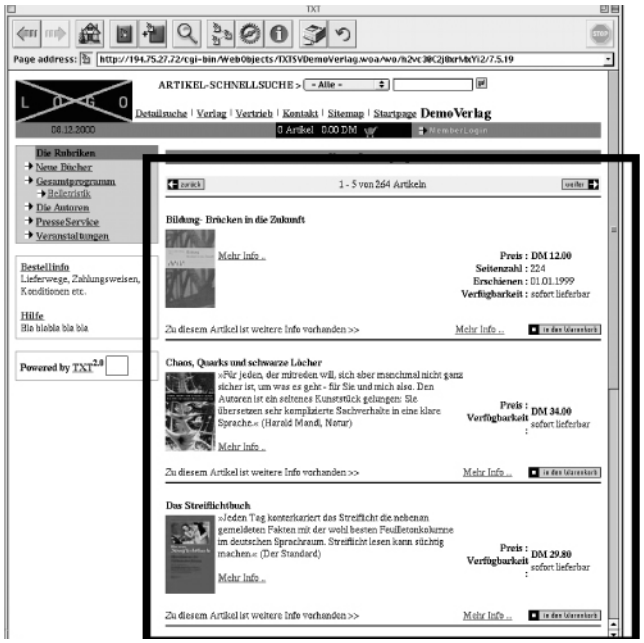


Abbildung 63: Artikel-List-Komponente

Die Artikel-List-Komponente wird z.B. genutzt, um alle Artikel einer Warengruppe übersichtlich anzuzeigen.

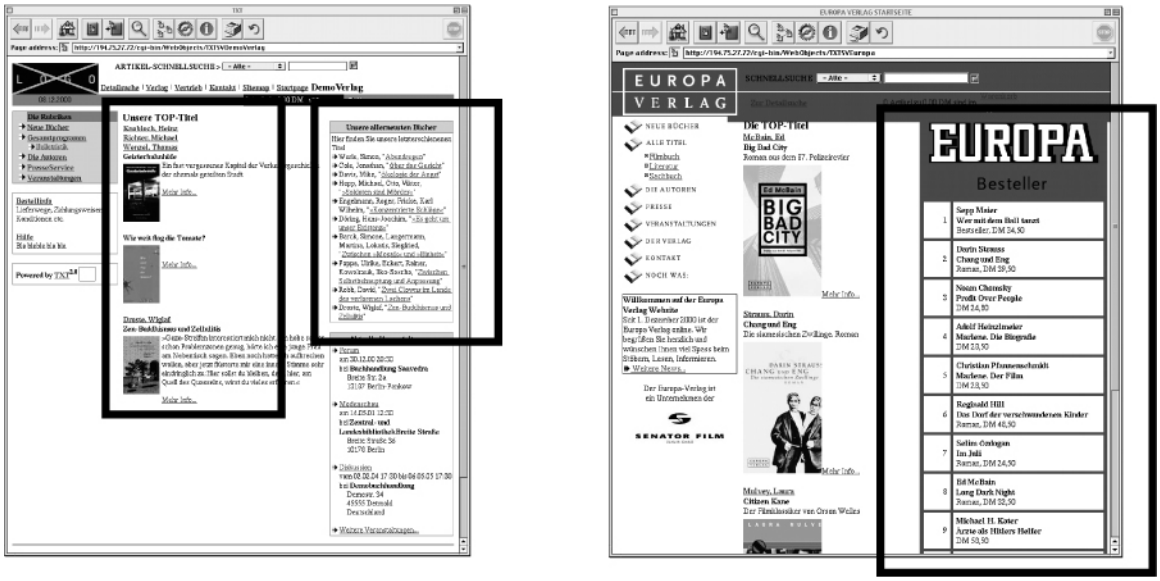


Abbildung 64: Verschiedene Erscheinungsformen der Artikel-List-Komponente

In Abbildung 64 (siehe auch Abbildung 91) sind weitere Varianten der Artikel-List-Page angegeben, wie sie für den Prototypen TXT umgesetzt wurden. Wie alle besprochenen dynamischen HTML-Komponenten sind auch diese durch einfache Tags auch von Graphikdesignern (d. h. Nicht-Programmierern) an jede gewünschte Stelle eines HTML-Source-Codes plazierbar.

6.6 eXBO Multimedia

Das eXBO Multimedia wird an vielen Stellen von den unterschiedlichsten eXBO verwendet um die verschiedensten multimedialen Zusatzinformationen über sich zu verwalten. eXBO, wie z.B. Artikel, Warengruppe oder Termin besitzen zu diesem Zweck ein Multimedia-Array, in dem die zugehörigen Multimedia-eXBO gehalten werden. Der Multimedia-Manager unterstützt die entsprechenden Objekte dabei, optimierte Repräsentationsformen der Multimedia-Informationen für verschiedene Endgeräte bereitzustellen.

Zu den wichtigsten Attributen des eXBO Multimedia gehört ein String-Objekt bzw. die „Pfadangabe“ zur eigentlichen Multimedia-Datei, eine Beschreibung des Inhalts der Datei und eine Bezeichnung bzw. der Name der Datei. Weitere Attribute wie „kind“, das ein Hinweis auf die Art der Multimedia-Informationen enthält (z.B. Video, Sound, Bild, Textdatei), „type“, das den jeweiligen Dateityp angibt (für ein Bild z.B. jpg, .tif oder .gif) und „size“, das die Dateigröße angibt, können von MultimediaManagern für die Vorauswahl von möglichen Multimedia-Daten für ein bestimmtes Endgerät herangezogen werden.

In der nachfolgenden Abbildung 65 ist das UML-Klassendiagramm für das eXBO-

Multimedia dargestellt.

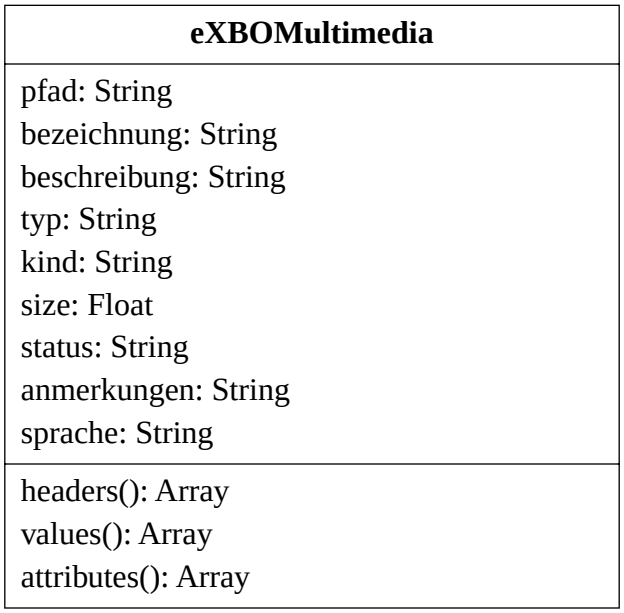


Abbildung 65: UML-Klassendiagramm für das eXBO Multimedia

1. pfad bis sprache: Die hier zusätzlich angegebenen Attribute, wie „info“, „status“ und „sprache“ eignen sich insbesondere für die Verarbeitung durch den Multimedia-Manager, dem mit Hilfe dieser Attribute eine gezielte Auswahl von Multimedia-Daten ermöglicht werden kann.

2. headers, values, attributes: Neben den drei bekannten Methoden „header“, „values“ und „attributes“, die die Ver-arbeitung durch die eXBO-Facilities ermöglichen (vergleiche Kapitel 5), benötigt das Multimedia-Objekt keine weiteren Methoden.

Die Bearbeitung der eXBO vom Typ Multimedia wird ebenfalls durch die eXBO-Facilities unterstützt. Für die Darstellung im Storefront sind keine gesonderten dynamischen HTML-Komponenten vorgesehen, da die Multimedia-Daten mit Hilfe des Attributs „bezeichnung“ als einfache Hyperlinks dargestellt werden können. Das Herunterladen und Anzeigen der Multimedia-Daten, mit den entsprechenden, zur Verfügung stehenden Programmen, übernimmt im Normalfall der Web-Browser. Für Endgeräte die nicht über entsprechende Anwendungen wie einen Web-Browser verfügen, der selbständig multimediale Daten laden und durch ausgewählte Programme zur Anzeige bringen kann, müßten andere Mechanismen in Betracht gezogen werden, deren Ausarbeitung aber nicht Bestandteil dieser Arbeit ist.

Lediglich der passende Ort für solche Erweiterungen, der zentrale, von allen Multimedia-Objekten genutzte MultimediaManager soll an dieser Stelle vorgeschlagen werden.

In der nächsten Abbildung 66 ist der Einsatz des eXBO Multimedia im System TXT gezeigt. In diesem Fall wurden Multimedia-Daten genutzt um Zusatzinformationen über einen Buch-Autor zu geben. Die zur Verfügung stehenden Multimedia-Daten (im Beispiel Fotos und ein Video) werden durch Klick auf die entsprechenden Einträge in der rechten oberen Box in einem neu geöffneten Fenster angezeigt.

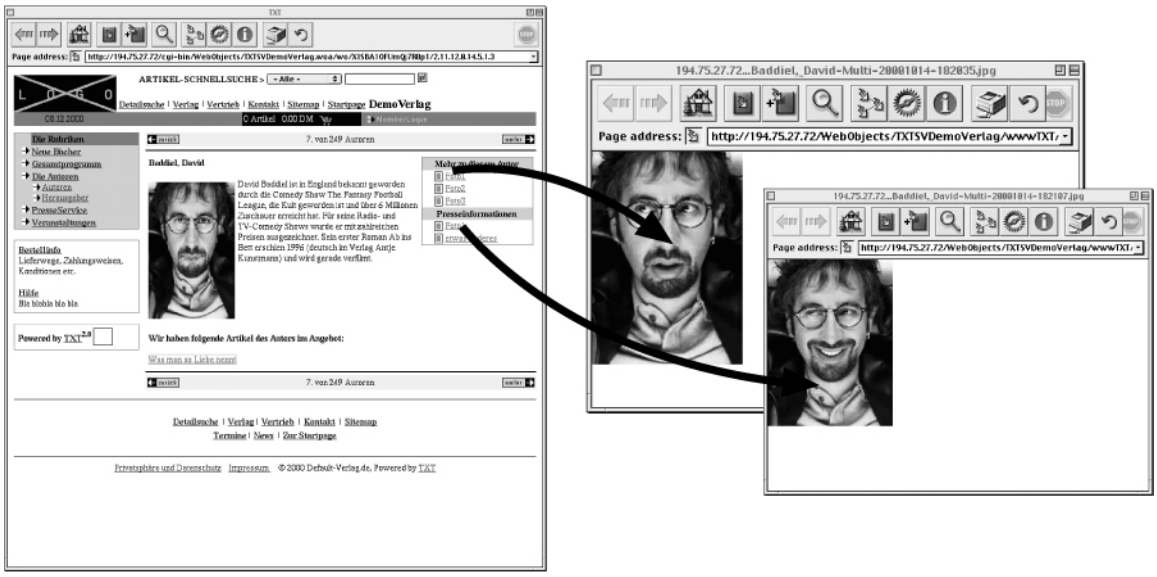


Abbildung 66: Multimedia-Objekte im Einsatz

6.7 eXBO Zahlungsarten

Das eXBO Zahlungsarten kann z.B. in einem eCommerce-Shop-System genutzt werden, um die verschiedenen, angebotenen Zahlungsarten zu verwalten. Zu den gebräuchlichsten Zahlungsarten gehören dabei die Online-Bezahlung per Kreditkarte, die Online-Bezahlung per Banküberweisung bzw. Kontoabbuchung, die spätere Bezahlung mit Hilfe einer beigefügten Rechnung oder auch die weitverbreitete Zahlung „per Nachnahme“, wie sie als Service-Leistung von verschiedenen Zustelldiensten, wie z.B. der Post angeboten wird. Dabei erfolgt die eigentliche Zahlung nach Überreichung der Ware durch den Mitarbeiter des Zustelldienstes. Mit Hilfe der eXBO Zahlungsarten können die in einem eCommerce-System angebotenen Zahlungsvarianten beliebig erweitert und den Erfordernissen und Erfahrungen angepaßt werden. Zu den wichtigsten Attributen des eXBO Zahlungsarten gehören zum einen die Bezeichnung der Zahlungsart und zum anderen ein Text, der den Zahlungsvorgang erläutert und auf eventuelle Risiken und Garantien hinweist. Weitere Attribute sind z.B. "Status", mit dem auf einfache Weise bestimmte Zahlungsvarianten als „nicht zu verwenden“ markiert werden können, um ihre Anzeige im Storefront zu verhindern oder das Attribut „picture“, mit dem eine entsprechende symbolische Graphik angezeigt werden kann (z.B. das Logo eines Kreditkarten-Unternehmens). In der nachfolgenden Abbildung 67 ist das UML-Klassendiagramm für das eXBO Zahlungsarten wiedergegeben:

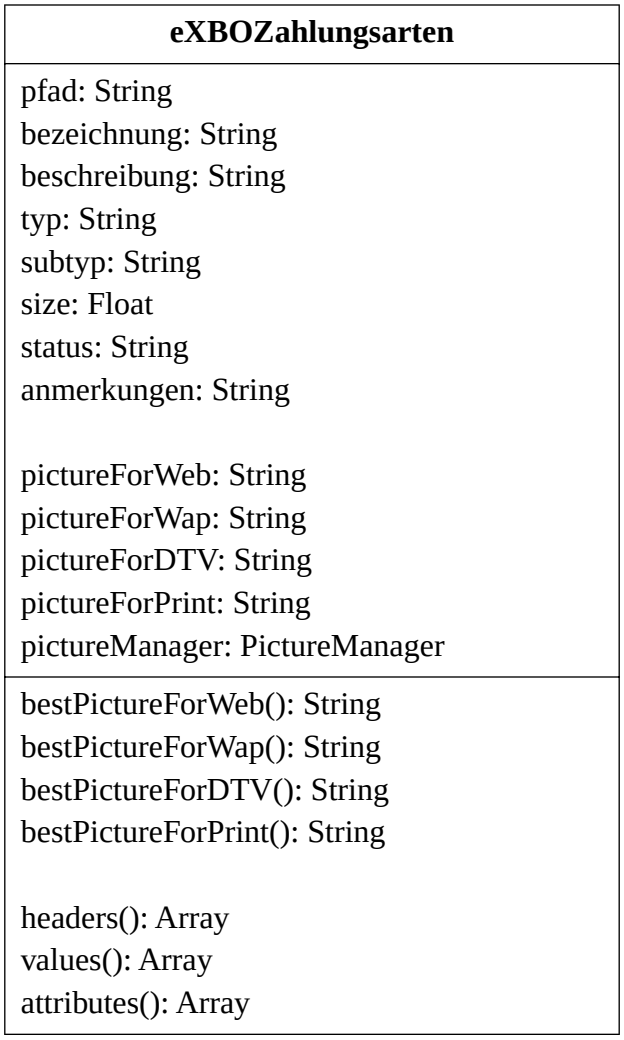


Abbildung 67: UML-Klassendiagramm für das eXBO Zahlungsarten

1. pfad bis pictureManager: Die vorgeschlagenen „normalen“ Attribute, das für verschiedene Endgeräte zu optimierenden Attribut „picture“ und der entsprechende „PictureManager“

2. bestPictureForWeb bis attributes: Die Zugriffsmethoden für die optimierbaren Attribute und die drei Methoden, die eine Verarbeitung mit den eXBO-Facility-Komponenten ermöglichen.

In der nachfolgenden Abbildung sind die dynamischen HTML-Oberflächen dargestellt, die im Storefront-Bereich der Anwendungen TXT bzw. MetaShop umgesetzt wurden (vergleiche Kapitel 8):

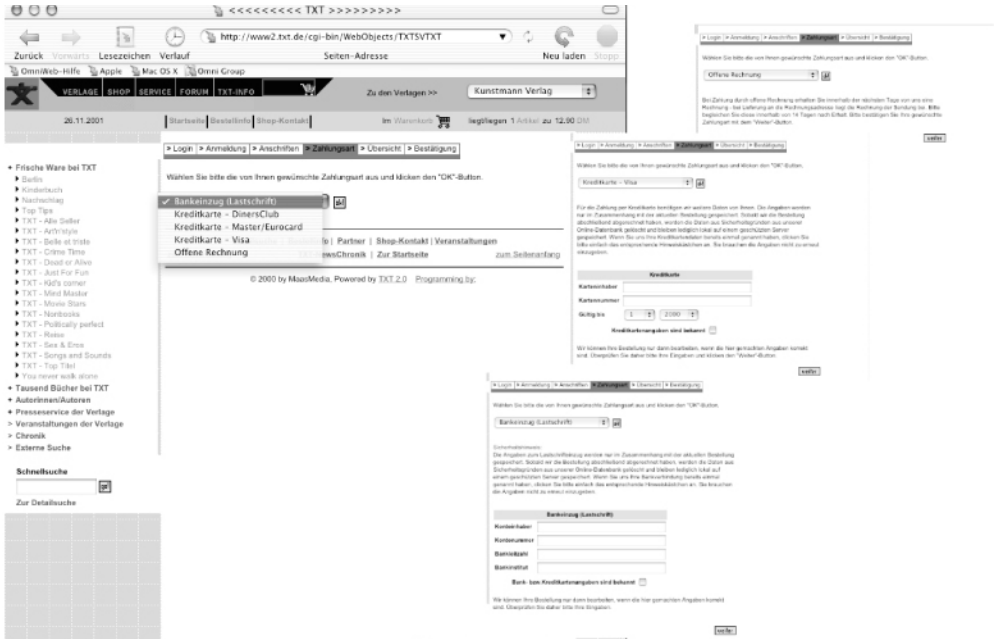


Abbildung 68: Auswahl von Zahlungsarten im Storefront der Anwendung TXT

Die Auswahl der verschiedenen Zahlungsvarianten wird in diesem Beispiel durch einen Choice gelöst. Je nach ausgewählter Zahlungsvariante werden entsprechende Eingabefelder in der Dialog-Box unterhalb der Choice eingeblendet. Die eigentlichen Zahlungsinformationen (wie z.B. Kreditkartennummer, Bankleitzahl usw.) werden in dem eXBO Bestellung verwaltet.

6.8 eXBO Liefervariante

Das eXBO Liefervariante kann genutzt werden, um dem Kunden die Auswahl einer Liefervariante oder eines Zustellunternehmens zu ermöglichen. So kann er selbst entscheiden, ob er einen kostengünstigen oder kostenlosen – aber langsamen - Transport, oder die „Expresslieferung“ mit garantierter Zustellzeit und entsprechendem Aufpreis wählt.

Zu den wichtigsten Attributen des eXBO Liefervariante gehören, wie schon beim eXBO Zahlungsvarianten eine Beschreibung und ein erklärender Text. Die Attribute „status“, mit dem Liefervarianten ein- und ausgeblendet werden können und „picture“, mit dem eine symbolische Abbildung oder das Logo eines Zustell-Unternehmens, wie z.B. UPS, Post oder FeedEx dargestellt werden können, werden ebenfalls vorgeschlagen. Mit dem Attribut „zusätzlicheKosten“ werden eventuelle Aufpreise für bestimmte Liefervarianten verwaltet. In der nachstehenden Abbildung 69 ist das UML-Klassendiagramm für das eXBO Liefervariante angegeben.

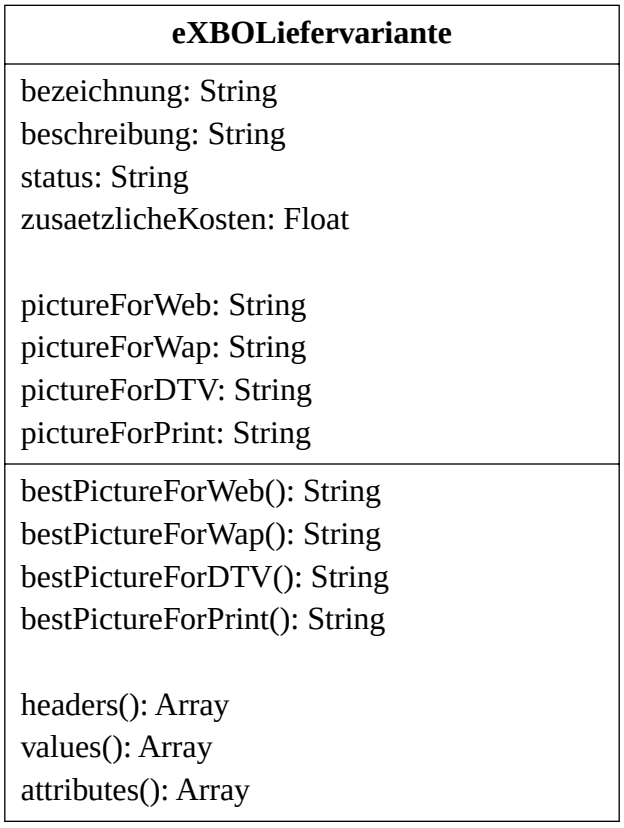


Abbildung 69: UML-Klassendiagramm für das eXBO Liefervariante

Das UML-Klassendiagramm des eXBO Liefervariante ist dem des zuvor vorgestellten eXBO Zahlungsvariante sehr ähnlich, so daß an dieser Stelle auf eine nähere Erläuterung der

Attribute und Methoden verzichtet werden kann und auf den vorangegangenen Abschnitt verwiesen wird. Wie alle eXBO kann auch das eXBO Liefervarianten mit den eXBO-Facilities (siehe Kapitel 5) bearbeitet werden, so daß für den Backoffice-Bereich bzw. für das Content-Management-System alle erforderlichen Komponenten bereits vorhanden sind.

6.9 eXBO Versandkosten

Das eXBO Versandkosten kann genutzt werden, um gestaffelte Versandkostenpreise zu realisieren. Mit den Attributen „preisVon“ und „preisBis“ lassen sich entsprechende Staffelungen aufbauen, die sich nach der Gesamtsumme der Bestellung richten (z.B. „normaler“ Versandkostenpreis bei kleineren Bestellungen, „ermäßigter“ Versandkostenpreis bei mittelgroßen Bestellsummen und „kostenlose“ Lieferung bei entsprechend großer Gesamtsumme). Durch die Attribute „gewichtVon“ und „gewichtBis“ können entsprechende Staffelungen nach Gewicht aufgebaut werden.

Andere Attribute (wie z.B. „größe“, „gefahrenklasse“ usw.) können nach Bedarf hinzugefügt werden. Das Attribut „land“ macht landesabhängige Versandkostenberechnungen möglich. Im Attribut „versandkosten“ ist die zu zahlende Summe vermerkt. In der folgenden Abbildung ist das UML-Klassendiagramm für das eXBO Versandkosten abgebildet:

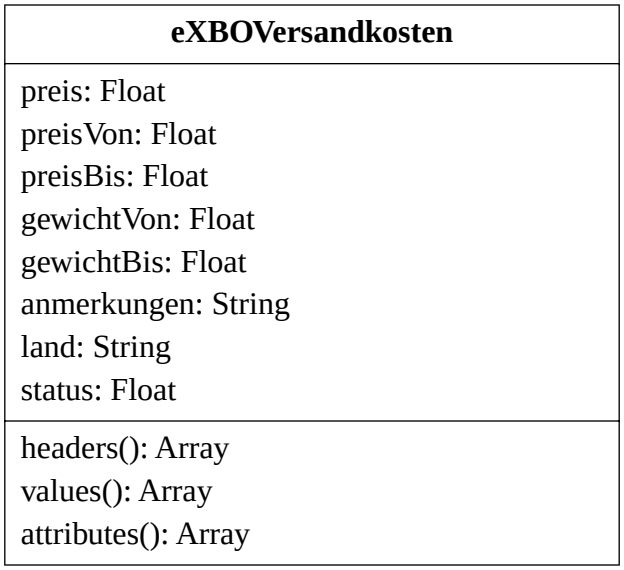


Abbildung 70: UML-Klassendiagramm eXBO Versandkosten

Bei der Verwendung des eXBO Versandkosten in eCommerce-Anwendungen, die verschiedene Landeswährungen unterstützen, sollten die entsprechenden zusätzlichen Preisattribute eingeführt werden und ein PreisManager (vgl. Abschnitt 6.1) hinzugezogen werden. Die eXBO Versandkosten können wie alle eXBO mit den eXBO-Facility-Komponenten bearbeitet werden. Die Versandkosten benötigen im Normalfall keine Darstellungskomponenten (z.B. für den Storefront eines eCommerce-Shop-Systems), sie werden lediglich als Berechnungsgrundlage für den Lieferpreis herangezogen.

6.10 eXBO Kunde

Das eXBO Kunde ist weitgehend selbsterklärend. Es beschreibt diejenige Person, Firma oder Behörde die als Kunde mit einem eCommerce-System in Kontakt tritt.

Zu den wichtigsten Attributen des eXBO Kunde zählen die Attribute, die zur Anschrift gehören, wie z.B. Name, Vorname, Straße, Telefon usw. Für eCommerce-Anwendungen sollten dabei mindestens zwei verschiedene Anschriften vorgesehen werden (eine Lieferanschrift und eine Rechnungsanschrift). In vielen Fällen, wenn die bestellende Person weder die Rechnung noch die Ware erhält, ist es sinnvoll, noch eine dritte Anschrift (die eigentliche Kontaktadresse des Bestellers) einzuführen. Neben der Anschrift sind Zahlungsinformationen, wie z.B. Bank oder Kreditkarten-Angaben für einen reibungslosen Zahlungsablauf erforderlich. Die Attribute Login und Passwort können genutzt werden um die Wiedererkennung eines Kunden zu ermöglichen, und ein wie auch immer geartetes „profiling“ durchzuführen. Beispielsweise kann einem wiedererkannten Kunden gestattet werden auf nicht-öffentliche Informationen zuzugreifen. Komplexere Profiling-Varianten bieten z.B. eine individualisierte Gestaltung der Benutzeroberfläche und eine persönliche Ansprache. Einfachere Varianten nutzen die Kundendaten, um das Ausfüllen der Liefer- und Rechnungsangaben zu erleichtern. Weitere Attribute, wie z.B. Geburtsdatum (Ermöglichung von entsprechenden Grußkarten), Status (z.B. um Stammkunden die Bestellung „per Rechnung“ zu ermöglichen) und Anmerkungen (für Informationen über den Kunden) werden ebenfalls vorgeschlagen.

In der nachfolgenden Abbildung ist das UML-Klassendiagramm für das eXBO Kunde angegeben:

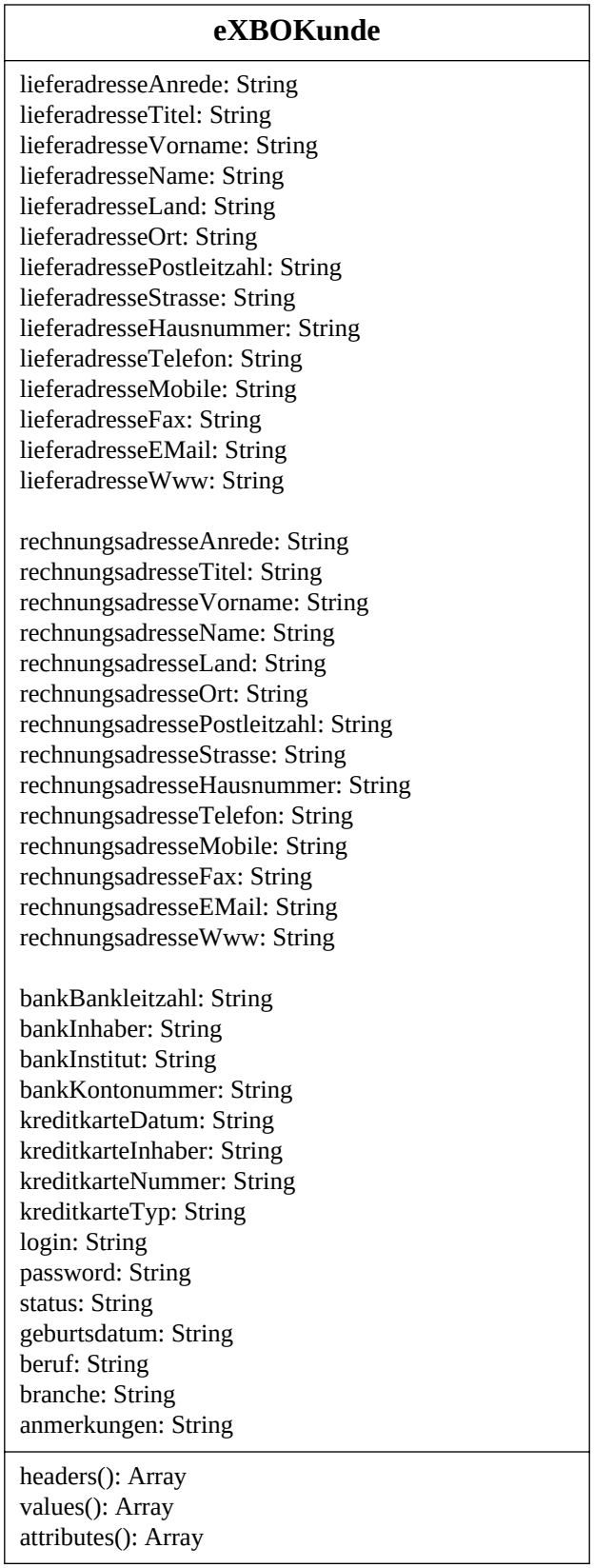


Abbildung 71: UML-Klassendiagramm für das eXBO Kunde

Wie alle eXBO kann auch das eXBO Kunde mit den eXBO-Facility-Komponenten bearbeitet werden, so daß alle erforderlichen Komponenten für den Backoffice bzw. für ein Content-Management-System bereits vorhanden sind. Für die Eingabe der Kundendaten, die in den meisten Fällen von dem Kunden selbst durchgeführt wird, können zwei verschiedene Varianten gewählt werden. Entweder eine Oberfläche, mit der ein Kunde gezielt und bewußt seine Daten einträgt oder eine davon unabhängige Variante, die sowohl anstelle als auch zusätzlich eingebunden werden kann, bei der die Kundendaten während der ersten Bestellung aufgenommen werden. Dabei kann ein „normales“ Bestelldateneingabeformular um zusätzliche Felder, wie z.B. Login und Passwort, erweitert werden.

Abbildung 72: Eingabe der Kundendaten während einer Bestellung

6.11 eXBO Bestellung

Das eXBO Bestellung wird eingesetzt, um sämtliche Bestelldaten zu erfassen. Dazu gehören Lieferadresse, Rechnungsadresse, Versandkosten, Zahlungsinformationen und natürlich die bestellten Produkte. Das eXBO Bestellung verändert in den verschiedenen Phasen der Geschäftsprozessunterstützung (Informationsphase, Vereinbarungsphase, Abwicklungsphase und After-Sales-Phase seinen Status und die zugehörige Repräsentationsform. Vor der eigentlichen Bestellung (in dem Zeitraum indem der Kunde seinen Warenkorb füllt) wird es genutzt, um den Warenkorb darzustellen und zu verwalten. Nach dem Betätigen des Knopfes „jetzt bestellen“ und der anschließenden Bestellbestätigung wird der Warenkorb zu einer rechtskräftigen Bestellung. Innerhalb des Objektes Bestellung, wird darauf durch die Änderung des Attributes „status“ reagiert. Mit dem Status einer Bestellung kann sich die Repräsentationsform des eXBO Bestellung weiter verändern, z.B. in „Lieferschein“, „Bestellbestätigung“, „Rechnung“, „Mahnung“, „offene Rechnung“, „bezahlte Rechnung“ usw. Die verschiedenen Stati können dabei durch das Attribut „status“ hinreichend differenziert werden. In der Abbildung 90 (Kapitel 8) ist ein Beispiel für eine dynamische HTML-Oberfläche für die Repräsentationsform als Warenkorb angegeben. In der folgenden Abbildung 73 ist zunächst das UML-Klassendiagramm für das eXBO Bestellung dargestellt.

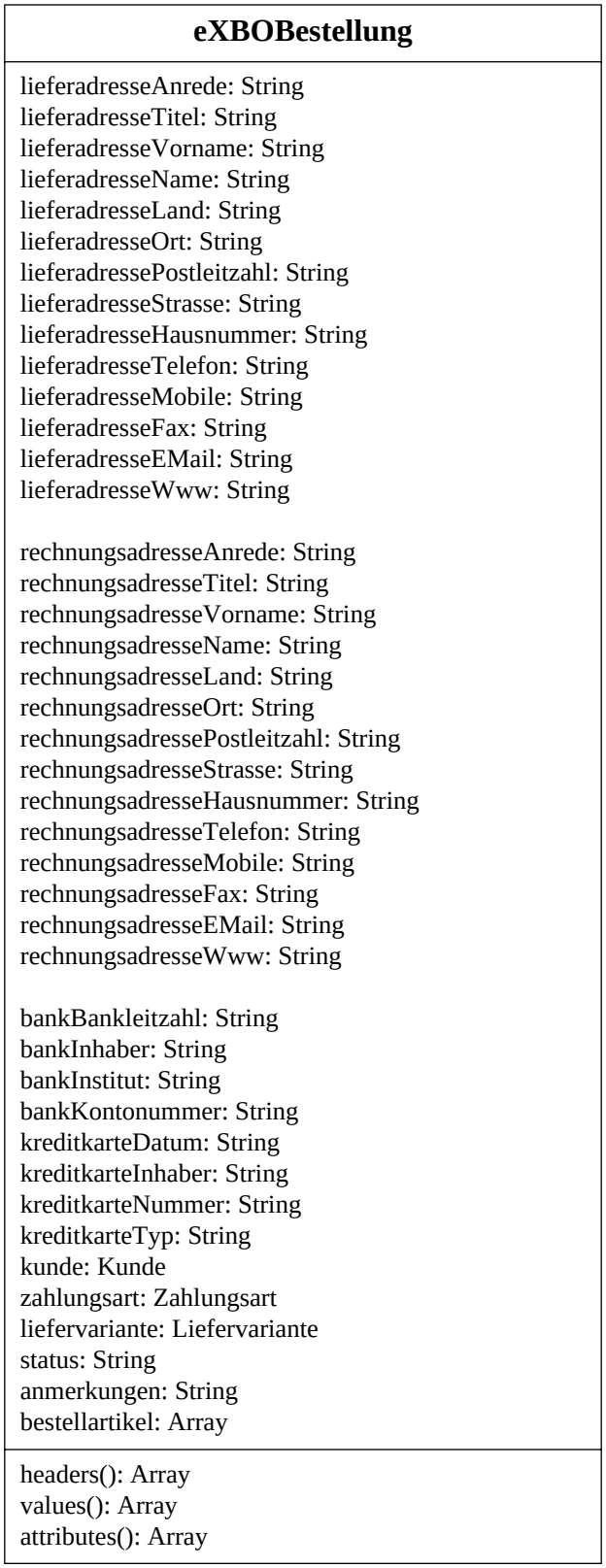


Abbildung 73: UML-Klassendiagramm eXBO Bestellung

Wie dem UML-Klassendiagramm zu entnehmen ist, wird im Falle des eXBO Bestellung die Verwendung von Attributen, die für verschiedene Endgeräte optimiert werden müßten, vermieden. Bei der Betrachtung des UML-Klassendiagramms fallen zwei Dinge auf:

1. Die relativ große Anzahl von Attributen, im Vergleich zu den sonstigen bisher vorgestellten eXBO.

2. Die Tatsache, daß fast alle Attribute bereits aus dem eXBO Kunde bekannt sind, hier also zum zweiten Mal präsentiert werden.

Aus Sicht eines „Datenbank-Puristen“, der stets darauf bedacht ist, doppelte Datenhaltung zu vermeiden, um seine Datenbank redundanzfrei bzw. widerspruchsfrei zu halten und vor allem um Platz zu sparen, muß dies zunächst sehr befremdlich erscheinen. Aus der Sicht von Business-Object-Anwendern, die versuchen die Dinge aus der realen Geschäftswelt möglichst direkt und wirklichkeitsnah nachzubilden, ist dieser Ansatz selbstverständlich. Auch in der „realen“ Welt befindet sich z.B. die Anschrift eines Kunden sowohl auf seinen Personalausweis, als auch auf jeder für ihn bestimmten Rechnung. Auch im Hinblick auf die Rechtskräftigkeit von elektronischen Dokumenten sollte ein Objekt „Bestellung“ exakt alle Informationen beinhalten, die auch das ausgedruckte „Original-Dokument“ enthält. Also beispielsweise nicht nur eine Referenz zum Kunden, über die auf seine Adressdaten zugegriffen werden kann. Nach einem Umzug könnte dies z.B. zur automatischen Anpassung (bzw. „Verfälschung“) des eXBO Rechnung führen. Im nächsten Abschnitt wird das eXBO Bestellartikel beschrieben, das wesentlicher Bestandteil einer Bestellung ist.

6.12 eXBO Bestellartikel

Das eXBO Bestellartikel wird eingesetzt, um die Artikel, die zu einer Bestellung gehören, zu verwalten. Jedes Mal wenn ein Kunde in einem eCommerce-Shop-System einen Artikel in seinen Warenkorb legt, wird ein neues Bestellartikel-Object erzeugt und der Bestellung hinzugefügt. Dabei werden die für die Rechtskräftigkeit der Bestellung relevanten Daten des ausgewählten Artikels in das Bestellartikel-Object kopiert. Wie im vorangegangenen Beispiel des eXBO Bestellung erläutert, dient dies dazu, den Zustand des Artikels zum Zeitpunkt der Bestellung „einzufrieren“ und eventuelle Probleme mit sich ändernden Artikelbezeichnungen oder Artikelpreisen zu vermeiden.

Zu den wichtigsten Attributen gehören neben der Bezeichnung bzw. dem Namen des Bestellartikels, sein Preis und die anzuwendende Mehrwertsteuer. In dem nachfolgenden UML-Klassendiagramm sind weitere Attribute, wie z.B. die Anzahl von Artikeln und Referenzen zum Artikel und zur Bestellung vorgeschlagen.

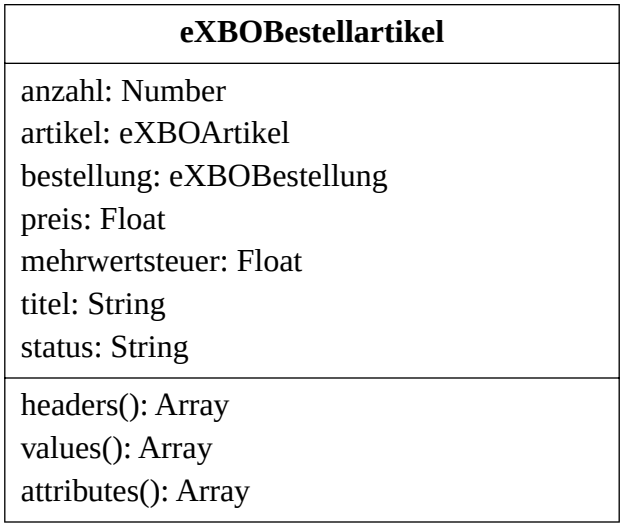


Abbildung 74: UML-Klassendiagramm für das eXBO Bestellartikel

6.13 Zusammenfassung eXBO

In der Einleitung zu diesem Kapitel wurde zunächst auf die Problematik bei der Darstellung von Inhalten in unterschiedlichen Endgeräten eingegangen und es wurden die beiden grundsätzlichen Lösungsansätze angesprochen. Im Abschnitt 6.1 wurde ein Konzept vorgestellt, das es ermöglicht, beide Ansätze miteinander zu vereinen und das für verschiedene Einsatzgebiete und Anwendungsfälle individuell anpassbar und ausbaufähig ist.

Für die beschriebene Endgeräteunabhängigkeit (bzw. für die optimierte Unterstützung verschiedener Endgeräte) müssen „normale“ Business-Objects dabei lediglich in drei Punkten erweitert werden:

- Für diejenigen Attribute (wie z.B. „picture“, „longDescription“ usw.), die bei der Darstellung in verschiedenen Endgeräten zu Problemen führen können, werden spezialisierte Attribute für jedes zu unterstützende Endgerät eingeführt.
- Zusätzlich zu den Standart set- und get-Zugriffsmethoden wird für jedes spezialisierte Attribut eine zugehörige „best...For...“-Methode eingeführt.
- Manager (wie z.B. „Description-Manager“ oder „Picture-Manager“) übernehmen die Ermittlung des Rückgabewertes innerhalb der „best...For...“-Methode.

Kapitel 7

Weitere Komponenten

Ein komplettes Framework für eCommerce-Anwendungen benötigt selbstverständlich noch weitere Komponenten und Objekte, die den Entwickler bei der Fertigstellung von komplexen Applikationen unterstützen. Während der umfangreichen Implementierungsarbeiten und Tests der verschiedensten Prototypen (vergleiche Kapitel 8) konnten zahlreiche wiederverwendbare Komponenten identifiziert werden, die geeignet sind, das vorgestellte Framework zu ergänzen. In den nachfolgenden Abschnitten wird ein Teil dieser wiederverwendbaren Objekte und Komponenten vorgestellt. Um den Umfang der Arbeit nicht endgültig zu sprengen, werden diese weiteren Komponenten oder auch „supporting Facilities“ in „Kurzportraits“, d.h. weniger ausführlich als die eXBO-Facilities und eXBO in den vorangegangenen Kapiteln, vorgestellt.

Wenngleich nur ein Teil der implementierten Komponenten vorgestellt werden kann, soll an dieser Stelle deutlich ausgesprochen werden, daß kein Framework – also auch nicht das hier vorgestellte – jemals einen Anspruch auf Vollständigkeit erheben sollte. Ein Framework sollte vielmehr ständig für Erweiterungen offen gehalten werden und nach Möglichkeit durch jede neue Implementierung und jede neue Erkenntnis ergänzt und iterativ verbessert werden.

7.1 XLocalizableString

Bei der Konstruktion von Anwendungen, die in verschiedenen Regionen der Erde genutzt oder verkauft werden sollen, steht man häufig vor dem Problem, verschiedene Sprachen unterstützen zu müssen. Für die Lösung dieses Problems stehen grundsätzlich zwei verschiedene Ansätze zur Verfügung. Entweder wird für jede Landessprache eine eigene angepaßte Applikation, bzw. eine angepaßte Oberfläche entwickelt, was bei größeren Anwendungen zu entsprechenden Mehraufwänden vor allem bei Wartungs- und Ergänzungsarbeiten führt, oder es wird versucht „dynamische“ Benutzeroberflächen einzuführen, die sich je nach ausgewählter Landessprache selbständig an ihren Benutzer anpassen. Das hier vorgestellte XLocalizableString-Object wurde entworfen, um den letzteren der beiden Ansätze zu unterstützen. In der nachfolgenden Abbildung ist eine Login-Seite einer Beispielanwendung wiedergegeben, bei der das XLocalizableString-Object zum Einsatz kommt.

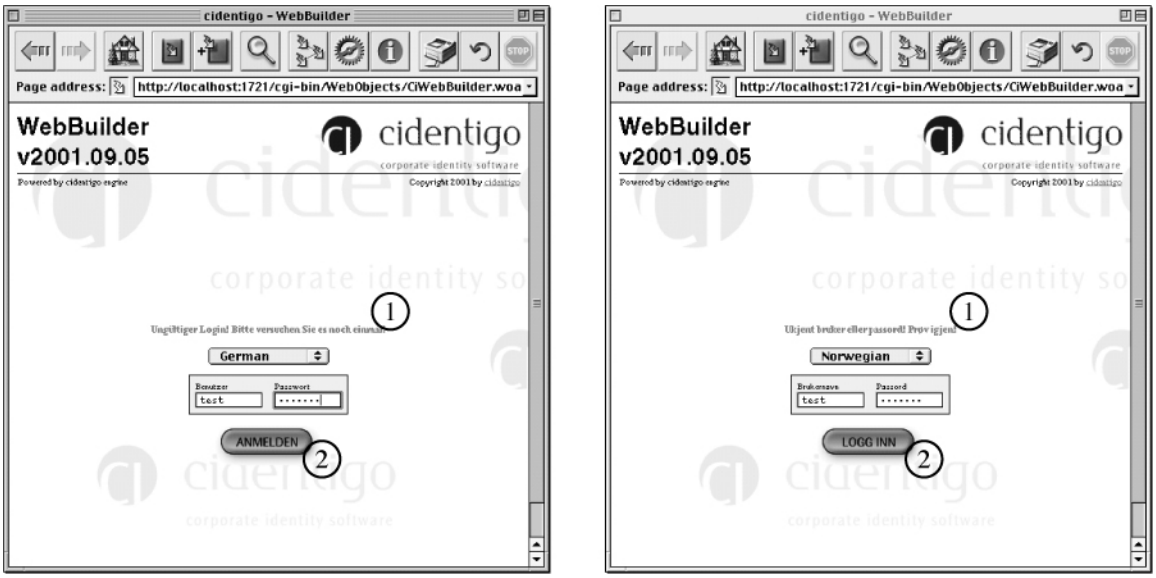


Abbildung 75: Mehrsprachige Login-Seite

Bei dem Entwurf des XLocalizableString-Objectes stand im Vordergrund, einfachste Verwendbarkeit mit geringstem Mehraufwand zu ermöglichen. Anstatt einen String, wie im

Beispiel der Abb.ildung 75: „Ungültiger Login! Bitte versuchen Sie es noch einmal“ direkt in das HTML zu schreiben, wird er als „Default-Value“-Attribut des XLocalizableString-Objects angegeben. In der HTML-Seitenbeschreibung wird an der Stelle des ursprünglichen „festen“ Strings das dynamische XLocalizableString-Objekt eingefügt (vergleiche hierzu die Einführung in dynamische HTML-Seiten aus dem 2. Kapitel). Per Voreinstellungen, bzw. wenn keine Übersetzungen für den String vorhanden sind, gibt das XLocalizableString-Object den Default-Value zurück. Diese Eigenschaft hat sich für eine rasche Implementierung von Prototypen, die zunächst keine Rücksicht auf verschiedene Sprachen nehmen, als vorteilhaft erwiesen. Bevor ein XLocalizableString-Object seinen Default-Value zurückgibt, kontrolliert es selbständig eine entsprechende ihm zugeordnete Textdatei und versucht eine passende Übersetzung in der ausgewählten Landessprache zu ermitteln und diese zurückzugeben. Auf diese Weise sind neue Anwendungen bereits auf Mehrsprachigkeit vorbereitet, ohne in der Prototyp-Phase einen nennenswerten Mehraufwand erfordert zu haben. Bei der Umsetzung der Übersetzungs-Textdatei, erwies es sich als vorteilhaft, für jede Landessprache eine separate menschenles- und bearbeitbare ASCII- oder XML-Datei zu halten. Auf diese Weise können die Übersetzungsdateien in entsprechende Länder verschickt und dort selbständig von Sprachkundigen bearbeitet werden.

In der nachfolgenden Abbildung ist das UML-Klassendiagramm für das XLocalizableString-Objekt wiedergegeben:

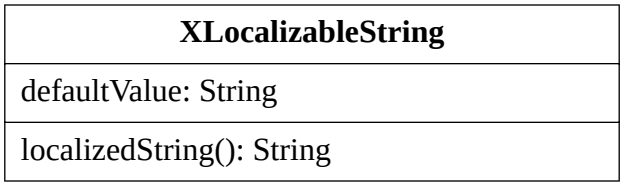


Abbildung 76: UML-Klassendiagramm für das XLocalizableString-Object

1. defaultValue - In der Variable „defaultValue“ wird, wie beschrieben, der Default-String abgelegt.

2. localizedString - Der dynamische Rückgabewert des XLocalizableString wird intern durch den Rückgabewert dieser Methode bestimmt. Dazu wird zunächst die gewählte bzw. eingestellte Landessprache der Anwendung ermittelt, dann nach der entsprechenden Textdatei mit der Übersetzung gesucht und – wenn vorhanden – der entsprechende landesspezifische String zurückgegeben. Wird keine Übersetzung gefunden, wird der „defaultValue“-String zurückgegeben.

7.2 XLocalizableImage

Viele Elemente einer Benutzeroberfläche, wie z.B. Knöpfe bzw. Buttons werden in HTML-basierten Oberflächen gerne mit Hilfe von Bildern umgesetzt. Knöpfe (wie z.B. der Login-Knopf in Abbildung 75), die eine Aufschrift aufweisen, müssen dabei bei mehrsprachigen Anwendungen ebenfalls dynamisch ausgetauscht werden. Diese Funktionalität unterstützt das vorgestellte XLocalizableImage-Object. Genau wie bei dem zuvor vorgestellten XLocalizableString-Object wird dabei ein Bild, bzw. der Pfad zum Bild, nicht direkt in den HTML-Code geschrieben, sondern im Attribut „defaultImage“ im XLocalizableImage-Object. abgelegt. Anstelle des „festen“ Bildes, bzw. der festen Pfadangabe, wird das XLocalizableImage-Object in den HTML-Code eingefügt, das wie zuvor das XLocalizableString-Object eine landesspezifische Textdatei kontrolliert und versucht von dort den Pfad zu einem sprachspezifischen Bild zu ermitteln. Aufgrund der starken Ähnlichkeit zum XLocalizableString-Object wird auf die Angabe des UML-Klassendiagramms für das XLocalizableImage-Object an dieser Stelle verzichtet. Ein Beispiel für den Einsatz des vorgestellten XLocalizableImage-Object ist der vorangegangenen Abbildung 75 zu entnehmen ((2) der Login-Knopf).

7.3 XInlayLogin

Bei der Konstruktion von eCommerce-Anwendungen, mit denen häufig empfindliche Finanz- und Geschäftsdaten verwaltet werden, ist die Schaffung von personen- bzw. rollenbedingten Zugangsbeschränkungen für Teilbereiche einer Anwendung überdurchschnittlich häufig

notwendig. Mit der hier vorgestellten, wiederverwendbaren XInlayLogin-Komponente wird dem Entwickler eine fertige Login-Komponente inklusive Oberfläche und Nutzerdialog zur Verfügung gestellt. Vorgeschlagen wird natürlich, die XInlayLogin-Komponente unter Verwendung der zuvor vorgestellten XLocalizableString- und XLocalizableImage-Objects umzusetzen. Dadurch wird die XInlayLogin-Komponente mehrsprachig und kann sich außerdem automatisch jeder neu hinzukommenden Sprache anpassen. Das wichtigste Attribut innerhalb der XInlayLogin-Komponente ist das „userArray“. In diesem werden die zugangsberechtigten Personen bzw. die entsprechenden Objekte abgelegt (vgl. z.B. eXBO-Kunde). In der nachstehenden Abbildung ist das UML-Klassendiagramm für die XInlayLogin-Komponente wiedergegeben.

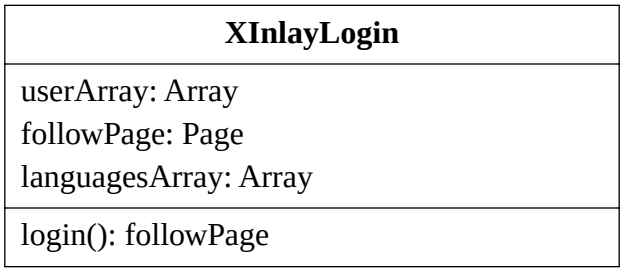


Abbildung 77: UML-Klassendiagramm für das XInlayLogin-Object

1. userArray: Wie oben beschrieben, werden in diesem Array alle User-Objekte, die mindestens die Attribute "Login" und "Passwort" aufweisen müssen, gehalten, für die eine Zugangsberechtigung erteilt werden soll.

2. followPage: Mit diesem Attribut wird die sogenannte „followPage“ verwaltet. Dies ist diejenige Seite, zu der nach einem erfolgreichen Login gewechselt werden soll.

3. languagesArray: Mit diesem Array werden die anzubietenden Sprachen bestimmt.

Eine prototypische Umsetzung der XInlayLogin-Komponente ist der vorangegangenen Abbildung 75 zu entnehmen.

7.4 XPageMenu

Wie bereits in Kapitel 4 beschrieben, stellt die Menu-Komponente, von der aus die verschiedenen „Manager“ für die Bearbeitung von Business-Objects aufgerufen werden können, eine zentrale Komponente in jedem Backoffice bzw. jedem Content-Management-System einer eCommerce-Anwendung dar. Je nach Anwendung und eingesetzten Business-Objects, muß diese Komponente inhaltlich angepaßt werden. Trotzdem kann eine Basis-Implementierung, wie die hier vorgestellte XPageMenu-Komponente zu einer beschleunigten Entwicklung beitragen. In der nachfolgenden Abbildung ist eine prototypische Umsetzung dargestellt (vgl. dazu Kapitel 8).

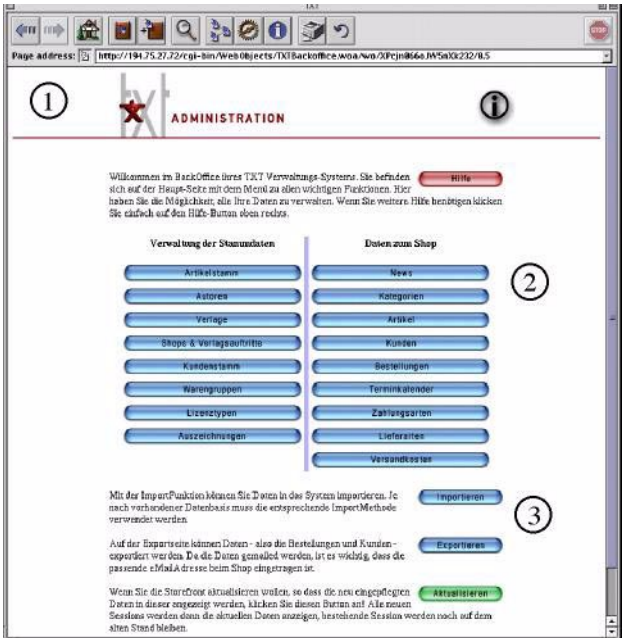


Abbildung 78: Menu-Page

Wie der Abbildung 78 zu entnehmen ist, wird die XPageMenu-Komponente zu großen Teilen aus anderen wiederverwendbaren „Fertig“-Komponenten zusammengesetzt z.B. (XHeader (1) oder XFooter (nicht abgebildet)). Diese Komponenten werden auf den nachfolgenden Seiten angesprochen. Der mit (2) gekennzeichnete Bereich der XPageMenu-Komponente ist der Startpunkt für die Bearbeitung von Business-Objects. Die hier abgebildeten Knöpfe führen bei Betätigung entsprechende Methoden aus, die für den Aufruf der eXBO-Facility List

zuständig sind, und so das „Manager-Muster“ in Gang setzen. Wie in dem entsprechenden Abschnitt des 5. Kapitels beschrieben, müssen dazu entsprechende Parameter an die List-Komponente übergeben werden. Die Ermittlung dieser Parameter, allen voran die Ermittlung des Arrays mit den zu bearbeitenden Business-Objects, muß in der jeweiligen Methode anwendungsspezifisch implementiert werden. Der mit (3) gekennzeichnete Bereich der XPageMenu-Komponente ermöglicht den Zugang zu allen Sonderfunktionen. Im Beispiel des abgebildeten Prototypen können Artikel z.B. aus fremden Systemen importiert und exportiert werden (im Falle des vorgestellten Prototypen TXT z.B. aus der Buchverwaltungssoftware „BookHit“).

7.5 XHeader

Die Verwendung einer separaten Komponente, die einfach per „Tag“ in das HTML geschrieben wird, steigert nicht nur die Übersichtlichkeit, sondern ermöglicht auch die einfache Wiederverwendung dieser auf fast allen Seiten eines Systems vorkommenden Funktionalität einer Kopfzeile. Die in der vorhergehenden Abbildung 78 abgebildete Implementierung einer XHeader-Komponente (vergleiche (1)) weist die Attribute „imageLeft“ und „imageRight“ auf. Das dritte Attribut „text“ ist in Abbildung 78 leer. Aufgrund des einfachen Aufbaus der XHeader-Komponente wird auf die Darstellung eines UML-Klassendiagramms an dieser Stelle verzichtet. Selbstverständlich sollte die XHeader-Komponente ebenfalls unter Einsatz der zuvor vorgestellten XLocalizableString- und XLocalizableImage-Objects implementiert werden, was eine sprachunabhängige Umsetzung ermöglichen würde.

7.6 XFooter

Die Komponente XFooter besitzt in den meisten Anwendungen ähnliche, oft sogar identische, Aufgaben und Erscheinungsformen wie die zuvor vorgestellte XHeader-Komponente, so daß auf eine nähere Erläuterung an dieser Stelle verzichtet werden kann.

7.7 XInlayNavigation

Die Einführung von verschiedenen wiederverwendbaren XInlayNavigation-Komponenten hat sich im Laufe der Implementierungsarbeiten (vergleiche Kapitel 8) ebenfalls als produktivitätssteigernd herausgestellt. Durch die Verwendung der eXBO-Facility-Komponenten (vergleiche Kapitel 5), die ihre eigenen „XInlayNavigation“ besitzen, verringert sich der Bedarf für wiederverwendbare XInlayNavigation-Komponenten zwar erheblich, trotzdem sind in der Praxis noch immer genügend zusätzliche Komponenten zu implementieren, die eine Navigation erfordern. Wie der Abbildung 39 zu entnehmen ist, sollte eine solche XInlayNavigation-Komponente mindestens die folgenden Attribute besitzen: einen Überschriftstext, einen erklärenden Text und die Knöpfe bzw. Hyperlinks für die Navigation, die bei Betätigung die entsprechenden Methoden aufrufen.

7.8 XWarningDialog

Bei allen Funktionen, die wie beispielsweise das Löschen eines Business-Objects, zu permanenten Zustandsänderungen des Systems führen können, sollte der Benutzer durch einen XWarningDialog – einer Sicherheitsabfrage – vor unbeabsichtigten und mitunter folgenschweren Handlungen geschützt werden. In der nachfolgenden Abbildung ist eine solche Sicherheitsabfrage, die mit einem XWarningDialog umgesetzt wurde, abgebildet:

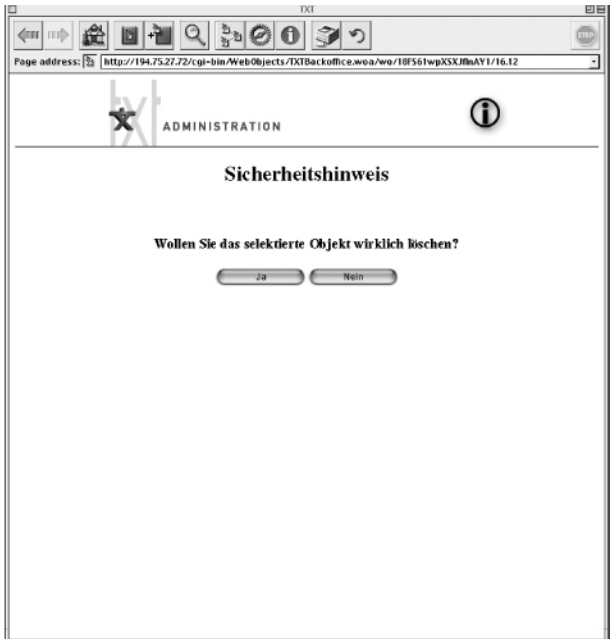


Abbildung 79: XWarningDialog-Komponente

Natürlich sollte insbesondere die XWarningDialog-Komponente mit Hilfe der sprachunabhängigen XLocalizableString- und XLocalizableImage-Objects aufgebaut werden, um sprachliche Missverständnisse zu vermeiden. Die Attribute und Funktionen die hierfür von dem XWarningDialog benötigt werden, sind in dem nachfolgenden UML-Klassendiagramm wiedergegeben.

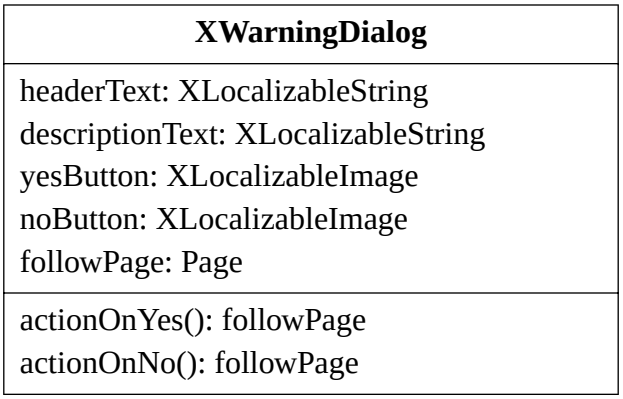


Abbildung 80: UML-Klassendiagramm für die XWarningDialog-Komponente

1. headerText bis noButton: Die Attribute für die sprachunabhängigen Bezeichnungen und Hinweistexte (vgl. auch Abbildung 39).

2. followPage bis actionOnNo: Das Attribut „followPage“ wird von der aufrufenden Komponente (z.B. einer List-Komponente in der die Funktion „Löschen“ betätigt wurde) vor dem Aufruf des XWarningDialog übergeben. Durch dieses Attribut kann die XWarningDialog-Komponente später wieder zur jeweiligen aufrufenden Komponente weiterleiten. Die „actionOnYes“- bzw. „actionOnNo“-Methoden werden durch die Betätigung der entsprechenden YES- bzw. NO-Buttons ausgeführt. Vor der Rückkehr zur aufrufenden Seite (bzw. vor der Weiterleitung zur angegebenen „followPage“) wird innerhalb dieser Methode mit der FollowPage-Komponente kommuniziert und die Entscheidung übermittelt. Die Folgebehandlung wird der aufrufenden bzw. der FollowPage-Komponente überlassen.

7.9 XDataUpload

Die XDataUpload-Komponente ermöglicht es, beliebige Multimedia-Daten, wie z.B. Bilder, HTML-Texte oder auch Videos von einem lokalen Rechner über das Internet zum Server zu übertragen. In der nachfolgenden Abbildung ist eine mögliche Nutzeroberfläche für die XDataUpload-Komponente wiedergegeben:

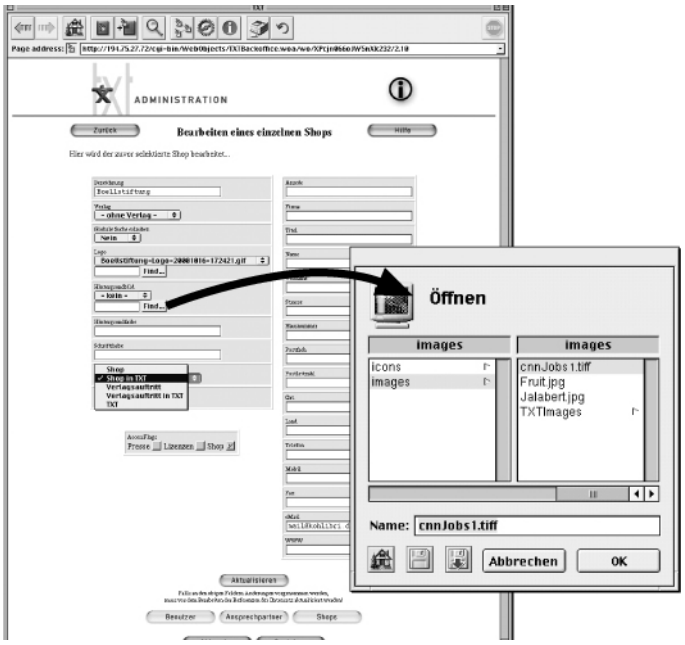


Abbildung 81: XDataUpload-Komponente

Durch die Betätigung von „Find“ öffnet sich der Browser (je nach Betriebssystem „Browser“, „Finder“ oder „Explorer“), der die Auswahl einer Multimedia-Datei ermöglicht.

Die Bezeichnung, unter der sie im Server abgelegt wird, erscheint in dem nebenstehenden Textfeld. Durch eine zusätzliche Choice, die z.B. die Angabe des Typs der Multimedia-Datei ermöglicht, kann beispielsweise das Einsortieren der Datei in entsprechende serverseitige Multimedia-Ordner unterstützt werden. Neben dem Array, in dem die String-Objekte für die Choice abgelegt werden, benötigt die XDataUpload-Komponente eine individuell zu implementierende Methode, die für die Ablage der entsprechenden Multimedia-Dateien in dem jeweiligen Systemen zuständig ist.

7.10 XArchiver

Analog zu den Adapter-Objekten, die die Datenhaltungsebene von der Business-Logik-Ebene trennen (vergleiche Kapitel 2) werden im folgenden Adapter- bzw. XArchiver-Objekte vorgestellt. Diese sind in der Lage, Business-Objects in unterschiedlichen Formaten und Datenhaltungssystemen abzuspeichern. Im Rahmen der Implementierungsarbeiten zu dieser Arbeit (vergleiche Kapitel 8), wurden verschiedenste XArchiver entwickelt und eingesetzt. Zu ihnen gehört ein XML-Archiver, der es ermöglicht, ein Array von Business-Objects als XML-Datei zu speichern, ein XBookHit-Archiver, der es ermöglicht die Autoren- und Artikel-Objekte aus dem Portal-System für Buchverlage (TXT) in die fachspezifische Anwendung BookHit zu exportieren, ein XSAP-Archiver, der genutzt wurde, um Business-Objects in SAP Anwendungen zu exportieren, ein Code-Archiver, mit dem Business-Objects im Binär-Format gespeichert werden können und verschiedene XDB-Archiver (DBOracle-Archiver, DBSybase-Archiver, DBOpenBase-Archiver, DBOpenBaseLight-Archiver), um Business-Objects in Datenbanken zu speichern. Jede der XArchiver-Implementierung unterscheidet sich dabei von den anderen und ist speziell auf die Bedürfnisse des jeweiligen „Speichermediums“ zugeschnitten. Ein DB-Archiver beispielsweise, der datenbankseitig mit der „passenden“ SQL-Dialektik des eingesetzten DBRMS zu kommunizieren hat, benötigt beispielsweise individuelle Informationen über den Ort der Datenbank (z.B. eine IP-Adresse) und entsprechende Zugriffsberechtigungen (z.B. Login und Passwort). In der Praxis variieren die Anzahl und der Typ der benötigten Zugangsdaten von Datenbank-Management-Systemen zu Datenbank-Management-Systemen. Andere XArchiver, wie z.B. der XML-Archiver benötigen als Zusatzinformation höchstens eine Pfadangabe zu dem gewünschten Speicherort. Interessant aber sind nicht die Unterschiede und Eigenheiten der jeweiligen Implementierungen (auf die deshalb hier auch nicht weiter eingegangen werden soll), sondern die eine entscheidende Gemeinsamkeit: alle XArchiver sollen das gleiche einheitliche Interface in Richtung Business-Logik-Ebene (vergleiche Kapitel 2) aufweisen, so daß sie immer auf dieselbe, möglichst einfache Weise genutzt werden können. Die im Rahmen dieser Arbeit implementierten XArchiver besitzen alle die gleiche Methode um Business-Objects zu speichern:

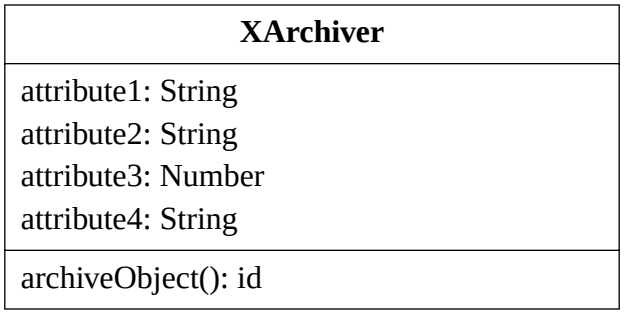


Abbildung 82: UML-Klassendiagramm für die XArchiver-Komponente

1. archiveObject: Die einheitliche Methode die alle XArchiver mindestens anbieten, ermöglicht das Abspeichern von beliebigen Objekten.

2. attribute1 bis attribute4: Die Art und die Anzahl der Attribute eines XArchivers kann in den meisten Fällen nicht im vorhinein bestimmt werden. Sie ist dem jeweiligen Speichermedium bzw. der jeweiligen Datenquelle individuell anzupassen. Je nach Implementierung können verschiedene zusätzliche Attribute benötigt werden, die bei der Instanziierung eines XArchivers eventuell bekannt sein müssen.

7.11 XUnarchiver

Das XUnarchiver-Object ist das passende Gegenstück zum XArchiver. Objekte, die mit XArchivern gespeichert wurden, können mit dem entsprechenden XUnarchiver wieder „geladen“ werden. Genau wie beim XArchiver sollten alle XUnarchiver-Implementierungen die gleiche einheitliche Methode für das Einlesen von Objekten besitzen.

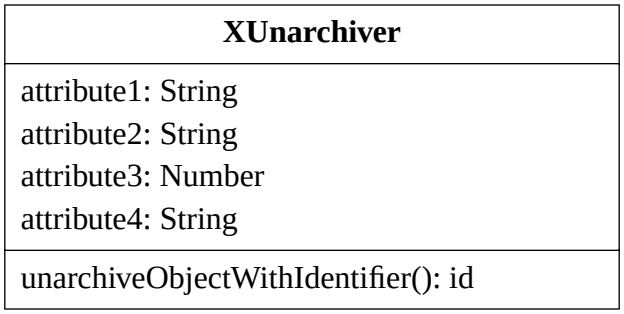


Abbildung 83: UML-Klassendiagramm für die XUnarchiver-Komponente

1. attribute1 bis attribute4: Wie schon beim XArchiver-Objekt variieren Art und Anzahl der benötigten Attribute mit der verwendeten Datenquelle.

2. unarchiveObjectWithIdentifier: Die einheitliche Methode die alle XArchiver mindestens anbieten, ermöglicht das Einlesen von beliebigen Objekten. Die "unarchiveObjectWithIdentifier"-Methode besitzt mindestens immer einen Parameter, der z.B. den Pfadnamen einer XML-Datei enthält. Einige XUnarchiver, wie z.B. ein DB-Unarchiver benötigen zum Teil komplexere Such-Statements als Parameter, um die gewünschten Objekte in einer Datenbank identifizieren zu können.

7.12 XDirectAction

Bei web-basierten eCommerce-Anwendungen handelt es sich um hochdynamische Systeme, die keinesfalls mit statischen HTML-Seiten verglichen werden können. Wie in den vorangegangenen Kapiteln mehrfach beschrieben (vergleiche dazu die Kapitel 2 und 5) werden die zur Anzeige kommenden HTML-Beschreibungen bzw. -Oberflächen erst unmittelbar vor der Anzeige dynamisch erzeugt. Dies führt dazu, daß die „normalen“ HTML-Verzweigungen bzw. „Hyperlinks“ zu HTML-Seiten, wie sie jedem Web-Designer bekannt sind, zunächst nicht unterstützt werden können. Mit Hilfe von XDirectAction-Komponenten kann diese zunächst verlorengegangene Eigenschaft wiedergewonnen werden. In der Anwendung TXT (vergleiche Kapitel 8) werden XDirectAction-Komponenten für die Integration von vorhandenen statischen HTML-Seiten genutzt. Jeder Buch-Autor hat dadurch zum Beispiel die Möglichkeit, auf seinen eigenen privaten HTML-Seiten Hyperlinks zu integrieren, die eine feste URL aufrufen, die es ermöglicht, direkt zur entsprechenden Katalogseite im Portal-System zu springen. Eine solche URL hat in etwa folgendes Aussehen:

http://www.txt.de/nameDirectActionObject?ISBN=334587742

Die XDirectAction-Komponenten besitzen entsprechende Attribute (z.B. das Attribut ISBN) und eine entsprechende Methode, die für die Rückgabe bzw. den Aufbau der gewünschten dynamischen HTML-Seiten sorgt. Die Mehrzahl der Entwicklungsumgebungen für web-basierte Anwendungen besitzen heutzutage bereits entsprechende Klassen bzw. Komponenten, die nur noch den eigenen Bedürfnissen angepaßt werden müssen.

7.13 XInlayFileDirectoryBrowser

Die XInlayFileDirectoryBrowser-Komponente erlaubt es, in dem File-System des Servers zu browsen, um dort gewünschte Dateien ausfindig zu machen und auswählen zu können. Bei der Gestaltung des User-Interfaces kann dabei versucht werden, das Aussehen und die Funktionsweise von üblichen „Browsern“, „File-Managern“ oder „Explorern“ innerhalb einer dynamischen Web-Oberfläche nachzuahmen. Abbildung 84 zeigt eine prototypische Umsetzung der XInlayFileDirectoryBrowser-Komponente.

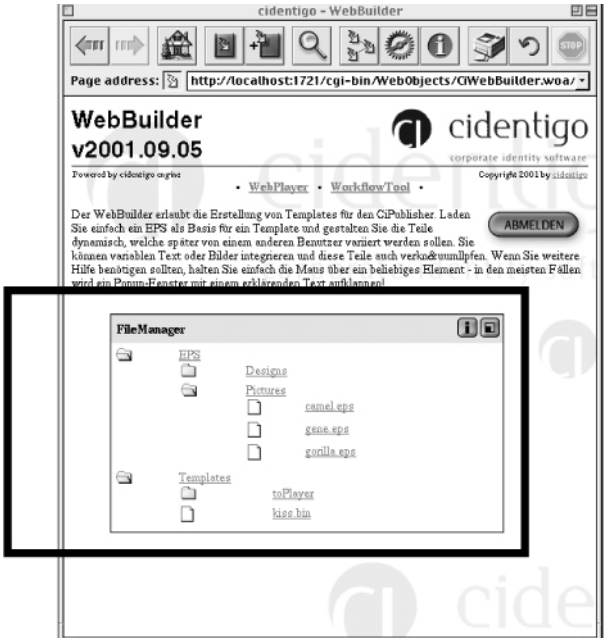


Abbildung 84: XInlayFileDirectoryBrowser-Komponente

Zur Erfüllung ihrer Aufgaben benötigt die XInlayFileDirectoryBrowser-Komponente eine Reihe von Attributen und Methoden, die in dem folgenden UML-Klassendiagramm wiedergegeben sind.

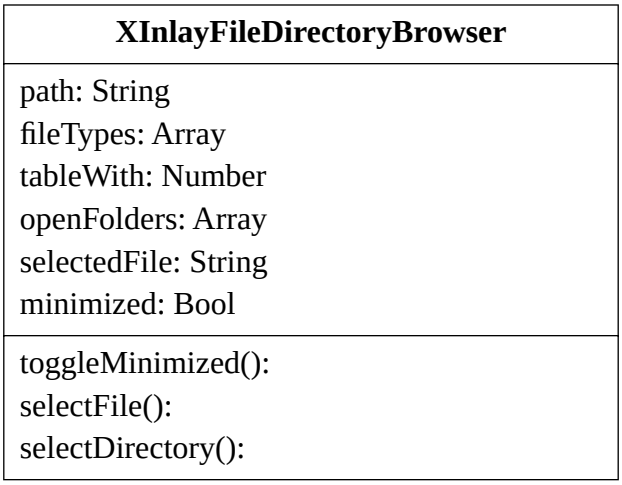


Abbildung 85: UML-Klassendiagramm für den XInlayFileDirectoryBrowser

1. path bis minimized: Mit dem Attribut „path“ wird der Einstiegspfad angegeben. Der Nutzer kann sich nur unterhalb dieses Verzeichnisses bewegen. Durch das Attribut „fileTypes“ kann zusätzlich eine Vorauswahl der anzuzeigenden „Datei-Typen“ vorgenommen werden (z.B. nur Dateien mit den Endungen .jpg, .gif und .eps). Mit dem Attribut „size“ kann die Breite der XInlayFileDirectoryBrowser-Komponente eingestellt werden. Mit dem Attribut „openFolders“ können die geöffneten („aufgeklappten“) bzw. „geschlossenen“ Ordner voneinander unterschieden werden. In dem Attribut „selectedFile“ wird die zuletzt selektierte Datei gespeichert. Mit dem Attribut „minimize“ wird der Zustand der XInlayFileDirectoryBrowser-Komponente verwaltet. Durch Klick auf den „Minimize-Button“ kann die Komponente bis auf die Größe des „Minimize-Button“ verkleinert werden.

2. toggleMinimized bis selectDirectory: Die Methode „toggleMinimized“ schaltet den Zustand im Wechsel von „minimized“ zu „Normalgröße“. Durch die Methode „selectFile“ wird das Attribut „selectedFile“ gesetzt. Mit Hilfe der Methode „selectDirectory“ werden Ordner auf- bzw. zugeklappt.

7.14 XInlayCalendar

Die XInlayCalendar-Komponente, die verschiedene Ansichten eines Kalenders implementiert und geeignet ist, um eine bequeme übersichtliche Auswahl von Kalenderdaten zu ermöglichen, soll an dieser Stelle die Vorstellung von zusätzlichen Komponenten, die den Framework erweitern, abschließen bzw. abrunden. In der Abbildung 86 ist eine Instanz der XInlayCalendar-Komponente dargestellt:

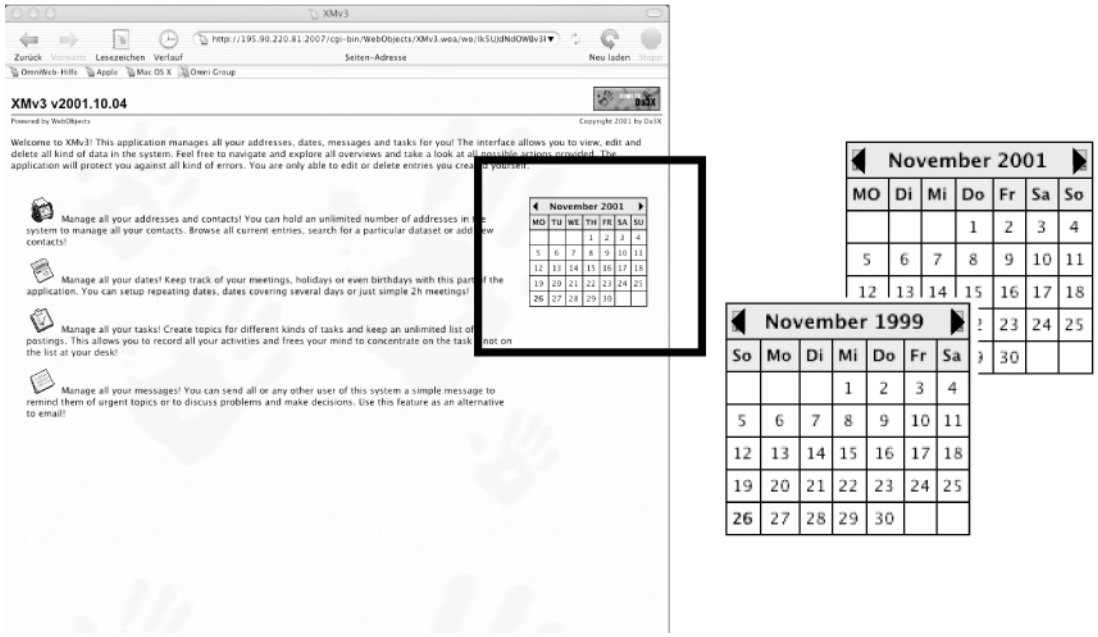


Abbildung 86: XInlayCalendar-Komponente

Die für die Implementierung der XInlayCalendar-Komponente benötigten Attribute und Methoden sind dem folgenden UML-Klassendiagramm zu entnehmen.

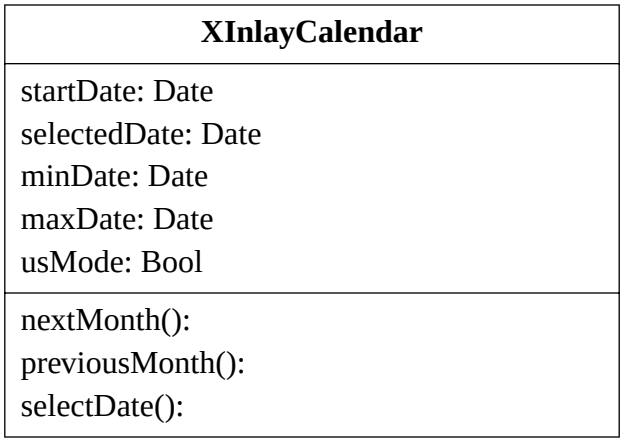


Abbildung 87: UML-Klassendiagramm für die XInlayCalendar-Komponente

1. startDate: Das Attribut „startDate“ wird im Normalfall auf „today“ gesetzt, um mit der Anzeige des aktuellen Wochentags und Monats zu beginnen.

2. selectedDate: Mit dem Attribut „selectedDate“ wird das selektierte Datum gespeichert.

3. minDate: Durch die Attribute „minDate“ und „maxDate“ kann bei Bedarf der darstellbare Zeitraum eingeschränkt werden.

4. nextMonth: Die Methoden „nextMonth“ und „previousMonth“ werden für das „Weiterblättern“ von Monaten genutzt.

5. selectedDate: Die Methode „selectedDate“ wird beim „Klicken“ auf einen Kalendertag ausgeführt und führt zum Überschreiben des alten „selectedDate“-Attributes.

6. usMode: Mit diesem Attribut wird festgelegt, ob sich der Kalender im US-Mode (mit Sonntag als Wochenbeginn) oder im Euro-Mode (mit Montag als Wochenbeginn) aufbauen soll.

7.15 Zusammenfassung „zusätzliche Komponenten“

In diesem Kapitel wurde eine Auswahl von Objekten und Komponenten vorgestellt, die geeignet sind, das in dieser Arbeit vorgestellte Framework von eXBO und eXBO-Facilities zu ergänzen. Um den Rahmen der Arbeit nicht zu sprengen, wurde dabei darauf verzichtet, alle implementierten Objekte und Komponenten aufzuführen und stattdessen eine Auswahl von besonders interessanten bzw. nützlichen Komponenten vorgenommen. Wie bereits in der Einleitung zu diesem Kapitel angesprochen, sollte die Entwicklung eines Frameworks zu keinem Zeitpunkt als abgeschlossen betrachtet werden. Vielmehr sollte man danach streben, das Framework durch jede neue Implementierung, Erfahrung und Erkenntnis fortlaufend zu verbessern und auszubauen.

Kapitel 8

Realisierte Anwendungen

Wie in den vorangegangenen Kapiteln an vielen Stellen erwähnt, wurden im Rahmen dieser Arbeit zahlreiche Prototypen entwickelt. Einerseits dienten diese Prototypen dazu, die Ansätze und Konzepte zu testen, andererseits ermöglichten es gerade die umfangreichen Implementierungsarbeiten Erfahrungen und Erkenntnisse zu sammeln, die wiederum durch neue oder verbesserte Ideen reflektiert und in die Arbeit eingebracht werden konnten.

In diesem Kapitel sollen die – zum Teil mehrfach preisgekrönten – eCommerce-Anwendungen vorgestellt werden, die im Rahmen dieser Arbeit als Prototypen umgesetzt wurden. Der umfangreichste und aufwendigste Prototyp TXT (ein Portal-System für Buchverlage) wird in den nachfolgenden Abschnitten in Auszügen vorgestellt.

Die anderen Prototypen werden (in der Reihenfolge ihrer Entstehung) an dieser Stelle nur kurz aufgelistet. Genauere Informationen sind den jeweiligen Literaturverweisen zu entnehmen:

- Shop1: Das erste Shop-System, das in „Look and Feel“ dem damaligen Marktführer Intershop nachempfunden wurde, aber nur sehr wenig wirkliche Funktionalität aufwies. Weiterführende Informationen sind [155] zu entnehmen.

- FactoryCircus: FactoryCircus war das erste vollständige eCommerce-Shop-System, das besonders im Backoffice-Bereich einige Spezialanforderungen befriedigte, die durch handelsübliche Shop-Systeme „von der Stange“ nicht erbracht werden konnten. FactoryCircus war zugleich das erste System, das auch im „realen“ Einsatz getestet und erprobt werden konnte. Weitere Informationen sind [156] zu entnehmen.

- DeAmmo: Mit diesem Prototypen konnte gezeigt werden, daß große Teile des Frameworks und der vorgestellten Konzepte wiederverwendbar sind (und Rekordzeitentwicklungen ermöglichen), auch wenn die zugrundeliegenden Geschäftsmodelle und Business-Objects vollkommen anders sind. Mit dem DeAmmo-Prototypen gelang es dem Auftraggeber, einem der weltgrößten Unternehmen im Bereich DeAmmonition (Vernichtung von Munition), erfolgreich die enormen Rationalisierungspotentiale und die Steigerung der Sicherheitsstandards durch die Einführung von elektronischen Systemen zu demonstrieren. Weitere Informationen sind [157] zu entnehmen.

- ReisenUndSpesen: Dieser Prototyp der ebenfalls im „realen“ Einsatz getestet werden konnte, ermöglichte es Handlungsreisenden (z.B. während einer längeren Dienstreise oder einem längeren Auslandsaufenthalt) per Internetanwendung Reise- und Spesenkosten fortlaufend, d.h. bereits vor der Rückkehr einzureichen. Weitere Informationen sind [158] zu entnehmen.

- MetaShop: MetaShop ist ein hochflexibles eCommerce-Shop-System, das vollständig aus den in dieser Arbeit vorgestellten Objekten und Komponenten zusammegesetzt ist. Weitere Informationen sind [159] zu entnehmen.

8.1 TXT

Der Prototyp TXT, der ein Portalsystem für Buchverlage umsetzt, wird seit längerer Zeit erfolgreich im Wirkbetrieb eingesetzt. Den Kern des Portal-Systems TXT bildet ein – für die Bedürfnisse von Buchverlagen – spezialisiertes Shop-System. Die einzelnen, replizierbaren Buchverlag-Shop-Systeme sind sowohl selbständig als auch innerhalb des Portalsystems betreibbar. Gegenwärtig nutzen ca. 20 zum Teil renommierte Buchverlage, wie z.B. Heinrich-Böll-Stiftung, Rotbuch-Verlag, Europa-Verlag, Antje Kunstmann Verlag oder der Hammer Verlag die Anwendung TXT, um ihre Bücher im Internet zu vermarkten und um sich selbst, ihre Autoren und Veranstaltungen zu präsentieren.

Der Prototyp TXT, der schon während der Entwurfsphase für spätere Weiterentwicklungen konzipiert wurde, ermöglicht es den Buchverlagen in bisher unbekannter Art und Weise detaillierte multimediale Informationen über sich selbst, ihre Autoren und nicht zuletzt über ihre Bücher zu publizieren. Weitere Möglichkeiten, wie z.B. das zur Verfügungstellen von gesonderten Presse-Informationen (z.B. für neu erscheinende Bücher) oder die Möglichkeit Lizenzen für Bücher und andere Artikel zu vertreiben, heben das Portal-System TXT deutlich von anderen wesentlich bekannteren Anwendungen (wie z.B. www.amazon.com oder www.libri.de) ab. Nicht nur die Summe der technischen Neuerungen, die bisher unbekannte Präsentations- und Handelsformen im Buchbereich ermöglichen, auch die gelungenen Gestaltungen der einzelnen Systeme (vergleiche dazu auch das Kapitel 2) trugen dazu bei, daß das System TXT mit mehreren Preisen ausgezeichnet wurde.

8.1.1 Verliehene Preise und Auszeichnungen

- Im Jahr 2000 konnte TXT den „Business-Plan-Wettbewerb“, ausgerichtet vom Berliner Senat und der Investitionsbank Berlin (IBB) vor über 850 Mitbewerbern gewinnen. Weitere Informationen sind z.B. [160] zu entnehmen.

- Im Sommer 2001 wurde TXT bzw. die TXT-Anwendung des Antje Kunstmann Verlages auf der Leipziger Buchmesse mit dem „Buchmarkt-Award“ ausgezeichnet. Weitere Informationen sind z.B. [161] zu entnehmen.

Die Implementierung eines derart komplexen Systems durch ein Kleinstteam von nur zwei Entwicklern wurde überhaupt erst durch den konsequenten Einsatz objektorientierter Konzepte und der Wiederverwendung von fertigen Framework-Komponenten ermöglicht. Durch Prototypen wie TXT können die Potentiale von Frameworks, wie dem hier vorgestellten „eXBO-Framework für eCommerce-Systeme“ eindrucksvoll demonstriert werden.

8.1.2 Datenbank

In der umseitigen Abbildung 88 ist das ER-Modell der zugrundeliegenden relationalen Datenbank abgebildet, in der die Business-Objects permanent gespeichert werden. Ein Teil der Business-Objects aus Kapitel 6, z.B. „Bestellungen“ und „Bestellartikel“, können in der Abbildung in nahezu unveränderter Form wiedererkannt werden. Andere Business-Objects, wie z.B. „Artikel“ sind den spezialisierten Bedürfnissen des Buchhandels angepaßt und durch zahlreiche Attribute erweitert worden. Der geübte Beobachter erkennt in dem abgebildeten ER-Modell weitere Objekte, wie z.B. „Bewertungen“, „Terminkalender“, „Auszeichnungen“ oder „Vormerkungen“, die auf weitere, bisher nicht erwähnte Funktionalitäten hinweisen.

Gemäß der Ausführungen in Kapitel 4 konnte auch die Anwendung TXT in die Bereiche Storefront und Backoffice geteilt werden. Im Falle von TXT wurde die logische Trennung konsequent fortgesetzt und für die verschiedenen Teilbereiche separate Applikationen implementiert, die auf einen gemeinsamen Datenbestand aufsetzen. In den nächsten beiden Abschnitten werden Ausschnitte aus dem TXT-Backoffice und aus den verschiedenen Storefront-Versionen präsentiert.

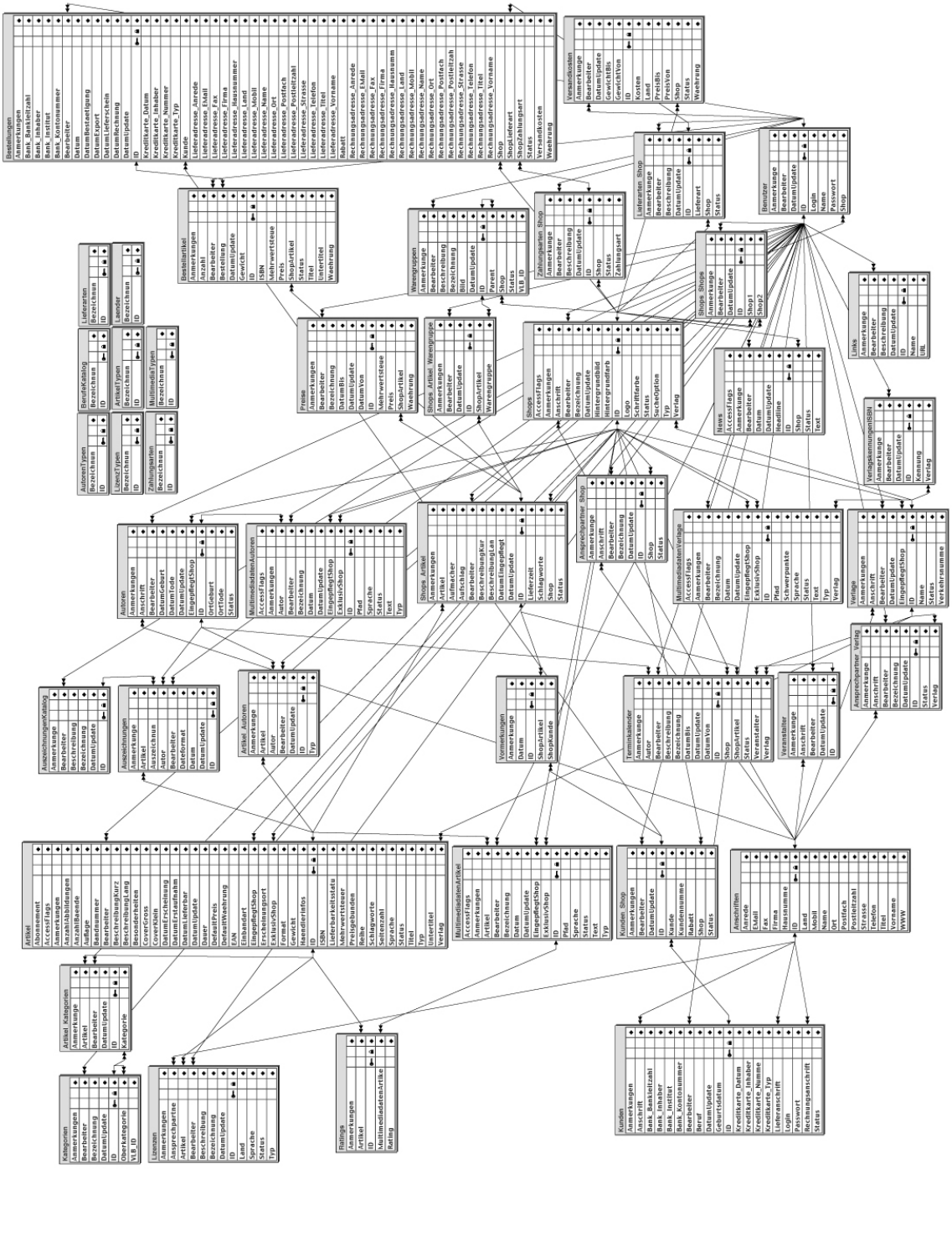


Abbildung 88: ER-Modell der Anwendung TXT

8.1.3 Storefront

Wie in Kapitel 2 vorgeschlagen, wurde bei der Umsetzung des Storefronts auf die konsequente Trennung von Logik und Design geachtet. Dies führte zu einer maximalen Unabhängigkeit der Web-Designer, die eigenständig und ohne Gefahr zu laufen, den Source-Code verändern bzw. zerstören zu können, kreative Ideen zur Oberflächen- und zur Seitengestaltung einbringen konnten. Abbildung 89 versucht einen Teil der dadurch ermöglichten Resultate wiederzugeben. Besonders zu beachten ist die Verschiedenheit der abgebildeten Startseiten, die – kaum zu erkennen – alle auf der gleichen „technischen Komponente“ beruhen.

In Abbildung 94 (siehe Abschnitt 8.1.4 Backoffice) ist die Detail-Ansicht eines Buch-Autors dargestellt. Besonders zu beachten sind die zahlreichen Multimedia-Daten über den Autor. Die Links zu allen seinen Büchern und die Hinweise zu den kommenden Veranstaltungen an denen der Autor teilnimmt.

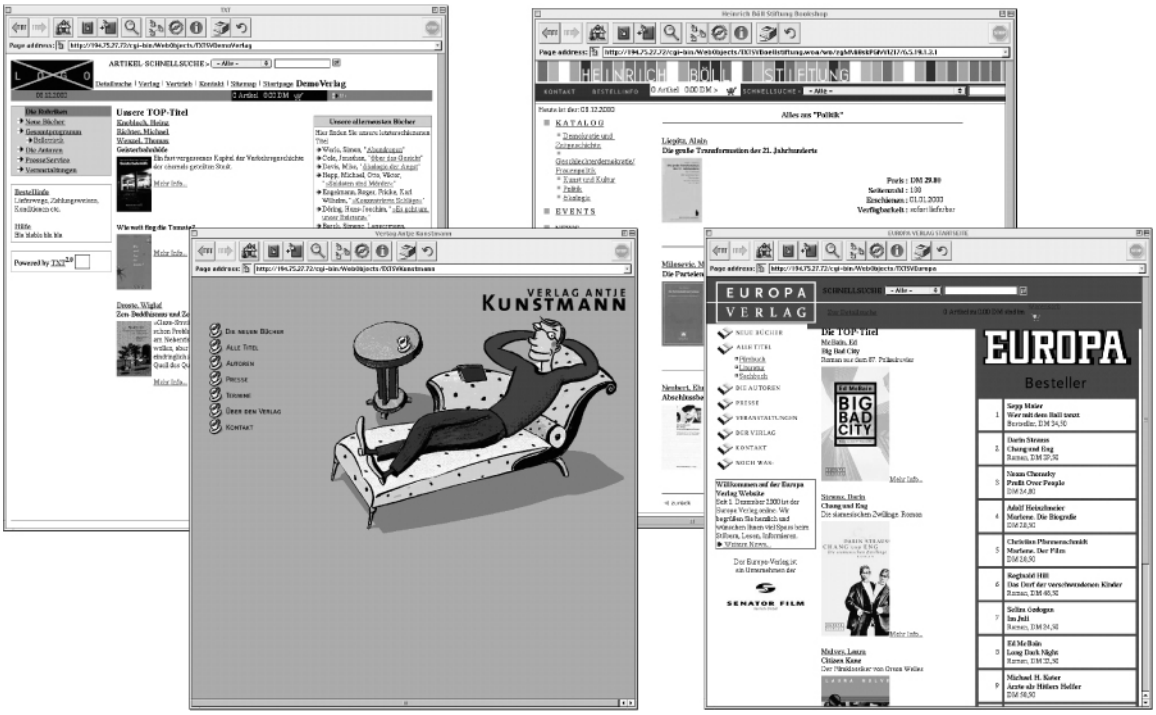


Abbildung 89: Verschiedene Start-Page-Designs mit gleicher „technischer“ Komponente

In Abbildung 90 ist der Warenkorb des TXT-Storefronts dargestellt:

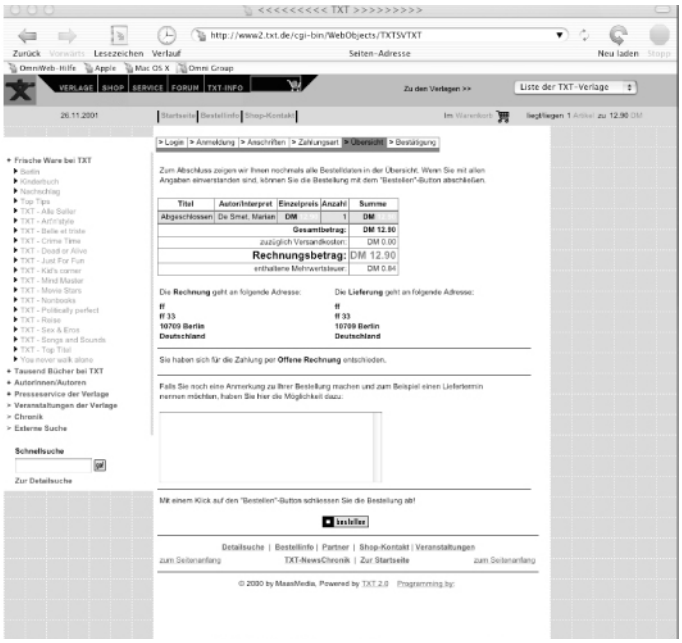


Abbildung 90: Warenkorb des TXT-Systems

Mit der Abbildung 91, die aufzeigt an wie vielen Stellen die List-Komponente (mit jeweils nur leicht angepaßtem Design) wiederverwendet werden konnte, soll nochmals veranschaulicht werden, wie es durch die Wiederverwendung von fertigen Framework-Komponenten möglich werden konnte, mit einem Kleinstteam von Entwicklern eine entsprechend große komplexe Anwendung zu erschaffen.

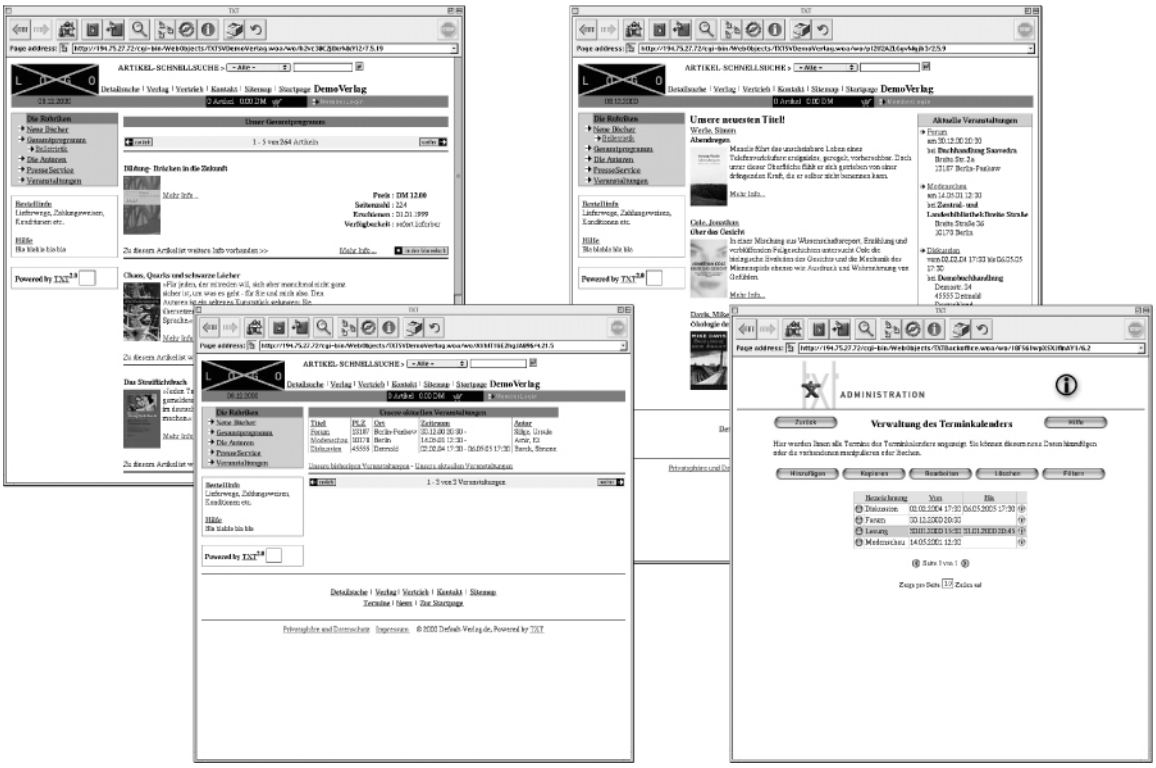


Abbildung 91: Wiederverwendung der List-Komponente

8.1.4 Backoffice

Ähnlich wie beim Storefront ist ein großer Teil der implementierten Komponenten bereits im Laufe der vorangegangenen Kapitel dargestellt worden. Im folgenden wird dementsprechend nur ein kleiner Teil der vorhandenen Backoffice-Komponenten präsentiert. In Abbildung 92 sind die Login-Komponente und verschiedene personalisierte Menu-Pages dargestellt. Abhängig vom Nutzer wird die Menu-Page unterschiedlich gestaltet (siehe z.B. das Verlags-Logo oben rechts). Durch entsprechende Filter wird dafür gesorgt, daß jeder Verlag nur auf seine Bücher zugreifen kann. In der Rolle des Portal-Besitzers bzw. Portal-Verwalters (siehe Abbildung 92 Mitte) stehen zudem weitergehende Menüpunkte zur Verfügung als für einen

„normalen“ Buchverlag (siehe Abbildung 92 rechte Seite).

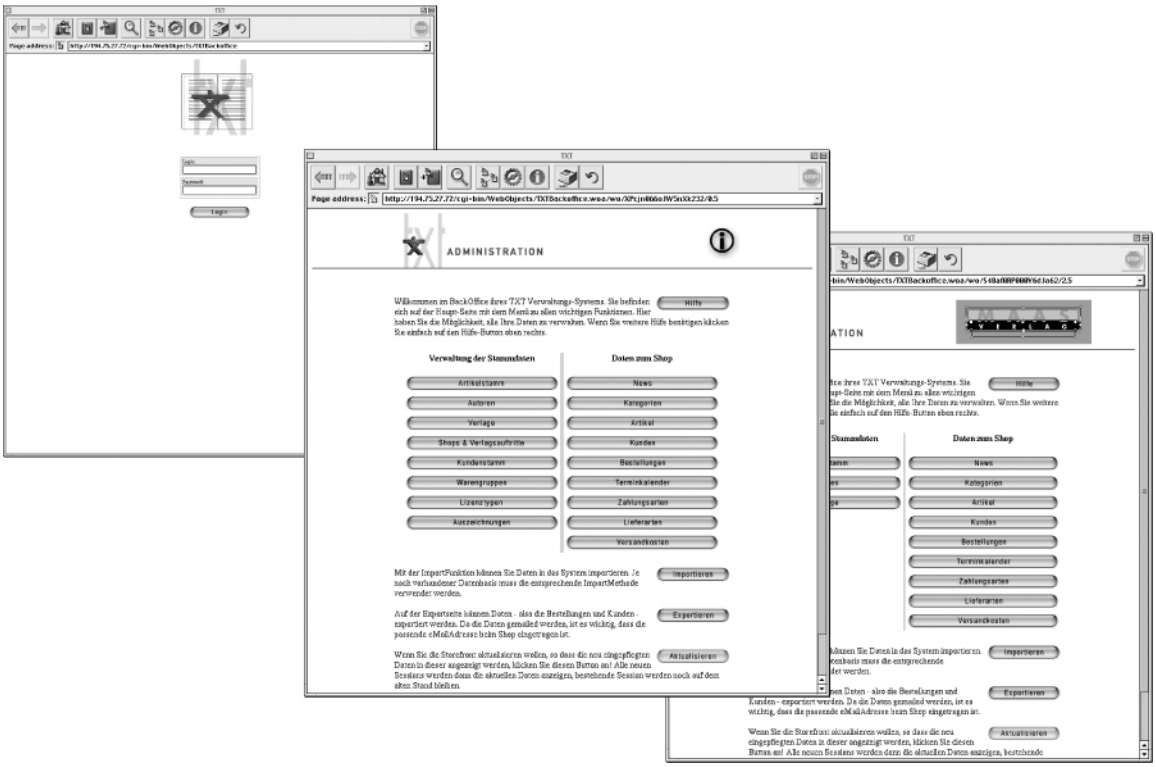


Abbildung 92: Personalisierte Menu-Komponente

Abschließend zeigen die Abbildungen 93 und 94 das Arbeiten mit Autor-Objekten. In Abbildung 93 wird unter Verwendung der eXBO-Facility List und der eXBO-Facility Edit ein Autor-Objekt bearbeitet. Die Resultate in der Storefront sind in der Abbildung 94 dargestellt.

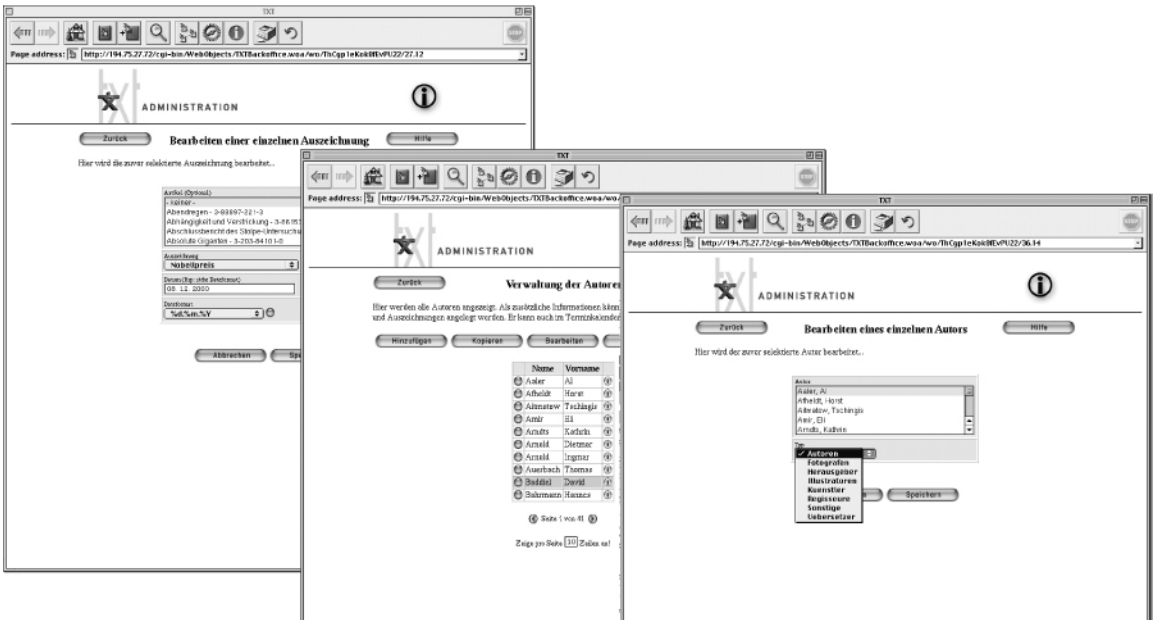


Abbildung 93: Bearbeiten eines Autor-eXBOs

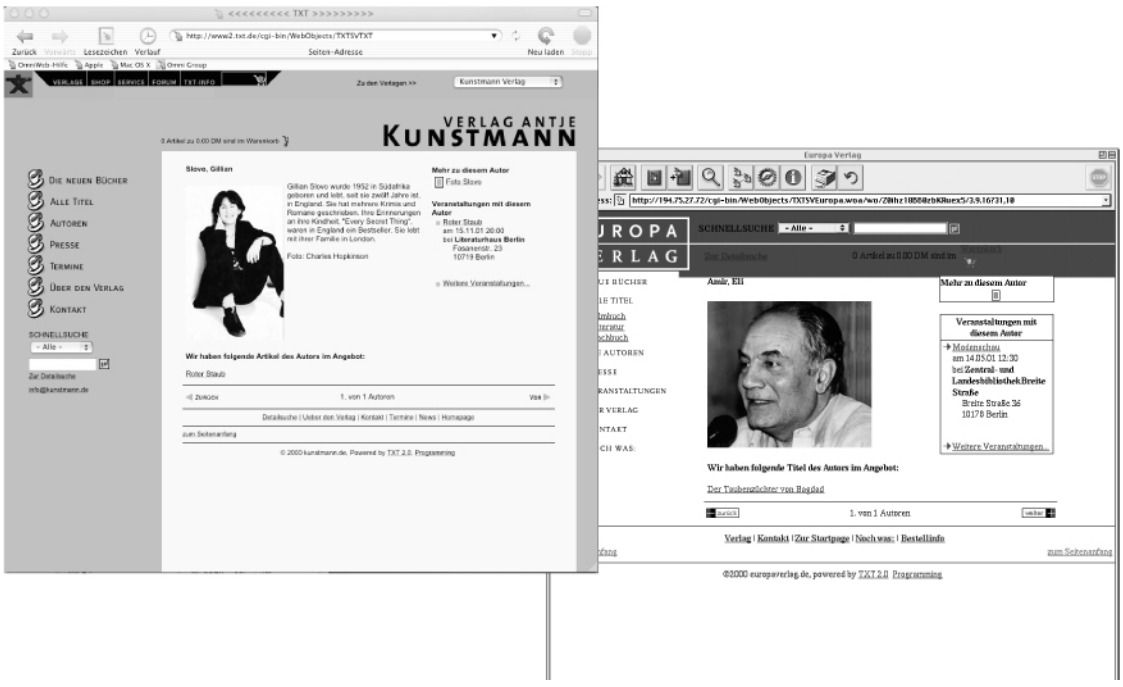


Abbildung 94: Das Resultat in der Storefront

Kapitel 9

Zusammenfassung und Ausblick

In dieser Arbeit wurde ein Framework vorgestellt, mit dem Unternehmen in die Lage versetzt werden, deutlich schneller als bisher und deutlich kostengünstiger als bisher eCommerce-Systeme zu erstellen, die elektronische Umsetzungen von Geschäftsmodellen darstellen.

Dies gilt sowohl für neuartige Geschäftsmodelle, bei denen es oft von besonderer Bedeutung ist, sie besonders schnell bzw. als erster vermarkten zu können, als auch für vorhandene Geschäftsmodelle von Unternehmen, die durch elektronische Lösungen optimiert und rationalisiert werden sollen.

Desweiteren wurden in dieser Arbeit Architekturen und Entwurfskonzepte für eCommerce-Systeme vorgestellt, die es ermöglichen ein Maximum an Erweiterbarkeit, Flexibilität und Integrationsfähigkeit in vorhandenen und zukünftigen IT-Strukturen zu gewährleisten.

Die Betrachtung von Geschäftsobjekten, von Objekten also die ihre Entsprechung in der real

existierenden Geschäftwelt haben, ermöglicht dabei eine Annäherung zwischen der betriebswirtschaftlich-anwendungsbezogenen Sichtweise von Auftraggebern und Managern und der technisch-implementierungsbezogenen Sicht von Entwicklern. Die nötige enge Zusammenarbeit zwischen Software-Ingenieuren und Experten aus den jeweiligen Geschäftsbereichen, um schnell und korrekt Anforderungen aus der Geschäftswelt in einsatzfähige Systeme umzusetzen, wird durch diese Betrachtungsweise optimal unterstützt.

Durch die Einführung von eXBO, die in der Lage sind verschiedenste, spezialisierte Repräsentationen ihrer Attribute zu verwalten und teilweise sogar ineinander zu überführen, wird es möglich eCommerce-Systeme zu erstellen, die gleichzeitig verschiedene, unterschiedlichste Endgeräte individuell unterstützen können. Darüberhinaus können eXBO selbständig über die Art und Weise Auskunft geben, in der sie bearbeitet werden sollen. Dies ermöglichte die Einführung von automatisierten Content-Management-Komponenten in Form der „eXBO-Facilities“, die eine „extreme“ Wiederverwendbarkeit ermöglichen. Es wurde zum Beispiel gezeigt, wie ein und dieselbe Instanz der generischen eXBO-Facility-Komponente „Edit“ genutzt werden kann, um jedes beliebige eXBO zu bearbeiten. Durch diese Eigenschaften kann die Entwicklungsgeschwindigkeit von eCommerce-Systemen um ein zig-faches erhöht werden.

Durch die teilweise preisgekrönten eCommerce-Systeme, die bereits mit dem Framework erstellt werden konnten, ließen sich mehrere der gesteckten Ziele verifizieren:

- sehr flexible Erweiterungsmöglichkeiten zu jedem Zeitpunkt der Entwicklung und des Betriebes;
- flexibelste Integrationsmöglichkeiten in bestehende IT-Strukturen;
- der Bereich von umsetzbaren Geschäftsmodellen ist breiter als zunächst angenommen;
- der Einsatz von wiederverwendbaren getesteten und erprobten Software-Bausteinen;
- die Möglichkeit grafische Entwicklung von technischer Entwicklung zu trennen bzw. Systeme zu schaffen, deren grafische Oberflächen von Grafikern in Eigenverantwortung und getrennt bearbeitet werden können;
- die Möglichkeit eCommerce-Systeme zu erschaffen, die mit geringstem Mehraufwand in der Lage sind verschiedenste Endgeräte, wie z.B. WebBrowser, WAP-Geräte, PDAs oder auch Printer (bzw. „die Druckvorstufe“) (zum Beispiel für die Herstellung von Katalogen) spezialisiert unterstützen zu kön-

nen;

- Drastisch verkürzte Entwicklungszeiten. Selbst der umfangreichste Prototyp (TXT - der mittlerweile mit Preisen ausgezeichnete Marktplatz für Buchverlage) wurde von nur zwei Entwicklern innerhalb weniger Monate erstellt. Vergleichbare Anwendungen benötigen normalerweise einen Entwicklungsaufwand von mehreren Mannjahren. Die Verkürzung der Entwicklungszeiten muß dabei als mehrstufiger Prozeß angesehen werden:

    - Zunächst steigert die konsequente Anwendung von objektorientierter Programmierung die Entwicklungsgeschwindigkeit von Hause aus.

    - Durch die Verwendung von fertigen, erprobten, wiederverwendbaren Geschäftsobjekten verkürzt sich die Fertigstellungszeit eines eCommerce-Systems ein weiteres Mal.

    - Der Einsatz von eXBO und eXBO-Facilities birgt darüberhinaus das Potential, die Entwicklungsgeschwindigkeit noch ein weiteres mal in bisher unerreichte Größenordnungen zu steigern.

Abschließend bleibt zu sagen, daß es in der Zukunft eine Vielzahl von Elementen, Einflussfaktoren und Gestaltungsparametern (wie z.B. stärkere Personalisierung, mobile Endgeräte, UMTS-Dienste, leistungsfähigere Netzwerkstrukturen, neue Datenformate und vieles mehr) geben wird, die die zukünftige Ausgestaltung und Verbreitung des Electronic-Commerce beeinflussen werden. Auch wenn keine exakten Aussagen über die zukünftigen Entwicklungen des eCommerce getroffen werden können, haben trotzdem fast alle Prognosen eine entscheidende Gemeinsamkeit. Alle sagen dem eCommerce nicht nur eine gute, sondern eine sehr gute Zukunft voraus. Eine Zukunft, in der mit und durch den eCommerce exponentielle Umsatzsteigerungen und enorme Rationalisierungspotentiale zu erreichen sind, eine Zukunft, in der eCommerce eine treibende Kraft nicht nur für die Entwicklung der Software-Industrie, sondern für die Weiterentwicklung der gesamten Weltwirtschaft angesehen wird.

Aber ebenso wenig wie die Erfindung der Eisenbahn im 19. Jahrhundert nicht zur Substitution oder zu Verhinderung des Aufkommens anderer Transportmittel geführt hat (es benutzten noch nie so viele Menschen Autos, Fahrräder, Busse und Flugzeuge wie heutzutage), wird eCommerce herkömmliche Transaktionsprozesse vollständig ablösen. ECommerce wird jedoch Märkte neu definieren und verkrustete Marktstrukturen aufbrechen. Die nötigen leistungsfähigen „Werkzeuge“ bzw. eCommerce-Systeme waren durch hohe

Entwicklungskosten und lange Entwicklungs- und Einführungszeiten bisher zu größten Teilen nur den „großen“ bzw. finanzstärksten Unternehmen vorbehalten.

Durch flexibel erweiterbare „Baukästen“, wie dem in dieser Arbeit vorgestellten *„eXBO-Framework für eCommerce-Systeme“* wird es möglich, leistungsfähigste, individuelle eCommerce-Systeme deutlich schneller als bisher und deutlich kostengünstiger als bisher entwickeln zu können. So kann ein bescheidener Beitrag dazu geleistet werden, die Einstiegsbarrieren für eCommerce zu senken und einer größeren Zahl von kleinen und mittelständigen Unternehmen die Möglichkeit gegeben werden, die prognostizierten Markt- und Rationalisierungspotentiale mit leistungsfähigen, individuellen eCommerce-Systemen auch wirklich nutzen zu können.

Abbildungsverzeichnis

Literaturverzeichnis

[1] Fox, Oliver; Mendes, Manuel: "e-Business and e-Government. Technological aspects and product evaluation". GMD Fokus Competence for Distributed Object Technology, Plattforms and Services Platin. Berlin, 2000

[2] Morgan Stanley Dean Witter: "The B2B Internet Report. Collaborative Commerce ". Morgan Stanley & Co. Incorporated, USA 2000

[3] Forester Research: "The economy and social impact of electronic commerce". USA, 2000

[4] Korte, Werner; Reinhard, Ulrike (Hrsg.): "Who is who in electronic commerce". Whois Verlags- und Vertriebsgesellschaft, Heidelberg 1998

[5] Gehmeyr, Andreas: "Electonic Commerce - ein Überblick". In: OBJEKTspektrum, H.2, 1999, 56-62

[6] Fox, Oliver: "Untersuchungskriterien für e-Commerce Verkaufssysteme". GMD Fokus Competence for Distributed Object Technology, Plattforms and Services Platin. Berlin, 2000

[7] Lincke, David-Michael; Haertsch, Patrick et al: "Integrierte electronic commerce Systeme- Auswahlkriterein und Evaluation aktueller Produktangebote". Universität St. Gallen für Wirtschafts- , Rechts- und Sozialwissenschaften, 1997

[8] Kuri, Jürgen: "Geldbäume im Online-Land. Der Online-Handel zwischen Unternehmen verspricht gigantische Umsätze". In: C´t, H.7, 2000, 190-192

[9] Frauenhofer Paderborn EC Competence Center: "Grundlagen des e-Commerce". Paderborn 2000

[10] Wilde, Michael: "Stahlharte Kommerzwelle. Die nächsten Jahre des Internet". In: C´t, H.6, 1998, 162-169

[11] Hartge, Thomas: "Digitale Geschäfte. Shopping im Cyberspace - Chancen und Prognosen". In: C´t, H.3, 1997, 178-184

[12] Gutowski, Katja: "Spezial: e-commerce. Top oder Flop.com". In: Wirtschafts Woche, H. 18, 2000, 160-210

[13] Schroeder, Michael; Kossel, Axel: "Agenten im Kaufrausch. Perspektiven des elektronischen Handels". In: C´t, H.6,1999, 166-170

[14] Merz, Michael: "Electronic Commerce. Marktmodelle, Anwendungen und Technologien". Dpunkt. Verrlag, Heidelberg 1999

[15] Fischer, Malte; Gutowski, Katja; Gersemann, Olaf: "Spezial Internet. WWW." In: Wirtschafts Woche, H. 7, 2000, 82-87

[16] Zorn, Werner: "Hat Deutschland die Internet-Entwicklung verschlafen?". In: PIK, H. 21, 1998, 19-30

[17] Hruschka, Peter: " Neue Manager braucht das Land!" In: Objektspektrum, H.5, 1996, 24-26

[18] Bachmayer, Martin: "Entwicklung einer prototypischen, rechnergestützten Electronic Commerce Application (Internet Shop) unter Verwendung der WebObjects Technologie". Diplomarbeit an der Fachhochschule für Technik und Wirtschaft Berlin. Berlin, 2000

[19] Van de Weyer, Donald; Bager, Jo: "Am Cyber-Markt sind noch Läden frei. Internet-Miet-Shops: günstiger Einstieg in den e-Commerce". In: C´t, H. 23, 1998, 152-159

[20] Bichler, Martin; Freygner, Alexander: "Online-Shopping-Systeme im WWW. Make or Buy". In: iX, H. 8, 1997, 85-90

[21] Charlton, C.; Gittings, C.; Little,J.; Neilson, I.: "Technology transfer: connect, a strategy for promoting electronic commerce in an economically deprived region". 5th International Conference on Factory 2000, Conference Publication No. 435, Liverpool University, England 1997

[22] Unland, Rainer: "Objektorientierte Datenbanken. Konzepte und Modelle". International Thomson Publishing, Deutschland 1995

[23] Krause, Jörg: "Shop nach Maß". In: Internet Professionell, August 1999, 56-59

[24] Gärtner, Reiner: " Vernetzte Handelskette". Internet Professionell, September 1999, 60-

67

[25] Vermes, Timur: "Operation Nulltarif. Zwei deutsche Hersteller attackierern die Office-Preise". In: C´t, H.3, 1999, 112-114

[26] Seestaedt, Dr. Bernd; Schult, Dr. Thomas J.: "Fischertechnik macht mobil. Spielzeugroboter zum Selbstbau". In: C´t, H.6, 1998, 98

[27] Kossel, Axel: "Einkaufsbummel durchs Netz. Schnell und bequem einkaufen über das Internet". In: C´t, H.15, 1998, 142-150

[28] Albrecht, Thorsten; Eckert, Klaus-Peter; Fox, Oliver; Pusch, Herwart: "Analyse zur Integration von MMDP und Intershop in T-Mall". GMD Fokus Forschungsinstitut für offene Kommunikationssysteme. Berlin, 1998

[29] Menge, Rainald: "Globale Geschäfte im Web: E-Commerce verändert den Handel". In: Byte, Juni 1998, 44-51

[30] Gorgs, Claus: "Paket auf die Hallig. Das weltgrößte Versandhaus startet den ersten bundesweiten Lieferdienst für Lebensmittel im Internet". In: Wirtschafts Woche, H.7, 2000

[31] Kuri, Jürgen: "Modenschau. Nutzen und Nutzung von Web-Server-Software". In: C´t, H.3, 1997, 160-179

[32] V, R.: "Neue Geschäftszeiten". In: Macup, H.4, 1999, 160-165

[33] Bager, Jo: "Fliegende Händler. Merchant Server machen Web-Server zur Ladentheke". In: C´t, H.3, 1997, 170-176

[34] Schmidt, Jürgen: "Dasein oder Nicht-Dasein. Analyse von Ausfallzeiten von Web-Servern". In: C´t, H.8, 2000, 174-179

[35] Fox, Oliver: "Objektorienierte Entwicklungsumgebung für Datenbankanwendungen". Diplomarbeit an der TU Berlin Institut für Software und Theoretische Informatik Fachgebiet Offene Kommunikationssysteme. Berlin, 1996

[36] Steiner, Rene: "Theorie und Praxis relationaler Datenbanken". Vieweg Verlag, Braunschweig/Wiesbaden 1994

[37] POET Software: "XML und das Internet: Katalysatoren für die Zukunft von EDI". In: XML EDI White Paper, Februar 1998, 37-43

[38] Marlin, Steven: "EDI: An engine for electronic commerce". In: Bank systems Technology, Oct. 1997, 42-46

[39] Jung, Frank: "Einsatzbereiche von XML. Ohne XML kein Electronic Business". In:It Fokus, H.3, 2000, 9-14

[40] Beyer, Detlef; Schröter, Andre: "Geschäfte im Web. Mit JavaScript und Perl zum eigenen Online-Shop". In: C´t, H.15, 1997, 290-299

[41] Alber, Thomas: "Servlet Session-Tracking (2)". In: Internet world, November 1999, 84-88

[42] Zierl, Marco: "ASP, CFML und PHP im Vergleich - Seitendynamik". In: Internet professionell, Februar 2000, 38-51

[43] Zierl, Marco: "Aktiv & dynamisch - Dynamische Webseiten-Generierung". In: Internet professionell, Februar 2000, 34-36

[44] ZDNet News: "CCC warnt: Offene Hinterür in Windows NT. Sichere Verschlüsselung mit Windows in Frage gestellt". http://www.zdnet.de/news/artikel/1999/09/06010-wc.html

[45] NT Security: "Gain local administrator Access on NT", Mai 2000. http// www.ntsecurity.net

[46] NT Security: "ASP Bypass IIS settings", Mai 2000. http//www.ntsecurity.net

[47] NT Security: "IIS may expose ASP code", Mai 2000. http//www.ntsecurity.net

[48] Microsoft: "Microsoft Site Server. Evaluation Guide". USA 1998

[49] Valero, Jose; Fox, Oliver: "Untersuchung des P3P Standards, Projekt: Einkaufsasistent".GMD Fokus Institut für offene Kommunikationssysteme (OKS). Berlin, 2000

[50] Sadiq, Waqar; Cummins, Fred: "Developing Business Systems withe CORBA". Cambridge University Press 1998

[51] Tolksdorf, Robert: "Die Sprache des Web: HTML 3. Informationen aufbereiten und präsentieren im Internet". Dpunkt Verlag, Heidelber,1995

[52] Guther, Apitz; Hoffmann, Glaß: "Wissenschaftliches Arbeiten im World-Wide-Web". TU Berlin, November 1995

[53] GMD Fokus: "Design und Modellierung von Dienstmodulen für Business-Support". GMD Fokus Competence Center Platin, Berlin 1999

[54] GMD Fokus: "Studie für T-Berkom zum Projekt Dienstemodule für Business-Support". GMD Fokus Competence Center Platin, Berlin 1999

[55] T-Nova Deutsche Telekom Innovationsgesellschaft mbH: "Leistungsbeschreibung der Diensteplattform für die Virtuelle Kommne". Berlin, 2000

[56] Tschammer, Volker; Ouzounis, Vaggelis: "Plattformdienste zur Unterstützung virtueller Unternehmen im elektronischen Handel". In: PIK, H.21, 1998, 155-161

[57] TINA-C: "Overall Concepts and Principles of TINA". Berlin, 1995

[58] Eckert, Klaus-P.; Moeller, Eckhard; Schoo, Peter; Schürmann, Gerd; Tschichholz, Michael: "Verteilte Objekttechnologie in der Telekommunikation". In: PIK 21, 1998, 149-154

[59] Eckert, Klaus-P.; Schürmann, Gerd: "Die ´Telecommunications Information Networking Architecture´ TINA". In: PIK 19, 1996, 225-227

[60] Burkhardt, Rainer: "Die Unified Modeling Language - der neue Standard für die objektorientierte Modellierung?". In: Objektspektrum, H.1, 1997, 57-62

[61] Hruschka, Peter: "Ein pragmatisches Vorgehensmodell für die UML". In: Objektspektrum, H.2, 1998, 34-45

[62] Oestereich, Bernd: "Objektorientierte Geschäftsporzeßmodellierung mit der UML". In: Objektspektrum, H.2, 1998, 48-52

[63] Lewis, David: "A development methodology for service management systems using UML". Department of Computer Science, University College London.

[64] Burkhardt, Rainer; Hruschka, Peter; et al: "UML auf gut deutsch". In: Objektspektrum,

H.5, 1998, 48-49

[65] Wahl, Günter: " UML kompakt". In: Objektspektrum, H.2, 1998, 22-33

[66] Fowler, Martin; Scott, Kendall: "UML konzentriert. Die neue Standard-Objektmodellierungssprache anwenden". Addison-Wesley Verlag, USA 1998

[67] WAP Forum White Paper: "Wireless application protocol". Oct., 1999

[68] DV-Org.: "WAP objects". http//www.wapobjects.de

[69] DV-Org.: "Pressemitteilung der DV-ORG Team GmbH: Dynamische WAP-Dienste mit dem ´WAPobjects´ Framework". http//www.wapobjects.de

[70] Luckhardt, Norbert: "Siegelringe - Digitale Signaturen und Chipkarten". In: C´t, H.6, 1999, 172

[71] Blakowski, Gerold; Blum, Christof; Gerlhof, Carsten: "Zahlungsmittel für das Internet". In: PIK, H.20, 1997, 138-147

[72] Cianciolo, Dr. Sabine: "Kommerzveranstaltung. Internet World: Wege zum elektronischen Marktplatz". In: C´t, H.8, 1998

[73] Wasmeier, Michael: "Web-Währungen. Online-Bezahlungsverfahren für ECommerce". In: C´t, H.11, 1998, 152-156

[74] Deskin, Jonathan: "Schutzmechanismen für E-Commerce - Systeme". In: Obektspektrum, H.6, 1998, 24-30

[75] Manninger, M.; Göschka, K.M.; Dietrich, D.: "Die Smart Card im Internet. Internet-Kommerz und die Verbesserung der Sicherheit durch die Smart Card". In: PIK, H.20, 1997, 148-154

[76] Wrona, Konrad; Keller, Ralf; Williams, Fiona: "Electronic Commerce in Deutschland und weltweit: Neue Zahlungssysteme und ihre Anwendungsbereiche". In: PIK, H.21, 1998, 3-10

[77] Richter, Tim: "Electronic Commerce Plattformen. Eine Evaluation der marktführenden Anwendungen". Studienarbeit an der TU Berlin, Lehrstuhl für offene Kommunikationssysteme Institut für Informatik. Berlin, 2000

[78] Plog, Felix: "Electronic Commerce Plattformen. Eine Evaluation der marktführenden Anwendungen". Studienarbeit an der TU Berlin, Lehrstuhl für offene Kommunikationssysteme Institut für Informatik. Berlin, 2000

[79] Intershop: "User´s Guide". Intershop Communication GmbH, Jena/Deutschland 1998

[80] Intershop: "Intershop Business Case Study". 2000

[81] Intershop: "Intershop Entinity White Paper". 2000

[82] IBM: "IBM WebSphere Commerce Suite Fundamentals". IBM, USA 2000

[83] iCat: "iCat Carbo Editor User Guide". iCat Corporation, USA 1998

[84] iCat: "iCat Commerce Publisher User Guide". iCat Corporation, USA 1998

[85] iCat: "iCat Getting Started Guide". iCat Corporation, USA 1998

[86] OpenShop: "OpenShop Business 1.01", 2000. http://www.openshop.de

[87] OpenShop: "OpenShop Business Handbuch". OpenShop Internet GmbH, Ulm 1999

[88] Mercantec: "Mercantes Soft Cart". Mercantec Inc., Oldenburg/Deutschland 1999

[89] Mercantec: "Mercantes Soft Cart User´s Guide Version 5". Mercantec Inc., Oldenburg/ Deutschland 1999

[90] iCat: "iCat Electronic Commerce Suite 3.01". iCat Corporation, USA 2000

[91] Commerce One: "Commerce One Online Guide". http://www.commerceone.com

[92] Microsoft: "Commercial Internet System (MCIS)". Microsoft Corporation: http:// www.microsoft.com

[93] Microsoft: "Commercial Internet System (MCIS). White Paper". Microsoft Corporation, WA/ USA 1998

[94] Siemens: "Inter Express Service Selection. Intelligent Access Solution". Siemens AG, Germany 1999

[95] Siemens: "Business over IP Solutions". Siemens AG, Germany 1999

[96] Siemens: "Partnership for Business over IP". Siemens AG, Germany 1999

[97] T-Nova: "Leistungsbeschreibung der Diensteplattform für die Virtuelle Kommune". T-Nova, Berlin 2000

[98] Wasmeier, Michael: "Shop in the box. Komplettlösungen für Online-Shops von iCat, Intershop und Microsoft". In: C´t, H.11, 1997, 226-240

[99] Wasmeier, Michael: "Shop in the box. Funktionsweise von Online-Shop-Komplettpaketen". In: C´t, H.7, 1997, 268-275

[100] Hüskes, Ralf; Ehrmann, Stephan: "Großer Auftritt. Innternet-Präsenz für Privatanwender und Firmen". In: C´t, H.63, 1997, 134-142

[101] Running Start: "RS Commerce. The fast easy way to build profitable Web sites".Tucson, AZ: www.running-start.com

[102] OMG: "Business Object DTF Common Business Objects". USA, 1997, Version 1.5

[103] OMG: "Common Business Objects and Business Object Facility". Object Management Group, MA/USA 1996

[104] OMG: "Business Object Domain Task Force RFI ". Object Management Group, MA/USA 1997

[105] IBM: "IBM Common Business Objects. Response to Business Object Task Force RFP". IBM Corporation, MN/USA 1997

[106] IBM; Oracle: "Business Object Facility. Response to Business Object Task Force RFP". IBM Corporation, MN/USA 1997

[107] Data Access; SEMATECH; Prism; IONA: "Business Object Facility. Response to Business Object Task Force RFP". Data Access Technologies, USA 1997

[108] NIIIP: "Task and Session Objects. Response to Business Object Task Force RFP". NIIIP Consortium, USA 1997

[109] Rösch, Martin: "OMG-Standards und ihre Bedeutung für die Praxis". In:

Objektspektrum, H.1, 1994, 19-23

[110] Eeles, Peter; Sims, Oliver: "Building Business Objects". John Wiley& Sons Verlag, N.Y/USA 1998

[111] Ihme, Falko; Sauer, Petra; Reschke, Volker: "Der Einfluß objektorientierter Softwareentwicklung auf das Projektmanagement". In: Objektspektrum, H.5, 1996, 27-34

[112] Frost, Stuart; Allen, Paul: "Businessorientierte Komponenten-Modellierung". In: Objektspektrum, H.3, 1997, 32-39

[113] Itter, Ralf; Schumann,Matthias: "Internet-Technologie als universelle Middleware für Business-Objekte". Institut für Wirtschaftsinformatik, Universität Göttingen

[114] Hertel, Uwe: "Spezifikation und Implementierung von flexiblen Business-Objekten für eine sich wandelnde Welt". In: Objektspektrum, H.6, 1995, 36-39

[115] Fox, Oliver; Solarski, Marcin: "The current state of the Object Management Group´s standardization process for CORBA Business Objects and Components". GMD Fokus Berlin

[116] Parsons, Jim: "Komponentenbasiertes Entwickeln im Bereich e-Commerce". In: Objektspektrum, H.6, 1999, 36-40

[117] Taylor, David A.: "Wie leitet man sein erstes objektorientiertes Projekt?" In: Objektspektrum, H.4, 1995, 36-37

[118] Böhme, Andreas: "OO-Projekte als Dienstleistung". In: Objektspektrum, H.5, 1998, 54-58

[119] Loviscach, Dr. Jörn: "Paßgenau. Mit Komponenten zur persönlichen Software". In: C´t, H. 3, 1997, 224-228

[120] Vorwerk, Raymund: " Verteilung von Objeken". In: Objektspektrum, H.5, 1994, 42-44

[121] Eisenecker, Ulrich; Hirscfeld, Robert: "Eine Einführung in Hewlett-Packards Distributed Smalltalk". In: Objektspektrum, H.5, 1994, 20-28

[122] Rösch, Martin: "Sein und Schein von Business-Objekten". In: Objektspektrum, H.6,

1995, 10-14

[123] Lewis, David: "Implementing Management Business Process from Reusable Components: A Development Methodology". University College London

[124] Heuer-Hasenpatt, H.; Hollunder, Bernhard; et al: "Bausteinorientierte Anwendungsentwicklung: Vorraussetzungen, Anforderungen und Auswirkungen". In: Objektspektrum, H.3, 1997, 40-51

[125] Taylor, David: "Das Aufspüren geeigneter Objekte". In: Objektspektrum, H.1, 1994, 50-51

[126] Rösch, Martin: "Business-Objekte vereinfachen die Struktur von Informationssystemen". In: Objektspektrum, H.4, 1995, 70-75

[127] Fox, Oliver: "IBM´s San Francisco Project". GMD Fokus Forschungszentrum Informatiionstechnik GmbH. Berlin, 1998

[128] IBM: "IBM San Francisco Business Process Components for Java", 1998. http// www.ibm.com/java/sanfrancisco

[129] IBM: "San Francisco Road Map". IBM Corporation, MN/USA 1998

[130] IBM: "San Francisco Developer´s Guide". IBM Corporation, MN/USA 1998

[131] IBM: "San Francisco Java Business Process Components". IBM Corporation, MN/ USA 1998

[132] IBM: "San Francisco Administering and Configuring San Francisco". IBM Corporation, MN/USA 1998

[133] Wolff, Eberhard: "San Francisco: Framework für Geschäftsanwendungen. Im Rahmen". In: iX, H.7,1998, 116-121

[134] Garfinkel, Simson; Mahoney, Michael: " NeXTSTEP Programming. Step one: Object-Oriented Applications". Springer Verlag, Cambridge/ USA 1993

[135] NeXT: "The NeXTSTEP Advantage. A technical guidebook to object-oriented programming". NeXT Computer Inc., CA./USA 1993

[136] NeXT: "Discovering openstep: Openstep". NeXT Computer Inc., CA./USA 1993

[137] NeXT: "Object-oriented programming and the objective C language". NeXT Computer Inc., CA./USA 1993

[138] Apple: "WebObjects Developer´s Guide". USA 1998

[139] Apple: "Getting Started with WebObjects". USA 1998

[140] Apple: "Enterprise Objects Framework Developer´s Guide". USA 1998

[141] Apple: "Programming WebObjects I". Apple Computer Inc., CA/USA 1998

[142] Apple: "Programming WebObjects II". Apple Computer Inc., CA/USA 1998

[143] Martins, Filipe; Kobylinska, Anna: "Apropos Mac OS X Server. Ein Mach-Kernel-Unix für das Internet-Zeitalter". Addison-Wesley Verlage, München 2000

[144] Backlin, Gene: " Developing NeXTSTEP Applications". Sams Publishing, USA 1995

[145] Busse, F.; Weller, A.; Müller, R. et al: "Objektorientierung zur effizienten Planung vo Mobilfunknetzen". In: Objektspektrum, H.3, 1996, 16-25

[146] Gervae, Nik; Clark, Peter: "Developing Business Applications with Open Step". Springer Verlag, N.Y.1997

[147] Craighill, Nancy: "OpenStep for Enterprises". John Wiley&Sons Verlag, Canada 1997

[148] Lipps, Peter: "Enterprise Objects Framework - Fachspezifische Objekte in OpenStep". In: Objektspektrum, H.5, 1995, 50-59

[149] Booker, Ellis: "NASA Center Offers Formula for Picking Middleware". http://www.intranetworld.com

[150] NASA: "Web Middleware Study Final Results". Mai, 1997

[151] Christiani, Andreas: "OpenStep aus Programmierersicht. Alles eins". In: iX, H.5, 1997, 142-147

[152] Braun, Primin: "Business-Software - ein 00-Erfahrungsbericht". In:

Objektspektrum,H.6, 1995, 30-35

[153] Röscheisen, Hermann: "Application-Server. Das Ganze im einzelnen". In: Screen Business Online,H.5, 1999, 16-20

[154] SQLI: "Application Server Study". 1999

[155] Fox, Oliver: "Prototyp 1. Machbarkeitsbeispiel für Shopsystem". GMD Fokus Forschungszentrum Informationstechnik GmbH. Berlin, 1997

[156] Fox, Oliver: "Prototyp 2. Factory Circus". GMD Fokus Forschungszentrum Informationstechnik GmbH. Berlin, 1998

[157] Fox, Oliver: "Das DeAmmo-Projekt". GMD Fokus Forschungszentrum Informationstechnik GmbH. Berlin, 1999

[158] Fox, Oliver: "Prototyp 3. Reisen und Spesen". GMD Fokus Forschungszentrum Informationstechnik GmbH. Berlin, 1999

[159] Fox, Oliver: "MetaShop". GMD Fokus Forschungszentrum Informationstechnik GmbH, IKV TechnologiesAG, Berlin, 2002

[160] TXT: "Preisverleihung BusinessPlan Wettbewerb", www.txt.de/ BusinessPlanWettbewerb, TXT GmbH, Berlin, 2000

[161] TXT: "Buchmesse Leipzig - Preisverleihung Buchmarkt-Award", www.txt.de/ Buchmarkt-Award, TXT GmbH, Berlin, 2001

[162] Booch, Grady: "Entwurfsmuster". In: Objektspektrum, H.1, 1994, 14-18

[163] Sommerlad, Peter: "Entwurfsmuster für Software-Architektur". In: Objektspektrum, H.1, 1991, 16-20

[164] Zimmer,Walter; Neumann, Rainer: "Entwurfskonzepte für die Entwicklung von Frameworks". In: Objektspektrum, H.2, 1996, 22-29

[165] Geppert, Ingo: "ADC- Ein echtes Framework für Benutzungsschnittstellen in VisualWorks". In: Objektspektrum, H.5, 1995, 28-35

[166] Eckstein, Jutta: "Wiew lernt man ein fremdes Framework am besten kennen?". In:

Objektspektrum, H.6, 1998, 80-83

[167] Paulisch, Dr. Frances: "Framework - Ein heißes Thema". In: Objektspektrum, H.5, 1995, 6

[168] Weyerhäuser, Markus: "Taligents CommonPoint-Architektur: Frameworks - mehr als nur Objekte". In: Objektspektrum, H.5, 1995, 60-67

[169] Birrer, Andi; Bischofberger, W.R.; Eggenschwiler, Th.: "Wiederverwendung durch Framework Technik - vom Mythos zur Realität". In: Objektspektrum, H.5, 1995, 18-26

[170] Auszug aus White Paper "Leveraging Object-Oriented Frameworks": "Vorteile objektorientierter Frameworks". In: Objektspektrum, H.5, 1995, 10-16

[171] Bäumer, Dirk; Knoll, Rolf; Gryczan, Guido; et al: "Objektorientierte Entwicklung anwendungsspezifischer Rahmenwerke". In: Objektspektrum, H.6, 1995, 48-59

[172] Wegener, Hans: "Für und Wider des Extreme Programming. Extreme Ansichten". In: iX, H.12, 1999

[173] Buschmann, Frank; Meunier, Regine: "Software-Konstruktion mit Entwurfsmustern". In: Objektspektrum, H.5, 1994, 48-57

[174] Tschichholz, Michael; Fox, Oliver: "Pojektantrag zur BMBF-Förderung - Aktive eGovernment Dokumente". GMD Fokus Forschungszentrum Informationstechnik GmbH, IKV^{++} Technologies AG, Berlin 2000

[175] Tambouris, Efthimios; Fox, Oliver, et al: "An Integrated platform for realising online one-stop government - Proposal from IST Project IST-00-4-1A". Archetypon S.A., Greek 2001

[176] Sedeil, Malte; et al: "VeZuDa Leitfaden - Vereinheitlichung bzw. Zusammenführung der verschiedenen Datenstrukturen in der Berliner Verwaltung". Senatsverwaltung für Inneres, Berlin 2000

[177] Fox, Oliver: "eCommerce Shop Systeme im Vegleich". Vortrag, GMD Fokus Forschungszentrum Informationstechnik GmbH, IKV^{++} Technologies AG, Berlin, 2001

[178] Fox, Oliver; et al: "Der NeXT Computer - Der Rechner mit dem das World Wide Web

erfunden wurde". Multimedia- Anwendung, GMD Fokus Forschungszentrum Informationstechnik GmbH, Musum für Verkehr und Technik, Berlin, 1998

[179] Cern: "Cern - where the Web was born". www.cern.ch, 2001

[180] Gamma, Erich; Helm, Richard; Johnson, Ralph; Vlissides, John: "Design Patterns Elements of Reusable Object-Oriented Software". Addison-Wesley, Menlo Park, California, 1995

Index

A

B

C

D

E

F

G

I

K

L

M

N

O

P

R

S

T

U

V

W

X