Diplomarbeit

Persistenz in offenen, verteilten Anwendungssystemen

Network Computing und die Bedeutung von Datenbanksystemen im Rahmen offener, verteilter Informationssysteme

Karim H. Attia
attia@acm.org

Arbeitsbereich Softwaretechnik
Fachbereich Informatik
Universität Hamburg

Diplomarbeit

Persistenz in offenen, verteilten Anwendungssystemen

Network Computing **und die Bedeutung von Datenbanksystemen im Rahmen offener, verteilter Informationssysteme**

Diplomarbeit

Persistenz in offenen, verteilten Anwendungssystemen

Network Computing und die Bedeutung von Datenbanksystemen
im Rahmen offener, verteilter Informationssysteme

Karim H. Attia
attia@acm.org

April 1999

Arbeitsbereich Softwaretechnik
Fachbereich Informatik
Universität Hamburg

Erstbetreuung:
Dr. Ingrid Wetzel, AB Softwaretechnik, FB Informatik, UNI Hamburg

Zweitbetreuung:
Prof. Dr. Florian Matthes, AB Softwaresysteme, TU Harburg

Erklärung

Diese Diplomarbeit wurde dem Fachbereich Informatik der Universität Hamburg zur teilweisen Erfüllung der Anforderungen zur Erlangung des Titels Diplom-Informatiker eingereicht.

Hiermit versichere ich, die vorliegende Diplomarbeit ausschließlich unter Verwendung der angegebenen Quellen und Hilfsmittel selbstständig angefertigt zu haben.

Schenefeld, 01. April 1999

Karim H. Attia

Karim H. Attia
Fritz-Reuter-Platz 3
22869 Schenefeld

Matrikel-Nr.: 3 940 896

ZUSAMMENFASSUNG

Durch die zunehmende Vernetzung vormals zumindest informationstechnisch autarker Organisationseinheiten gewinnt die verteilte Verarbeitung in offenen Systemen kontinuierlich an Bedeutung. Die unter dem Begriff Client/Server-System bekannten Architekturen erfahren durch die Einführung von Anwendungsservern sowie die Überschreitung von Unternehmensgrenzen eine neue Dimension der Komplexität. Das Internet in Form des World Wide Web entwickelt sich zur Standardplattform für die globale Kommunikation. Diese neue Art der verteilten Verarbeitung fasse ich in dieser Arbeit unter dem Begriff *Network Computing* zusammen. In ihr manifestiert sich der Übergang vom Industriezeitalter zum Informationszeitalter.

Über die bereits im Einsatz befindlichen wirtschaftlichen Kooperationsmodelle im Rahmen von EDI bis hin zu Traderkonzepten werden Spezifikationen für die Realisierung verteilter Verarbeitung erstellt und umgesetzt. Der Bereich der Persistenz in diesen Umgebungen wird kaum deutlich artikuliert. Dem gegenüber stehen die bereits ausgereiften Fähigkeiten von Datenbanksystemen, die in die Bereiche der verteilten Verarbeitung hineinreichen (z.B. mit ihren Sicherheits- und Transaktionskonzepten).

Damit stellen sich die Fragen, welche Dienste im Rahmen des *Network Computing* eine Rolle spielen und in welcher Form Datenbanksysteme diese Dienste erbringen, um damit ihrer traditionellen Rolle als Grundlage betrieblicher Informationssysteme weiterhin gerecht werden zu können.

Dazu werde ich zunächst ein Datenbank-Profil erstellen, in dem klassische Datenbankdienste zusammengefaßt werden. Anschließend beschreibe ich die aktuell verwendeten Verteilungsansätze und diskutiere deren jeweilige Vor- und Nachteile. Parallel dazu erläutere ich einen neuen Wirtschaftszweig auf der Basis interaktiver Medien und ergänze die Konzepte mit aktuellen Aspekten und Problemen der betrieblichen Informationsverarbeitung. Dies bildet die Grundlage für die Definition eines Infrastrukturprofils. Aufbauend auf diesem Profil werde ich dann ein *Network Computing*-Referenzmodell entwickeln, in welchem für jeden Bereich des Infrastrukturprofils eine Reihe von Technologien mit Standardprotokollen und -schnittstellen ausgewählt werden. Am Beispiel der Oracle Produkte stelle ich den aktuellen Stand der Unterstützung durch Datenbanktechnologie vor und bewerte ihn anschließend auf der Grundlage der bereitgestellten Profile und des Referenzmodells. Am Ende dieser Arbeit steht schließlich ein Überblick der aktuellen Diskussion zur Zukunft der Informationsgesellschaft sowie ein Ausblick auf deren Perspektiven.

INHALTSVERZEICHNIS

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS

Kapitel 1

EINLEITUNG UND MOTIVATION

Im Rahmen meiner universitären und beruflichen Arbeit habe ich mich auf die Bereiche der Softwaretechnik sowie der Datenbanksysteme spezialisiert. Seit 1987 beobachte ich die Entwicklung dieser beiden Fachrichtungen der Informatik und ihr Verhältnis zueinander. Zwei große Veränderungen sind in diesem Zeitraum festzustellen gewesen:

- Die Einführung von Client/Server-Architekturen.
- Der Siegeszug der objektorientierten Perspektive.

Die beiden Fachrichtungen sind von diesen neuen Rahmenbedingungen in unterschiedlicher Weise beeinflußt worden. Die Softwaretechnik hat die objektorientierte Sichtweise in Form von Methoden, Architekturen und Programmiersprachen umgesetzt. Die Datenbankwelt hat sich im Zusammenhang mit Client/Server-Architekturen gerade im Bereich der relationalen Datenbanken zu einem gewaltigen Markt entwickelt, in dem die objektorientierte Sichtweise noch eine untergeordnete Rolle spielt.

Im Rahmen meiner Studienarbeit *Persistenz in objektorientierten Anwendungssystemen: Über die Abbildung von Objektstrukturen auf relationale Strukturen* [Att96] habe ich Grundlagen einer Verbindung der beiden Ausrichtungen in Form von Abbildungsregeln zwischen den verwendeten Strukturelementen geschaffen und diskutiert. Eine Konvergenz der verschiedenen Sichtweisen ist jedoch bis heute nicht zu beobachten, Versuche wie der *Persistent Object Service* (POS) der OMG sind offiziell gescheitert [OMG97d].

Indes entwickelt sich zur Zeit darüber hinaus noch eine weitere, meiner Ansicht nach weit umfassendere neue Anforderung an die Informatik. Bedingt durch die explosionsartige Zunahme des Internet als Kommunikationsmedium[1] und die dadurch entstandenen wirtschaftlichen Interessen sowie die zunehmende Globalisierung und Konzentration wirtschaftlicher Organisationen stehen wir jetzt vor einem neuen Zeitalter - dem Übergang von der Industriegesellschaft zur Informationsgesellschaft.

Visionäre wie James Martin [Mar96] und Don Tapscot [2] [Tap97, TLT98] sprechen von einer Revolution, andere wie Steven McGeady [McG97] vergleichen die

[1] manifestiert durch das *World Wide Web* (WWW) und *Electronic Mail* (eMail)

[2] laut Hotwired der *President of the Net*,
vgl. `http://www.hotwired.com/synapse/katz/98/05/index1a.html`

Veränderungen[3] mit der Reformation. Einig sind sich alle, daß diese aktuellen Entwicklungen unsere Wirtschaftsordnung und die mit ihr zusammenhängenden sozialen Gefüge nachhaltig beeinflussen werden.

1.1 Problem und Aufgabenstellung

Die sich aus diesen Veränderungen ergebende neue Wirtschaftsordnung beinhaltet die Konvergenz dreier, bisher eher unabhängiger Bereiche:

- **Datenverarbeitung** (*Computing*) in Gestalt der klassischen IT-Unternehmen mit ihren meist produktorientierten, herstellerabhängigen Strategien.
- **Kommunikation** (*Communication*) repräsentiert durch die Leistungen der Telekommunikationsbranche, deren Produkte betont interoperabel sein müssen.
- **Inhalte** (*Content*) in Form der Produkte der Medien- und Unterhaltungsindustrie, die aus den klassischen Medien Druck, Radio und Fernsehen immer stärker in den Online-Bereich drängen.

Diese Bereiche bilden im folgenden das Gerüst für die Motivation eines Referenzmodells für die Entwicklung von Anwendungen des digitalen Zeitalters. Ein Modell, das folgende zwei Grundprinzipien der Informationsgesellschaft unterstützen muß :

- Den **Universellen Zugriff** auf die digitale Welt von jeder Person, von jedem Ort, zu jeder Zeit.
- Die **Verteilte Verarbeitung** als die Fähigkeit, Berechnungen losgelöst vom Nutzungsort der Ergebnisse durchführen zu können sowie die Berechnung auf mehrere Orte oder Maschinen zu verteilen.

Grundlage dafür ist ein **einheitliches Verarbeitungsmodell**, das über eine Menge von Standards und Basisdiensten die Erstellung und Nutzung von Hardware und Software unterstützt und die Interoperabilität erleichtert. Es steht für ein neues Datenverarbeitungsparadigma, das ich *Network Computing* nenne: *Network Computing* als Idealform der offenen, verteilten Informationsverarbeitung.

[3] Die Motivation der einzelnen Veränderungen wird im Rahmen dieser Arbeit nicht umfassend, sondern nur in Form von einzelnen Beispielen dargestellt. Auf die unterschiedlichen Perspektiven in bezug auf veränderungsrelevante Aspekte von Informationssystemen [AMJ+98] wird hier nicht explizit eingegangen.

Im Zusammenhang mit den geschilderten Veränderungen stellen sich nun die folgenden Fragen :

> *Was konkret verbirgt sich hinter dem Konzept des Network Computing*[4] *?*

Und darauf aufbauend :

> *Welche Rolle spielen Datenbanksysteme im Rahmen des Network Computing*[5] *?*

Die erste Frage ist Ausdruck meiner Faszination von der Entstehung einer neuen Gesellschafts- und Wirtschaftsform auf der Basis digitaler Informationsflüsse. Die zweite Frage basiert auf meinem langjährigen Interesse an Datenbanksystemen und ihrer Anwendung. Der Zusammenhang der Fragen ergibt sich aus der, meiner Meinung nach enormen Bedeutung, die einer Verbindung der beiden Konzepte zukommen würde.

1.2 Vorgehensweise

Um diese beiden Fragen zu beantworten, wähle ich folgende in Abbildung 1.1 dargestellte Vorgehensweise.

Gegliedert in zwei Abschnitte kläre ich zunächst im ersten Teil die abstrakten Grundbegriffe und Prinzipien der drei Themengebiete *Datenbank*, *Verteilung* und *Wirtschaft* und setze diese dann in Bezug zu meiner Arbeit. Das Ergebnis jeder dieser drei Analysen beschließt das jeweilige Kapitel.

In einem zweiten Abschnitt synthetisiere ich dann auf der Grundlage dieser Analyse-Ergebnisse zunächst das Konzept des Network Computing und ermittle anschließend die Rolle von Datenbanksystemen im Rahmen des Network Computing.

Am Anfang dieser Gliederung steht in Kapitel 2 des ersten Abschnitts die Definition abstrakter Datenbankbegriffe sowie die Vorstellung mehrerer konkreter Datenbanksystemgenerationen, deren Eigenschaften ich diskutiere. Aus diesen Analyseergebnissen erstelle ich dann ein Datenbank-Profil, das dann später in Kapitel 6 als Basis für die Beurteilung der Bedeutung von Datenbanksystemen im Rahmen des *Network Computing* dienen wird.

Zunächst jedoch wird das Konzept des *Network Computing* erläutert: einmal aus einer technischen, verteilungsorientierten Perspektive, sodann aus einer organisatorischen, wirtschaftlich orientierten Sicht.

[4] Der Begriff *Network Computing* wird zwar teilweise bereits verwendet, doch läßt sich keine eindeutige Definition finden. Die Verwendung findet meist im Zusammenhang mit einer neuen Generation von Arbeitsplatzrechnern, den sogenannten *Network Computer* [OG98] statt.

[5] Das World Wide Web als eine Grundform des *Network Computing* ist ein gigantisches Informationssystem. In [SSU95] wird beschrieben, daß Webmaster erst langsam begreifen, daß sie eigentlich Datenbankadministratoren sind, allerdings meist ohne ein unterstützendes Datenbankmanagementsystem.

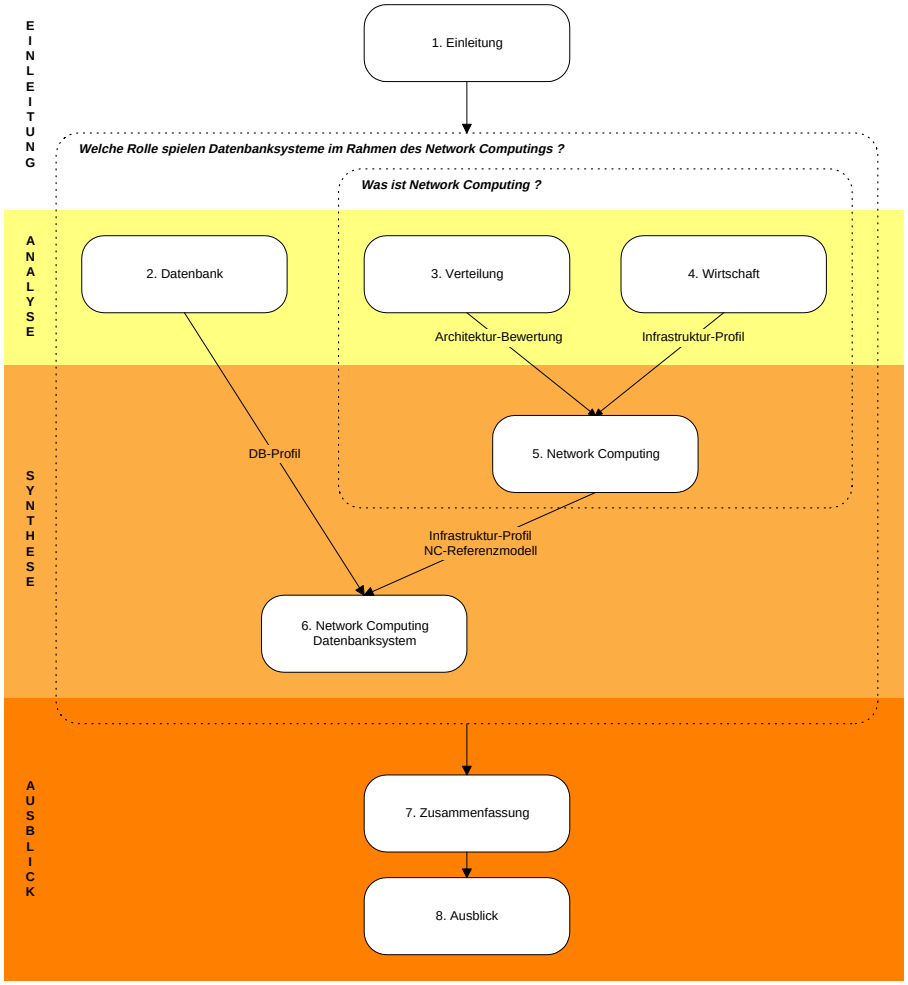

Fig. 1.1: Gliederung der Arbeit

Hierfür stelle ich in Kapitel 3 aktuelle Verteilungsansätze vor und analysiere deren jeweilige Vor- und Nachteile. Diese Analyse-Ergebnisse bilden die Grundlage für die Auswahl einzelner Technologien für das später in Kapitel 5 zu erstellende *Network Computing*-Referenzmodell, in dem die Vorteile der einzelnen Ansätze kombiniert und die jeweiligen Nachteile minimiert werden sollen.

Weitere Anforderungen des *Network Computing* ergeben sich aus den in Kapitel 4 dargestellten Erfordernissen von Organisationen, die im Rahmen der digitalen Wirtschaft agieren. Daher erläutere ich in diesem Kapitel zunächst anhand von Beispielen das Konzept der digitalen Wirtschaft und gehe dann auf aktuelle Aspekte der betrieblichen Informationsverarbeitung ein. Abschließend erstelle ich hier ein Infrastrukturprofil, das notwendige Dienste im Rahmen der digitalen Wirtschaft zusammenfaßt.

Diese gesamten Ergebnisse des ersten Abschnitts fließen nun im zweiten Ab-

schnitt zusammen, der in Kapitel 5 das *Network Computing* als Repräsentation der technischen Grundlage der digitalen Wirtschaft auf der Basis von Internet-Technologien vorstellt.

Unter Berücksichtigung der Ergebnisse des Kapitels 3 wird aus dem Infrastrukturprofil aus Kapitel 4 ein Referenzmodell generiert.

Anschließend werden relevante Standards ausgewählt und vorgestellt. Ergebnis dieses Abschnitts ist das *Network Computing*-Referenzmodell auf der Basis von standardisierten Internet-Technologien und die Klärung der ersten Frage. Zur Verdeutlichung der Konzepte wird ein Szenario bereitgestellt, welches Elemente des *Network Computing*-Referenzmodell im Rahmen einer Anwendung der digitalen Wirtschaft nutzt.

Zur Klärung der zweiten Frage wird in Kapitel 6 eine abstrakte Analyse auf der Basis eines Profilvergleichs zwischen dem Datenbankprofil und dem Infrastrukturprofil vorgenommen. Anschließend wird ein aktueller Vertreter eines Datenbanksystems vorgestellt, seine Dienste mit dem *Network Computing*-Referenzmodell verglichen und ermittelt, in welcher Form dieses Datenbanksystem zur Unterstützung von Anwendungen im Rahmen des *Network Computing* herangezogen werden kann. Als Ergebnis dieser Analyse werde ich dann Forderungen an Datenbanksysteme formulieren, die diese Systeme erfüllen müssen, damit sie als Grundlage offener, verteilter Informationssysteme auf der Basis des *Network Computing* dienen können.

In Kapitel 7 werde ich dann die Ergebnisse dieser Arbeit zusammenfassen und in Kapitel 8 - indem ich die aktuelle Diskussion über zukünftige Entwicklungen des Themenbereiches aufgreife - Perspektiven für das *Network Computing* in offenen verteilten Informationssystemen aufzeigen.

Da relevante Aussagen innerhalb der einzelnen Kapitel der Arbeit durch Beispiele von existierenden bzw. zur Zeit in Realisierung befindlichen Systemen und Technologien belegt werden, ist diese Arbeit auch als eine Zusammenfassung meiner zehnjährigen Erfahrung bei der Erstellung von betrieblichen Informationssystemen unter der Verwendung aller im Rahmen dieser Arbeit beschriebenen Technologien und Konzepte zu betrachten.

Teil I

ANALYSE

Kapitel 2

DATENBANK

Der Schwerpunkt dieser Arbeit liegt in der Betrachtung von betrieblichen Informationssystemen. Informationssysteme sind Programmsysteme, deren Aufgabe es ist, große Mengen langlebiger Daten sicher, flexibel und problemadäquat zu verwalten, zu manipulieren und darzustellen [Mat93]. Diese Aufgaben werden in der Regel durch die Nutzung von Datenbanksystemen erfüllt.

Im Mittelpunkt dieser Arbeit soll daher die Frage stehen, inwieweit Datenbanksysteme ihrer traditionellen Rolle als Grundlage betrieblicher Informationssysteme auch im Rahmen neuer verteilter und vernetzter Organisationskonzepte gerecht werden können.

Dazu beschreibe ich zu Beginn einige grundlegende Begriffe und Konzepte aus der Datenbankwelt in Form abstrakter Datenbankmodelle. Anschließend konkretisiere ich diese Konzepte in bezug auf die Arbeit, indem ich mehrere Datenbanksystemgenerationen vorstelle. An ihnen läßt sich verfolgen, wie sich die Datenbanktechnologie immer wieder weiterentwickelt hat, um erweiterten Anforderungen und Möglichkeiten gerecht zu werden.

Ausgehend von dieser Entwicklung soll daher im Rahmen dieser Arbeit die Frage diskutiert werden, ob sich diese Entwicklung auch auf den Bereich der offenen, verteilten Informationssysteme in Form des *Network Computing* erstreckt hat bzw. wie diese Entwicklung weitergehen müßte.

Dafür werde ich als Ergebnis dieses Kapitels die Aufgaben eines Datenbanksystems zusammenfassen und charakteristische Dienste definieren, die meiner Ansicht nach ein Datenbanksystem ausmachen. Diese werden in Form eines Datenbank-Profils zur Verfügung gestellt, das in Kapitel 6 dann dazu genutzt wird, das Ausmaß der Unterstützung des *Network Computing* durch Datenbanksysteme festzustellen.

2.1 Abstrakte Datenbankmodelle

2.1.1 Grobarchitektur

Entsprechend [STS97] ist eine Datenbank (DB) eine Sammlung von strukturierten Daten, die Fakten über Anwendungen der modellierten Welt repräsentiert, die dauerhaft und weitgehend redundanzfrei gespeichert werden. Das System, das den Umgang mit der Datenbank ermöglicht und organisiert, wird

Datenbankmanagementsystem (DBMS) genannt. Ein Datenbanksystem (DBS) ist die Kombination eines DBMS mit mehreren Datenbanken. Der Zusammenhang dieser Elemente ist in dem Modell in Abbildung 2.1 dargestellt.

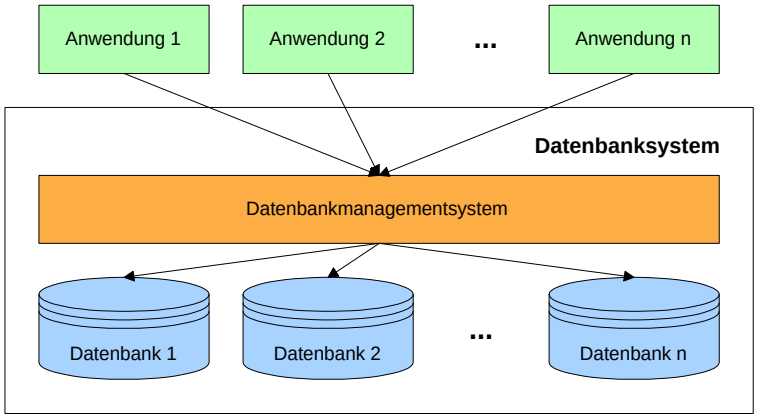

Fig. 2.1: Grobarchitektur eines DBS

2.1.2 Drei-Ebenen-Architektur

Ein Architekturmodell[1] im Datenbankbereich ist das ANSI/SPARC-Referenzmodell [ANS86], aus dem hier in Abbildung 2.2 die *Drei - Ebenen - Architektur* wiedergegeben wird. Sie ist die Grundlage für die logische und physikalische Unabhängigkeit der verwalteten Daten in bezug auf Anwendungen und Speichertechnologien und unterscheidet drei unterschiedliche Sichten der Datenrepräsentation:

Interne Sicht Die interne Sicht mit ihrem internen Schema spezifiziert, was tatsächlich in der Datenbank gespeichert wird und wie es gespeichert wird. Sie hat vor allem Effizienzgesichtspunkten Rechnung zu tragen und definiert die Repräsentation der Daten, Zugriffspfade und ähnliche technische Aspekte der Datenbank. Die Verwaltung der internen Sicht obliegt dem Datenbankadministrator (DBA).

Konzeptuelle Sicht Die konzeptuelle Sicht mit ihrem dazugehörigen Schema ist eine von der konkreten internen Realisierung der Datenspeicherung abstrahierende Gesamtsicht auf den verwalteten Datenbestand. Auch sie wird vom Datenbankadministrator verwaltet.

Externe Sichten Die externen Sichten sind anwendungsspezifische Sichten auf die Datenbank. Sie bieten jeweils spezielle Perspektiven des Gesamtdatenbestandes und werden von den jeweiligen Anwendungsentwicklern in Zusammenarbeit mit dem Datenbankadministrator definiert.

[1] andere Modelle sind z.B. die Senko-[Sen73] oder Strawman-Architektur[CCA82]

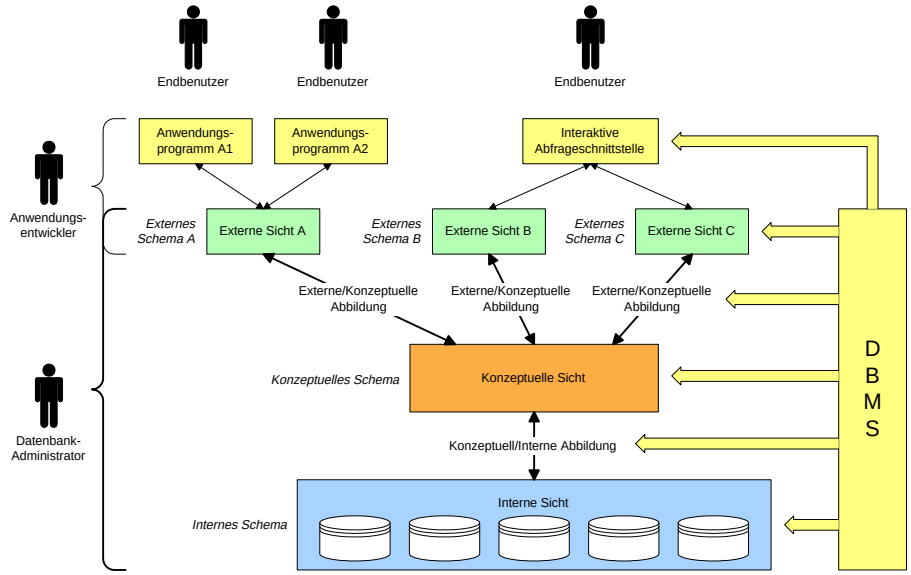

Fig. 2.2: ANSI/SPARC 3-Ebenen-Architektur

Zwischen den einzelnen Sichten existieren eindeutige Abbildungen, die bis auf die intern-konzeptuelle Abbildung vom Datenbankadministrator zu definieren sind. Die Endbenutzer agieren nur über die externen Sichten mit ihren jeweiligen Daten. Sämtliche Schemata und Abbildungen werden von dem Datenbankmanagementsystem in Form einer Meta-Datenbank[2] verwaltet.

2.2 Konkrete Datenbanksystemgenerationen

Die folgende Beschreibung beginnt bei einfachen Dateisystemdiensten und führt über verschiedene Generationen von Datenbanksystemen hin zu objektorientierten und multidimensionalen Datenbanken. Die Einteilung ist angelehnt an die Definitionen des *Third-Generation Database System Manifesto* [SRL+90] sowie [Cat94] und [Mar93]. Ziel dieses Abschnitt ist es deutlich zu machen, wie Datenbankmanagementsysteme immer wieder in ihren Funktionalitäten erweitert wurden, um den erhöhten Ansprüchen ihrer Anwendungsgebiete gerecht zu werden. Im Verlauf dieser Arbeit soll dann geklärt werden, ob dies auch für den Bereich des *Network Computing* der Fall ist.

Ursprünglich entsprach die Datenverarbeitung einer reinen Berechnung von Werten. Nur wenige Werte wurden zur Weiterverarbeitung in den Registern der Maschinen zwischengespeichert. Durch die immer komplexer werdenden Anwendungen stieg jedoch der Bedarf nach langfristiger Ablage und Wiedergewinnung von Daten kontinuierlich.

[2] in der auch alle weiteren, für den Betrieb des Datenbanksystems relevanten Informationen wie Benutzerangaben, Zugriffsrechte und Benutzungsstatistiken abgelegt werden.

2.2.1 Dateisysteme

Die einfachste Möglichkeit, Daten dauerhaft abzulegen und nach Bedarf wiederzugewinnen, waren ursprünglich die Dateisysteme der einzelnen Betriebssysteme [3]. Nach und nach boten sie dann folgende Unterstützung im Bereich der persistenten Verwaltung von Daten:

- Datensätze als Records [Hoa72] mit Elementen, die zu unterschiedlichen Typen gehörten.
- Die Möglichkeit diese Datensätze in einer Datei zu speichern, um mehr Datensätze nutzen zu können, als im Hauptspeicher Platz gefunden hätten und diese persistent ablegen zu können.
- Indizierungssysteme wie Hash-Tabellen oder B-Bäume[4].
- Dateisperren, um konkurrierende Zugriffe zu kontrollieren.

2.2.2 Datenbanksysteme der ersten Generation

Ende der 60er Jahre erschienen die ersten Datenbanksysteme im Sinne von Abschnitt 2.1 wie zum Beispiel IMS von IBM. Grundlegendster Unterschied zu den Dateisystem-basierten Lösungen war die Trennung von internem und externem Schema. Ziel war die Verwaltung der Daten unabhängig von einzelnen Anwendungen zu machen und bisher gängige Redundanzen in den anwendungsabhängigen Datendateien zu vermeiden. Diese Systeme boten hierarchische und später Netzwerkdatenmodelle (CODASYL) und fügten den Diensten nach 2.2.1 folgende hinzu:

- Datensatzidentifikatoren und Verweisattribute unter Nutzung der Identifikatoren, um effiziente hierarchische oder Netzwerkstrukturen aus den einzelnen Datensätzen zu erzeugen.
- Mehrere gleichzeitig nutzbare, indizierte Dateien, die als eine Datenbank mit internen Verweisen zu nutzen waren.
- Sicherheitsregeln, um den Zugriff auf Datensätze auf berechtigte Personen oder Programme zu beschränken.
- Transaktionen, um die Konsistenz der Datenbank auch bei mehreren, gleichzeitig arbeitenden Benutzern sicherzustellen.

Als großes Problem stellte sich jedoch der Bereich der Reorganisation der Datenbanken heraus, da das externe Schema noch zu eng mit dem internen Schema verbunden war. Das bedeutete, daß Schemaänderungen einen Großteil der Anwendungen betrafen und enormen Anpassungsaufwand erforderten.

[3] auch wenn ursprünglich noch der direkte Hardwarezugriff auf sekundäre Speicher implementiert wurde.

[4] die im Vergleich zu Hash-Tabellen auch Sortierungen sowie die Suche nach einem Wertebereich ermöglichten.

2.2.3 Datenbanksysteme der zweiten Generation

Anfang der 80er Jahre wurden die relationalen Datenbanksysteme[5] produktreif. Mit ihnen wurde die Trennung in externes und konzeptuelles Schema vollzogen. Die so gewonnene *Datenunabhängigkeit* sollte die Probleme der Systeme der ersten Generation überwinden. Diese neuen Systeme basierten auf dem relationalen Datenmodell nach [Cod70, Cod79] und boten zusätzlich zu den Diensten ihrer Vorgänger folgende Möglichkeiten:

- Eine deklarative Anfragesprache (SQL) mit Operationen, um Daten zu definieren sowie diese zu manipulieren.
- Werkzeuge für die Generierung von Masken und Listen sowie den interaktiven Zugang in Form von *Ad-Hoc*-Anfragen an die Datenbank.

Im Rahmen der Evolution dieser Systeme kamen weitere Fähigkeiten hinzu. Jetzt wurde der Schwerpunkt auf *Online Transaction Processing (OLTP)* gelegt, was dem Trend zu stark interaktiven Softwaresystemen zu verdanken war. Es galt, komplexe Integritätsregeln zentral zu verwalten und einer großen Anzahl von parallel zugreifenden Benutzern möglichst schnelle Transaktionen zur Verfügung zu stellen. Hierzu wurden folgende Technologien entwickelt:

- Integritätsbedingungen (*Integrity Constraints*), die bei jeder Operation geprüft werden und gegebenenfalls die Ausführung der Operation verhindern.
- Datenbankprozeduren (*Stored Procedures*), die die Möglichkeit bieten, innerhalb des Datenbanksystems Prozeduren oder Funktionen anzulegen und auch auszuführen.
- Datenbanktrigger, die in Abhängigkeit von Zustandsänderungen in der Datenbank Operationen oder Prozeduren ausführen.

2.2.4 Datenbanksysteme der nächsten Generation

Durch die Ausweitung der Unterstützung immer neuer Anwendungsbereiche wie Büroorganisation, Forschung und Medien haben sich auch die Anforderungen an Datenbanksysteme ausgedehnt. Sie sind nicht mehr festgelegt auf die Abspeicherung und Auswertung gleichförmiger, komplex strukturierter Daten, sondern in zunehmenden Maße auf das Zusammenspiel von Spezialsoftware zur koordinierten Präsentation und Verarbeitung heterogener Daten mittels unterschiedlicher Medien [Wet94]. Die neuen Anforderungen reichen von besonderen Datenmodellen bis hin zu technischen Funktionen wie dem *Streaming* von multimedialen Inhalten. So haben sich diverse Spezialdatenbanksysteme, aber auch Versuche universeller Lösungen in der Tradition der Datenbanksysteme der 2. Generation entwickelt.

In Anlehnung an eine Einteilung der Gartner Group ist in Abbildung 2.3 eine Übersicht dieser aktuell zu verwaltenden Informationen dargestellt. Sie reichen von multimedialen Daten über Zeitreihen und geographischen Informationen bis hin zu komplexen Dokumenten.

[5] z.B. Oracle, Informix, Ingres und DB2

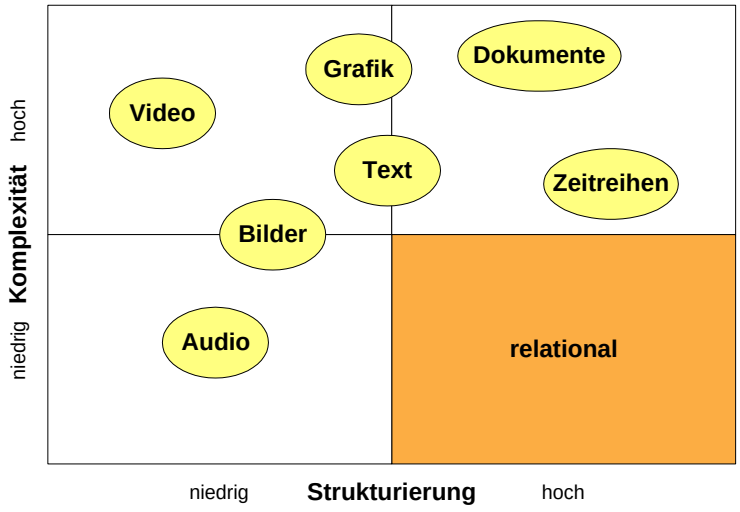

Fig. 2.3: Komplexe, unstrukturierte Datentypen

Objektdatenbanksysteme

Im Rahmen der zunehmenden Bedeutung unterschiedlicher objektorientierter Programmiersysteme und Architekturen [Boo94, GR89, Jac92, KGZ94, Mar93, Mey90, Mey94, RBP+91, WN95, Zül98] wurden Datenbanksysteme entwickelt, die das objektorientierte Programmierparadigma in Form entsprechender Datenmodelle unterstützten. Dabei ist im Bereich der objektorientierten Datenbankmanagementsysteme eine Art funktionaler Rückschritt zu beobachten. Zunächst wurde meist nur eine Persistenz-Erweiterung bestehender Entwicklungsumgebungen mit minimaler ergänzender Unterstützung angeboten. Diese Systeme basierten zum Großteil (u.a. Gemstone, O_2, Objectstore, ONTOS) auf einer Seitenserver-Strategie und unterschieden sich dadurch deutlich von den Datenbanksystemen der 2. Generation, die ja bereits einzelne Datensätze liefern konnten. Außerdem wurden kaum Sicherheitsmechanismen wie Benutzerverwaltungen mit entsprechenden Rollen- und Zugriffsrechten bereitgestellt. Diese Funktionen waren in der Regel vom Anwendungsentwickler zu programmieren und nicht deklarativ zu definieren.

Drei unterschiedliche Manifesti [ABD+89, SRL+90, DD95], lassen sich hier konstatieren, die jeweils unterschiedliche Anforderungen und Entwicklungsrichtungen objektorientierter Datenbanksysteme definieren. Außerdem gibt es ein Gremium, die *Object Database Management Group (ODMG)*, das die Standardisierung der Objektdatenbanksysteme zum Ziel hat. Allerdings hat der aktuelle Standard ODMG 2.0 [Cat97] keine besondere Relevanz : Der Anwender objektorientierter Datenbanksysteme wird wieder in die proprietäre Welt vor der Zeit von SQL zurückgeführt.

Objekt-Relationale Datenbanksysteme

Diese Systeme sind Weiterentwicklungen der relationalen Datenbanksysteme der zweiten Generation, zu ihnen gehört zum Beispiel UniSQL [Kim96]. Sie bieten Erweiterungen des Typsystems wie benutzerdefinierte Typen und echte Referenzen[6].

Online Analytical Processing (OLAP)

Für die effiziente Analyse großer Datenmengen sind spezielle Datenbanksysteme[7] entwickelt worden, die den Anforderungen in Sachen *Data Warehousing* und *Data Mining* entsprechen sollen[8]. Bei diesen geht es im Gegensatz zu OLTP-Anwendungen nicht um das operative Tagesgeschäft, sondern um die Vorbereitung von zukunftswirksamen Entscheidungen des Unternehmens und die Unterstützung bei der Ausrichtung der Unternehmensstrategie. Diese Aufgabe umfaßt das Durchdringen komplexer Informationen durch iterative Analyse von *multidimensionalen* Datenbeständen zur Informationssynthese. Aufgrund der Komplexität unternehmensweiter *Data-Warehouse*-Lösungen ist eine Einschränkung des Einsatzbereiches, etwa auf bestimmte Abteilungen, in Form der sogenannten *Data Marts* vorgesehen.

Multimedia-Datenbanksysteme

Für die Verwaltung multimedialer Daten wie Video oder Audio sind besondere Datenbanksysteme entwickelt worden, die zum Beispiel das *Streaming* der Daten aus der Datenbank ermöglichen, das für die Realisierung von Anwendungen wie *Video-on-Demand*[9] benötigt wird.

Universal Server

Als *echte* Vertreter der dritten Generation von Datenbanksystemen kann man Systeme bezeichnen, die versuchen, alle neuen Anforderungen zu erfüllen. Das Datenbanksystem *Oracle8* [Ora97h] ist ein Beispiel für ein entsprechendes System. Diese Systeme sollen Anforderungen aus den Bereichen OLTP, OLAP, multimedialer und Objekt-Relationaler Systeme genügen[10] und eine Plattform für den überwiegenden Teil der aktuellen Anforderungen bieten.

2.3 Profil

Als Ergebnis dieses Kapitels ergibt sich ein Datenbank-Profil, das die Dienste eines Datenbanksystems definiert. Basis dieses Profils ist eine Aufstellung aller

[6] nicht nur über Schlüsselbeziehungen in Form von Abfragen nutzbar, sondern auch direkt traversierbar.

[7] z.B. Applix (TM1), Intersystems (Cache) oder Express (Oracle)

[8] für weitere Informationen siehe z.B. `http://www.olapcouncil.org`.

[9] zum Beispiel für Schulungssysteme.

[10] auch wenn Oracle für den OLAP-Bereich (Express-Server) [Ora97e] und den Video-Bereich (Video-Server) eigene spezielle Server anbietet, was ein Hinweis auf mangelndes Vertrauen in das eigene Universalprodukt zu sein scheint.

Aufgaben einer Datenbank, die ich im folgenden vornehme.

2.3.1 Datenbankaufgaben

Wie in den vorherigen Abschnitten beschrieben, muß ein Datenbanksystem verläßlich eine große Menge von Daten in Mehrbenutzerumgebungen verwalten, so daß viele Benutzer gleichzeitig die gleichen Daten nutzen können. Dies muß unter Wahrung einer hohen Leistungsfähigkeit erfolgen. Außerdem muß ein DBMS unautorisierte Zugriffe verhindern und effiziente Lösungen für das Beheben von Fehlersituationen bieten [Ora97h].

Bei der Betrachtung der historischen Entwicklung von Datenbanksystemen kristallisieren sich einige grundlegende Aufgaben heraus, die ich in vier aufeinander aufbauende Gruppen unterteile[11].

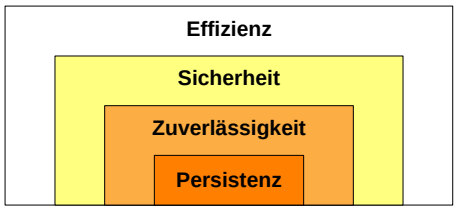

Fig. 2.4: Aufgaben eines Datenbanksystems

Persistenz

- **Langfristiger Zugriff auf große Datenmengen** Diese grundlegende Funktion sichert zu, daß die zu verwaltenden Daten über die Lebensdauer einzelner Prozesse hinaus existieren[12]. Außerdem wird erreicht, daß mehr Daten einer Anwendung zur Verfügung stehen, als im Hauptspeicher verwaltet werden könnten.
- **Metadatenverwaltung** Die Deklaration der zu verwaltenden Daten erfolgt in Form eines Datenbankschemas das über eine Datendefinitionssprache (DDL) definiert wird. Es muß ein benutzerzugänglicher Schemakatalog (*Data-Dictionary*) existieren, in dem im Rahmen des Datenbankschemas auch schemaweite Integritätsregeln definiert werden können. Ein wichtiger Aspekt in diesem Zusammenhang ist das unterstützte Datenmodell (hierarchisch, netzwerkartig, relational oder objektorientiert) bzw. das zur Verfügung stehende Typsystem. Außerdem sollten Möglichkeiten zur dynamischen Schemaevolution[13] zur Verfügung stehen.
- **Datenmanipulationsoperationen** Ein DBMS muß Operationen für die Speicherung, die Wiedergewinnung, die Aktualisierung und die Löschung

[11] Für weiterführende Informationen zu den einzelnen Aufgaben siehe unter anderem [Cod81, Dat95, EN89, LS87, STS97].
[12] Dies wird in erster Linie durch eine effiziente Speicherhierarchieverwaltung (z.B. von Sekundärspeichersystemen) erzielt.
[13] d.h. ohne den Betrieb des DBMS zu unterbrechen

von Daten anbieten[14]. Diese werden in Form einer Datenmanipulationssprache (DML) bereitgestellt. Insbesondere die Befriedigung beliebiger Anfragen an die Datenbanken ist eine der Hauptaufgaben eines Datenbanksystems.

- **Datenunabhängigkeit** Zustandsbasierte Benutzersichten ermöglichen es, ein externes Schema [15] zu definieren. Damit lassen sich fachlich motivierte Sichten auf die Datenbank definieren, der Zugriff auf Daten einschränken und außerdem Schemaänderungen vor existierenden Anwendungen verbergen. All dies bildet die Grundlage für die Integration unterschiedlicher Anwendungen über ein gemeinsames, konzeptuelles Datenbankschema. Dieses ist zudem noch unabhängig vom internen Schema. Dadurch können Optimierungen auf der internen Ebene durchgeführt werden, ohne daß die konzeptuelle Sicht beeinträchtigt wird.

Zuverlässigkeit

- **Fehlererholung** (*Recovery*) dient dem Schutz vor der physikalischen Zerstörung der Datenbank durch System-, Medien- oder Programmfehler. Aufgabe der Datensicherung ist es, die Wiederherstellung der Datenbank auch nach dieser Art von Fehlern zu ermöglichen. Wichtige Hilfsmittel für die Erfüllung dieser Aufgabe sind Transaktionen, Backup- und Restore-Werkzeuge sowie ReDo-Logs und Restart-Strategien.

- **Integritätssicherung** Die semantische Korrektheit der Datenbank wird bei jeder Operation durch Überprüfung von im Datenbankschema definierten Integritätsregeln sichergestellt. Ein wichtiges Konzept ist hier auch das der Transaktion.

- **Synchronisation** (Mehrbenutzerbetrieb) Ein DBMS erbringt seine Leistungen einem Benutzer gegenüber unabhängig von der Anzahl gleichzeitig aktiver Benutzer. Insbesondere müssen Transaktionen synchronisiert werden, um Schreibkonflikte[16] zu vermeiden.

Sicherheit

- **Authentisierung** Die Identifizierung von Benutzern oder Systemen ist insbesondere für den Bereich der Authorisierung wichtig. Aber auch für die Überwachung der Zugriffe ist die Sicherstellung der Identität der Benutzer relevant.

- **Authorisierung** Ein Aspekt von DBMS ist die Kontrolle des Zugriffs authentisierter Benutzer auf Daten und Ressourcen. Dies erfolgt im allgemeinen über Rechte auf die verschiedenen verwalteten Objekte des Sy-

[14] Diese Operationen müssen laut [LS87] generische bzw. polymorphe Funktionen sein, da die Typen der Parameter nicht starr feststehen.

[15] siehe Abbildung 2.2

[16] die grundlegendste Funktion ist hierbei das sogenannte Sperren relevanter Objekte (*Locking*), das aber zu dem Problem der möglichen Verklemmung *Deadlock* führt, dem ebenfalls über das DBMS zu begegnen ist.

stems, z.B. in Verbindung mit Rollen- oder Privilegienkonzepten, die in sogenannten Zugriffskontroll-Listen [17] definiert werden.

- **Verschlüsselung** (*Encryption*) Genauso wichtig ist es, nicht über das System kontrollierte Zugriffe zu verhindern. Um die Nutzung der Daten unabhängig vom System zu unterbinden, sollten bei der Übertragung und der Speicherung alle Daten verschlüsselt werden.
- **Überwachung** (*Auditing*) Ein DBMS sollte Möglichkeiten zur Überwachung des Zugriffs auf verwaltete Daten und Ressourcen bieten. Entsprechende Aktivitäten sollten aufgezeichnet werden, so daß später Rückschlüsse und Kontrollen möglich sind. Insbesondere sind alle Versuche, Sicherheitsmaßnahmen zu unterlaufen, zu protokollieren.

Effizienz

- **Optimierung** der Nutzung des Datenbanksystems. Dies kann in erster Linie über geschickte Strategien für die Speicher- und Wiedergewinnungsoperationen erzielt werden. Dazu zählt aber auch die Optimierung des Zugriffs einer großen Zahl von parallelen Benutzern, z.B. über gemeinsame genutzte Datenbankverbindungen oder eine Lastverteilung (*Load Balancing*) auf verschiedene Datenbankprozesse.
- **Leistungssteuerung** Ein DBMS muß Möglichkeiten zur Optimierung einzelner Funktionen entsprechend externer Anforderungen bieten[18]. Der Bereich der Leistungssteuerung umfaßt auch das komplexe Gebiet der Anfrageoptimierung. Ein Hilfsmittel sind Verarbeitungsstatistiken, die während des Betriebs aufgezeichnet und ausgewertet werden können.
- **Verteilung** Ein DBMS bietet die Möglichkeit, über Maschinengrenzen hinweg Funktionen zu nutzen und Operationen auszuführen bzw. Daten vom Datenbanksystem auf die Systeme der Benutzer (entfernter Datenzugriff) oder andere Datenbanksysteme zu übertragen (verteilte Datenbanken). Dabei ist eine Orts- und Replikationstransparenz wünschenswert. Außerdem müssen Transaktionen (siehe Zuverlässigkeit) auch in verteilten Umgebungen ihre Eigenschaften behalten.
- **Werkzeuge und Dienstschnittstellen** Ein DBMS bietet Schnittstellen zur Nutzung seiner Funktionen auf unterschiedlichen Ebenen. Dem Datenbankadministrator (DBA) müssen Werkzeuge für die Überwachung, Administration, Tuning und Betrieb des Systems zur Verfügung gestellt werden. Dem Anwendungsentwickler bieten Programmiersprachenschnittstellen und Entwicklungsumgebungen[19] Möglichkeiten für die Erstellung von Anwendungssystemen. Den Endbenutzern stehen Werkzeuge für interaktive Datenbankanfragen und -analysen zur Verfügung.

Die hier entworfene Aufstellung und Gliederung der Aufgaben eines Datenbankmanagementsystems ist unabhängig von dem unterstützten Datenmodell. Zur

[17] *access control list* (ACL)
[18] z.B. Sicherungsstrategien, physikalische Reorganisation, Zugriffspfade
[19] meist sogenannte 4GL-Systeme

Zeit werden die meisten der aufgeführten Aufgaben jedoch nur von relationalen (vgl. 2.2.3) oder universellen Datenbanksystemen (vgl. 2.2.4) erfüllt.

2.3.2 Datenbankprofil

Um die in Abschnitt 2.3.1 beschriebenen Aufgaben zu erfüllen, muß ein Datenbanksystem unterschiedlichste Dienste erbringen können. Als Ergebnis dieses Kapitels fasse ich nun diese Dienste in Form eines Datenbank-Profils zusammen:

Persistenz	Dieser Dienst bietet den langfristigen Zugriff auf große Datenmengen unter Bereitstellung von Datenzugriffsoperationen über entsprechende Dienstschnittstellen wie z.B. Abfragesprachen (*Query Languages*).
Verzeichnis	Der Verzeichnisdienst verwaltet die Datenbankobjekte wie Benutzer, Schemadefinitionen u.s.w. und bietet daher einen Zugriff auf die Metadatenverwaltung des Datenbankmanagementsystems. Dieser Dienst ist in der Regel über die Dienstschnittstellen des Persistenzdienstes nutzbar.
Sicherheit	Dieser Dienst realisiert die Identifizierung von Benutzern über Benutzername und Kennwort sowie die Regelung des Zugriffs auf Datenbankobjekte entsprechend meist deklarativer Beschreibung von Zugriffsregeln über die Dienstschnittstelle des Persistenzdienstes. Außerdem kann über diesen Dienst die Nutzung der Dienstschnittstelle des Persistenzdienstes protokolliert und überwacht werden.
Transaktion	Die Aufgaben der Fehlererholung, Integritätssicherung und Synchronisation im Rahmen der Zuverlässigkeit lassen sich durch einen Transaktionsdienst realisieren. Eine Transaktion faßt mehrere elementare Aktionen zusammen, die nach außen hin dem ACID-Prinzip genügen [HR83]. ACID steht dabei für die vier Eigenschaften einer Transaktion: Sie ist atomar[20], Konsistenz erhaltend (*consistent*)[21], isoliert[22] und dauerhaft[23].

[20] Eine Transaktion wird nur komplett durchgeführt oder komplett rückgängig gemacht.
[21] Nach einer Transaktion ist die Datenbank wieder in einem, in bezug auf definierte Integritätsregeln, korrekten Zustand.
[22] Nebenläufige Transaktionen beeinflussen sich nie gegenseitig. Dies wird durch Serialisierbarkeit von Transaktionen erreicht.
[23] Nach Abschluß einer Transaktion ist ihre Wirkung dauerhaft im Datenbanksystem gesichert.

Transport	Die Kommunikation zwischen Anwendungsprogramm und Datenbanksystem im Rahmen der Verteilungsaufgaben wird über Transportprotokolle und zugehörige interne Dienste[24] des Datenbanksystems[25] realisiert.
Programmierung	Dieser Dienst bietet die Möglichkeit, Anwendungsfunktionen in Form von Datenbankprozeduren oder Datenbanktrigger über eine Datenbankprogrammiersprache[26] zu definieren. Er erfüllt einen Teil der Aufgaben im Rahmen der Dienstschnittstellen.
Management	Für die Verwaltung des Datenbanksystems im Rahmen der Optimierungs- und Leistungssteuerungsaufgaben müssen Managementdienste[27] zur Verfügung stehen, die Informationen über den Zustand des Systems bereitstellen und die Beeinflussung einzelner Systemkomponenten erlauben. Diese Dienste erfüllen dabei Aufgaben aus dem Bereich der Werkzeuge und Dienstschnittstellen.

Tab. 2.1: Datenbank-Profil

[24] wie Lokalisierung von Kommunikationspartnern, z.B. über Oracle Names
[25] z.B. Oracle SQL*Net [Ora96c, Ora97f]
[26] z.B. Oracle PL/SQL [Ora96b, Ora97j]
[27] z.B. über den Oracle Enterprise Manager [Ora97d]

Kapitel 3

VERTEILUNG

In diesem Kapitel werde ich aktuell verfügbare Verteilungsansätze beschreiben und ihre jeweiligen Eigenschaften diskutieren. Ziel ist es, im Rahmen des Kapitels 5, im Zusammenhang mit den Aspekten aus Kapitel 4, eine optimale Kombination dieser Ansätze zu ermitteln, wobei die gemeinsamen Vorteile maximiert und die einzelnen Nachteile minimiert werden.

Zunächst werde ich dazu abstrakte Verteilungsansätze vorstellen, das technische Grundprinzip verteilter Verarbeitung beschreiben und dann eine aufgabenorientierte Betrachtungsweise einführen. Als Grundlage hierfür dient eine funktionale Gliederung, die sich an der klassischen Dreiteilung von Interaktion, Funktion und Daten orientiert. Anschließend werde ich fünf Klassen verteilter Verarbeitung vorstellen, in denen jeweils eine unterschiedliche Aufteilung der Aufgaben erfolgt. Diese dient der Klassifizierung der konkreten Architekturen, mit denen ich mich im folgenden Abschnitt eingehender auseinandersetze.

In diesem zweiten Teil des Kapitels werde ich vier aktuelle Verteilungsarchitekturen vorstellen und diskutieren. Ausgehend von Zentralrechner-basierten Architekturen über Client/Server- und Web-Architekturen bis hin zu verteilten Objektarchitekturen beschreibe ich diese einzelnen Ansätze und beurteile sie in bezug auf Kriterien wie Skalierbarkeit, Verwaltbarkeit, Zuverlässigkeit, Flexibilität, Kosten und Reifegrad. Eine Zusammenfassung der Vor- und Nachteile dieser einzelnen Ansätze bildet das Ergebnis dieses Abschnitts und dient damit als eine Grundlage für die Definition des *Network Computing* in Kapitel 5.

3.1 Abstrakte Verteilungsmodelle

Die Verteilung von Rechenleistung auf verschiedene Rechner wirkt sich auf die Architektur von Anwendungen gravierend aus. Daher müssen die einzelnen Dienste und Ressourcen sinnvoll zusammenwirken können. Für eine Beurteilung der entsprechenden Ansätze beschreibe ich im folgenden ein technisches Grundprinzip der verteilten Verarbeitung und daran anschließend fünf Klassen von Verteilungsansätzen, die auf unterschiedlichen Aufteilungen der Aufgaben innerhalb eines Anwendungssystems beruhen.

Außerdem werde ich Grundlagen erläutern und Begriffe definieren, die für das Verständnis der Diskussion verteilter Verarbeitung in dieser Arbeit von Bedeutung sind.

3.1.1 Technische Betrachtungsweise

Die technische Betrachtungsweise konzentriert sich auf das Zusammenwirken von Dienstanforderern (*Client*) und Diensterbringern (*Server*) in einer verteilten Umgebung. Grundannahme ist hier, daß eine Anwendung auf unterschiedliche Syteme verteilt werden kann, wenn sie in ihrer Gesamtheit aus einer Anzahl von dienstanfordernden und diensterbringenden Komponenten besteht.

Unter der Voraussetzung, daß die Kopplung von Dienstanforderer und Diensterbringer möglichst lose sein soll, entsteht das in Abbildung 3.1 dargestellte Drei-Ebenen-Modell.

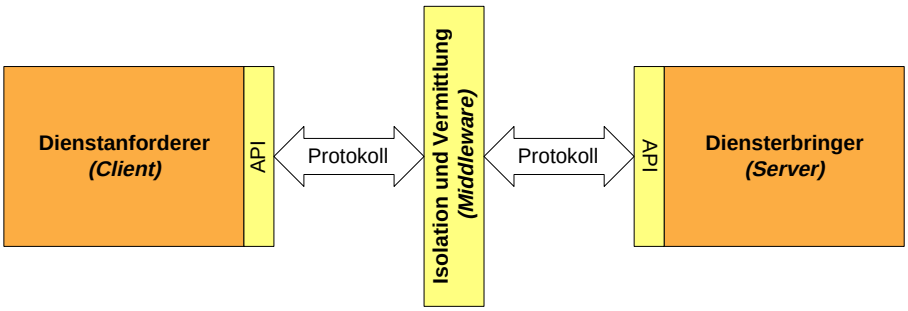

Fig. 3.1: Technisches Drei-Ebenen-Modell der Verteilung

Die einzelnen Ebenen repräsentieren dabei folgende Aufgaben:

- **Diensterbringungs- und Ressourcen-Ebene** (*Server*)
  Die Diensterbringungs- und Ressourcen-Ebene verwaltet Anwendungen (z.B. ausführbare Programme), Daten und Geräte (z.B. Drucker). Für dieses Modell ist es noch unerheblich, ob diese Dienste und Ressourcen von mehreren Anwendern gemeinsam genutzt werden oder nicht.
- **Isolations- und Vermittlungsebene** (*Middleware*[1])
  Die Isolations- und Vermittlungs-Ebene hat die Aufgabe, Lokation und Struktur der Diensterbringer und Resourcen vor der Dienstanforderungsebene zu verbergen. Erfüllt werden müssen dazu Aufgaben wie die Übertragung der Dienstanforderungen und ihrer Ergebnisse, die Protokollumsetzung, die Überwachung und Zusicherung von Sicherheitsanforderungen sowie die Lokalisierung von Diensterbringern oder Ressourcen.
- **Dienstanforderungsebene** (*Client*)
  Die Dienstanforderungsebene führt die Anwendungen aus und fordert im Verlauf der Ausführung einer Anwendung typischerweise Ressourcen an.

Die Verbindung der einzelnen Ebenen erfolgt über zwei Mechanismen:

- **Anwendungsprogramm-Schnittstellen (API)**
  Eine API definiert eine Syntax (z.B. *Embedded* SQL). Eine standardisierte API macht eine Anwendung unabhängig von der Implementierungs-Plattform und ermöglicht damit die Portierung der Anwendung.

[1] *Middleware* wird oft als der Schrägstrich zwischen Client und Server bezeichnet.

- **Protokolle**
  Protokolle definieren Semantik (z.B. HTTP). Sie ermöglichen auch in heterogenen Umgebungen die Interoperabilität zwischen Partnern.

Client und Server müssen über identische Protokolle kommunizieren, wobei aber unterschiedliche APIs verwendet werden können. Dieses Modell beschreibt ein Grundprinzip verteilter Verarbeitung aus einer technischen Perspektive.

3.1.2 Aufgabenorientierte Betrachtungsweise

In einer aufgabenorientierten Betrachtungsweise lassen sich ebenfalls drei logische Ebenen erkennen [Ber92, Zül98]. Ein entsprechendes Modell ist in Abbildung 3.2 dargestellt.

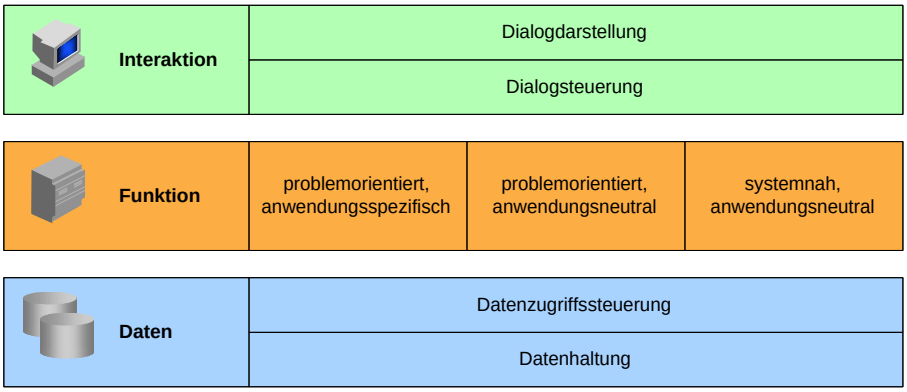

Fig. 3.2: Aufgabenorientiertes Ebenen-Modell der Verteilung

Die Komponenten dieser Ebenen erfüllen die folgenden Aufgaben:

- **Interaktion**
  Die Interaktions-Ebene realisiert die Benutzerschnittstelle zur Anwendung. Sie ist für die grafische Darstellung der Anwendung verantwortlich und kann in zwei Bereiche gegliedert werden:

  Die **Dialogdarstellung** übernimmt die Steuerung und Ausgabe des grafischen Layouts der Benutzerschnittstelle, also zum Beispiel die Lage und den Typ der Eingabefelder oder die Schriftarten.

  Die **Dialogsteuerung** verwaltet die Inhalte der Benutzerschnittstelle und die Interaktion mit dem Benutzer. Dabei koordiniert und synchronisiert sie die Anwenderaktionen, den Kontrollfluß der Anwendung und die Dialogoberfläche. Dies beinhaltet insbesondere die Entgegennahme von Dateneingaben, Plausibilitätsprüfungen sowie die Erzeugung von Dialogzuständen, insbesondere aufgrund dynamischer Ereignisse wie etwa vorherige Aktionen des Benutzers.

- **Funktion**
  Die Funktions-Ebene übernimmt die Vermittlung zwischen der Benutzerschnittstelle und der Datenhaltung bei gleichzeitiger Erfüllung des eigentlichen Anwendungszwecks.

  Die **problemorientierten, anwendungsspezifischen** Funktionen erfüllen alle anwendungsabhängigen Aufgaben wie den anwendungsspezifischen Kontrollfluß oder anwendungsabhängige Berechnungen. Auch Plausibilitätsprüfungen, die nicht über die Dialogsteuerung abgedeckt werden können (z.B. maskenübergreifend), werden hier durchgeführt.

  Die **problemorientierten, anwendungsunabhängigen** Funktionen führen problemorientierte Aufgaben aus, die von mehreren Anwendungen genutzt werden könnten (z.B. Datumsberechnungen, Steuerberechnungen oder Adreßprüfungen).

  Die **systemnahen, anwendungsunabhängigen** Funktionen bieten Dienste im Zusammenhang mit der Einbettung der Anwendung in die Systemumgebung an (z.B. die Integration in eine bestehende Systemmanagementumgebung oder die Bereitstellung grundlegender Druckdienste).

- **Daten**
  Die Daten-Ebene bietet den Zugriff auf persistente Speicherobjekte und repräsentiert die Daten der Anwendung.

  Die **Datenzugriffssteuerung** soll das Datenmodell und die Eigenheiten des jeweiligen Datenhaltungssystems soweit wie möglich vor der Anwendung verbergen. Dazu nimmt sie die logischen Zugriffsanforderungen der Anwendung entgegen und konvertiert sie in Zugriffsbefehle des Datenhaltungssystems.

  Die **Datenhaltung** bietet grundlegende Datenbankfunktionen und übernimmt die eigentliche persistente Verwaltung der Daten der Anwendung.

Je nach Aufteilung der Funktionalitäten lassen sich nun fünf Klassen unterscheiden, die in Abbildung 3.3 dargestellt sind:

Entfernte Benutzerschnittstelle (*Remote Interface*) Die erste Klasse der verteilten Verarbeitung ist die entfernte Benutzerschnittstelle. Sie basiert auf zentralen Rechnern (Server) und einfachen Terminals (Client). Die komplette Logik der Anwendung inklusive der Dialogsteuerung liegt auf dem Server, auf dem Client wird nur die Dialogdarstellung realisiert.

Verteilte Benutzerschnittstelle (*Distributed Interface*) Die nächste Klasse basiert auf der teilweisen Verlagerung von Funktionen der Dialogsteuerung auf die Client-Systeme.

Entfernter Datenzugriff (*Remote Data Access* (RDA)) In der dritten Klasse kehrt sich die Gewichtung bei der Verarbeitung um. Die Client-Systeme führen die Anwendungen lokal aus und nutzen Server-Systeme lediglich für Datenzugriffe und zum Zugriff auf periphere Geräte.

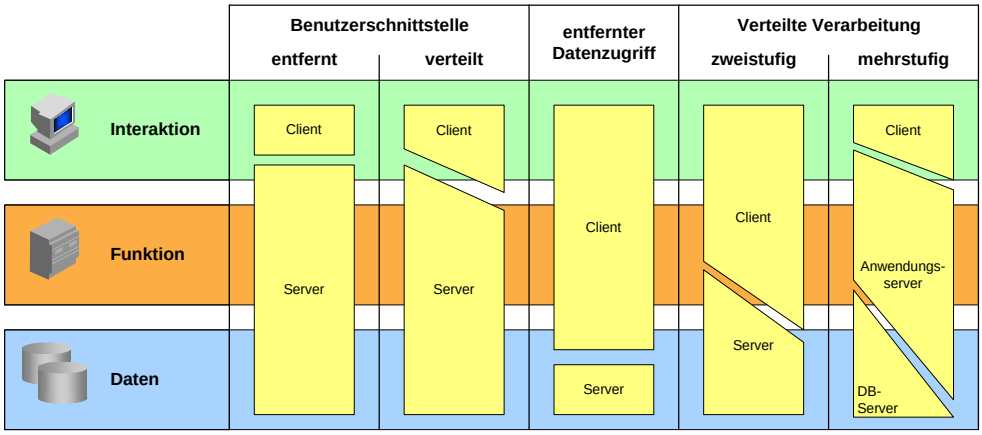

Fig. 3.3: Klassifizierung verteilter Verarbeitung

Zweistufige, verteilte Verarbeitung (*2-tier Distributed Processing*) Diese Klasse beschreibt die Verlagerung von Funktionen aus den Bereichen der Funktions-Ebene und der Datenzugriffssteuerung vom Client auf den zentralen Server.

Mehrstufige, verteilte Verarbeitung (*n-tier Distributed Processing*) Die letzte Klasse führt weitere Verarbeitungsplätze zwischen dem Client und den Servern ein. Bestimmte Funktionen werden auf speziellen Servern realisiert und zur Verfügung gestellt.

Diese hier aufgestellte Klassifizierung ermöglicht eine Einteilung von beliebigen Verteilungsansätzen in eine der fünf Klassen. Im nächsten Abschnitt werde ich nun konkrete Architekturen beschreiben und sie den Verteilungsklassen zuordnen.

3.2 Konkrete Verteilungsansätze

In diesem Abschnitt beschreibe ich vier unterschiedliche, konkrete Verteilungsarchitekturen mit ihren jeweiligen Vor- und Nachteilen. Die Reihenfolge, in der ich die einzelnen Architekturen beschreibe, entspricht der Reihenfolge ihrer Entwicklung und ihres Einsatzes im Rahmen betrieblicher Informationssysteme.

3.2.1 Zentralrechner

Die ersten großen kommerziellen Systeme wurden von Herstellern wie z.B. IBM entwickelt und werden heute unter dem Begriff *Mainframe*[2] zusammengefaßt. Sie repräsentieren auch heute noch den größten Teil der unternehmenskritischen Anwendungen.

[2] auch Zentralrechner, Host oder Großrechner genannt.

Die Anwender dieser Systeme erwarben Hardware, Software und Dienstleistungen für ihr Rechenzentrum von meist nur einem Hersteller. Die Kombination von Lösungen unterschiedlicher Hersteller war aufgrund nicht offengelegter Schnittstellen kaum möglich.

In den Rechenzentren dieser Struktur werden, wie in Abbildung 3.4 dargestellt, alle Daten auf dem Mainframe gespeichert. Alle Anwendungen werden auf dem Großrechner ausgeführt, selbst das Benutzerinterface wird vom Host gesteuert. Die Benutzer interagieren über Terminals ohne eigene Funktion mit dem System. Dies ermöglicht eine zentrale Verwaltung des Gesamtsystems, begrenzt jedoch auch die Funktionalität der Benutzerschnittstelle.

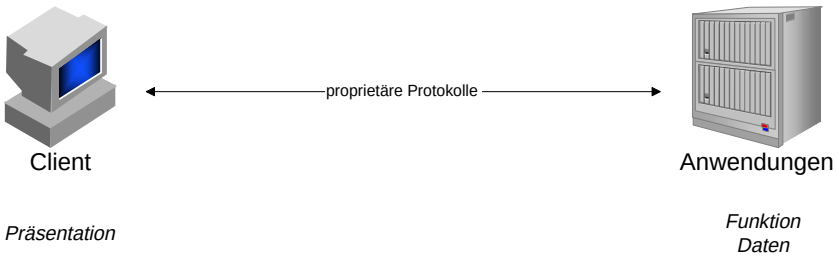

Fig. 3.4: Struktur von Zentralrechner-Architekturen

Aufgrund des bereits langen Einsatzes dieser Art von Strukturen gibt es umfangreiche Erfahrungen und ausgereifte Techniken, durch die eine hohe Skalierbarkeit, Verfügbarkeit und Verwaltbarkeit dieser Systeme erreicht werden kann. Die Sicherheit und Integrität der verwalteten Daten ist aufgrund des monolithischen Gesamtkonzeptes dieses Ansatzes relativ einfach herzustellen.

Allerdings tendieren diese Systeme dazu, teuer und unflexibel zu sein, da in ihnen große Softwaresysteme laufen, die oft ohne ausreichende Werkzeugunterstützung entstanden und kaum ohne großen Aufwand anpaßbar sind[3]. Interoperabilität ist bei diesen herstellerspezifischen Systemen nie ein Thema gewesen.

Diese Art der Verarbeitung ist der Klasse der **entfernten Benutzerschnittstelle** zuzuordnen.

3.2.2 Client/Server

Ende der 80er Jahre wurden sogenannte *Personal Computer* (PC) eingeführt. In erster Linie waren diese als Einplatzsysteme konzipiert und sollten ein vom Rechenzentrum unabhängiges Arbeiten mit *persönlichen* Anwendungen wie Textverarbeitung, Tabellenkalkulation oder Präsentationen ermöglichen. Die relevanten Betriebssysteme der ersten Stunde waren CP/M und MS-DOS, später dann MS-Windows. Ein großer Vorteil dieser Systeme war die Möglichkeit, graphische Benutzeroberflächen[4] einsetzen zu können und damit die Bedienerfreundlichkeit zu erhöhen.

[3] Bestes Beispiel sind die aktuellen Probleme bei der Euro- und Jahr-2000-Umstellung.
[4] *Graphical User Interface* (GUI)

Kurz nach der Einführung der PCs entstand indes der Bedarf, einzelne PCs an existierende Großrechnersysteme anzuschließen, um die dort vorhandenen Daten in Anwendungen wie Tabellenkalkulationen oder Präsentationen weiterzuverwenden. Hier wurden erste Nachteile der Dezentralisierung der Verarbeitung deutlich: Die Anbindung geschah in erster Linie über Terminalemulationen[5], d.h. daß diese PCs nun als Terminal im Rahmen **verteilter Interaktion** genutzt wurden.

Später wurden einzelne PCs zu sogenannten PC-Netzwerken zusammengeschlossen, um gemeinsam Sekundärspeicher- und Drucksysteme nutzen zu können. Diese Funktionen wurden von ausgezeichneten PCs, sogenannten File- und Printservern bereitgestellt. Diese Art der Verarbeitung entspricht dem **entfernten Datenzugriff**.

Der nächste Schritt war die Entwicklung spezieller Server, die weitere Dienste anboten. Zunächst wurden diese in zweistufigen Architekturen (Client/Server der ersten Generation), später in beliebigen Kombinationen (Client/Server der zweiten Generation) eingesetzt. Durch diese Client/Server-Architekturen wurde auch ein neuer Typ von integrierenden Softwareprodukten geboren, die sogenannte *Middleware*, welche die Vermittlungs- und Isolationsaufgaben entsprechend Abschnitt 3.1.1 übernommen hat[6]. Als Client/Server der dritten Generation könnte man die Architekturen verteiler Objekte bezeichnen, die in Abschnitt 3.2.4 näher beschrieben werden. Ähnlich wie bei den Datenbanksystemgenerationen aus Kapitel 2 ist ein Wechsel von einer Generation zu einer neueren Generation mit enormen Eingriffen in die Anwendungssysteme verbunden.

Client/Server-Architekturen der ersten Generation

Typische Client/Server-Anwendungen der ersten Generation konzentrieren Präsentation und Anwendungslogik auf den Arbeitsstationen[7], die Datenbanksysteme befinden sich auf einem zentralen Server. Der Begriff Client bezieht sich in diesem Zusammenhang auf die Interaktions- und Anwendungslogik, wird aber auch für die Arbeitsstation als Ganzes verwendet. Als Server wird neben dem Datenbanksystem auch das zugehörige Hardware-System bezeichnet. Eine entsprechende Strukur ist in Abbildung 3.5 dargestellt.

Die klassische Client-Server-Architektur kennt in erster Linie nur Datenbankclients und Datenbankserver. Diese übernehmen im Gegensatz zu Fileservern nicht die Verwaltung der Datenbankdateien, sondern der Daten selbst. Ein Client kann Dienste auf Datensatzebene und nicht auf Dateiebene in Anspruch nehmen. Über das Netzwerk werden hier nur die einzelnen Datensätze und nicht die kompletten Datendateien übermittelt. Hersteller dieser Systeme sind u.a. Oracle, Informix, Ingres, IBM, Sybase und Microsoft. Eine Weiterentwicklung dieser Server führt zu sogenannten *Transaktionsservern*: Hier schicken die Clients keine DML-Kommandos an den Server, sondern rufen sogenannte *Stored Procedures* auf, die auf dem Server ausgeführt werden. Diese Art der Verarbeitung ist der Klasse der **zweistufigen, verteilten Verarbeitung** zuzurechnen.

[5] vgl. etwa PC-Support für AS/400-Kommunikation

[6] Middleware ist für mich der komplexeste und am wenigsten standardisierte Teil des Client/Server-Paradigmas. Einzelne Middlewareprodukte überlappen sich meist in ihren Diensten, was sogar soweit führt, daß Middleware für Middleware geschaffen wird.

[7] Die daher oft auch als *Fat Client* bezeichnet werden.

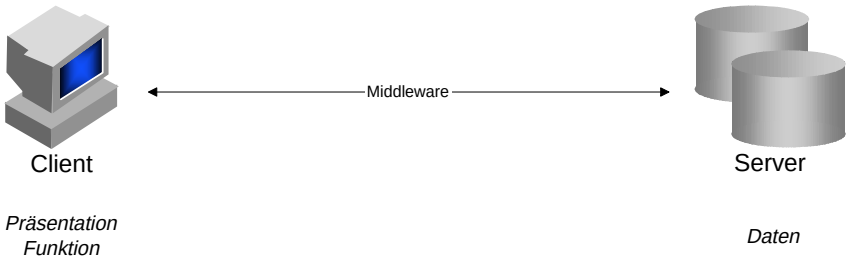

Fig. 3.5: Struktur von C/S-Architekturen der 1. Generation

Die Vorteile dieser Client/Server-Architekturen liegen in dem hohen Reifegrad der verwendeten Technologien. Vorhandenes Know-How der Entwickler beim Design und der Implementierung, Know How der Administratoren beim Einsatz der Technologie, ausgereifte Werkzeuge, Sprachen und Middleware sind wesentliche Vorteile dieses Architekturmodells.

Die Nachteile werden schnell in den hohen Anforderungen an Hard- und Software deutlich. Um Performance-Spitzen abzufangen, sind die Servermaschinen für den Normalbetrieb meist überdimensioniert ausgelegt, da Skalierung nur über Hardwareleistung des Servers möglich ist. Die Gesamtsysteme sind aufgrund des Zusammenspiels unterschiedlichster Technologien sehr komplex und daher meist unflexibel.

Client/Server-Architekturen der zweiten Generation

Nach den zweistufigen Client/Server-Architekturen entstehen nun Mehr-Schichten-Architekturen. Die konsequente Trennung von Interaktion, Funktion und Daten führt zu weiter verteilbaren Systemen. Dabei wird die erste Ebene der zweistufigen Client/Server-Architekturen noch einmal aufgeteilt[8] und mindestens eine weitere Schicht definiert, in der anwendungsfachliche Dienste zusätzlich zu den reinen Datenbankdiensten zur Verfügung gestellt werden. Performanzsteigerungen durch den Einsatz speziell ausgelegter Anwendungsserver und automatischer Lastverteilung sowie die leichte Austauschbarkeit von Komponenten oder ganzen Schichten führen hier zu flexibleren Systemen. Diese Art der Verarbeitung gehört der Klasse der **mehrstufigen verteilten Verarbeitung** an. In Abbildung 3.6 ist eine derartige Struktur dargestellt.

Ein Beispiel ist die Trennung von Dialogdarstellung und Dialogsteuerung einer Anwendung in Form der **entfernten Benutzerschnittstelle**. Dieses Konzept soll unter dem Namen *Thin-Client/Server-Computing* [Gam99, Kan98] mit der Einführung von sogenannten WinTerminals und Produkten wie CITRIX WinFrame[9] die Vorteile der Client/Server-Systeme mit den Vorteilen der Zen-

[8] Interessant zu beobachten ist hier, daß eine Parallele zur Entwicklung der Datenbanksysteme zu erkennen ist. Auch hier wurde die anwendungsfachliche Ebene bei Systemen der zweiten Generation in zwei neue Ebenen (extern und konzeptuell) aufgeteilt, um eine höhere Flexibilität zu erreichen.

[9] vgl. `http://www.citrix.com`

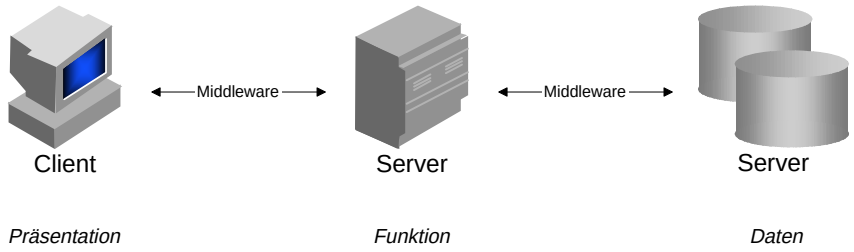

Fig. 3.6: Struktur von C/S-Architekturen der 2. Generation

tralrechnerwelt verbinden und somit die Verwaltungskosten der Arbeitsplätze senken.

Der Hauptvorteil von mehrstufigen Architekturen liegt darin, daß die Integration unterschiedlicher Anwendungen auf der Ebene der Geschäftslogik und nicht mehr auf der Ebene der Daten erfolgen kann. Dadurch wird die Wiederverwendbarkeit bereits realisierter Funktionalitäten erhöht und die Sicherheit durch die Verlagerung der Geschäftsprozesse auf zentral kontrollierbare Server verbessert.

Nachteil dieser neuen Technologie ist in erster Linie die erhöhte Komplexität durch die notwendige Partitionierung von Funktionen auf die verteilten Systeme. Auch ist die Unterstützung durch entsprechende Werkzeuge noch gering. Um die unternehmensweite Wiederverwendung von bereits realisierten Funktionen zu ermöglichen, ist ein komplexes Design in der Tradition der Unternehmensdatenmodelle [Sch90] erforderlich. Dieses Unterfangen wird durch zusätzliche Berücksichtigung der Geschäftsprozesse nicht gerade vereinfacht. Mit dem komplexen Design und der stärkeren Verteilung der Systemkomponenten erhöhen sich daher auch die Komplexität der Administration sowie die dazugehörigen Kosten.

3.2.3 World Wide Web

Obwohl Internet-basierte Systeme eigentlich den Client/Server-Systemen zuzuordnen sind, führe ich sie hier getrennt auf, da sie für diese Arbeit von besonderer Bedeutung sind.

Anwendungen werden hier nicht auf den Arbeitsstationen installiert, sondern werden zur Ausführungszeit über das Netzwerk geladen. Als Konsequenz müssen die Arbeitsstationen nur noch die Ausführungsumgebung für die Anwendungen bereitstellen. Sekundärspeicher sind nicht mehr unbedingt erforderlich.

Der Server übernimmt hier in erster Linie Kommunikationsdienstleistungen und bietet quasi weltweite Verfügbarkeit der Anwendungen im Zusammenhang mit minimalen Anforderungen an die Fähigkeiten der Arbeitsstationen (*Browser*). Diese Dienste basieren auf einer universellen Methode zur netzweiten Adressierung von Ressourcen, dem *Universal Resource Locator* (URL), einem einfachen, genormten Kommunikationsprotokoll namens *Hypertext Transfer Protocol*

(HTTP) sowie einer zugehörigen Dokumentenbeschreibungssprache, der *Hypertext Markup Language* (HTML).

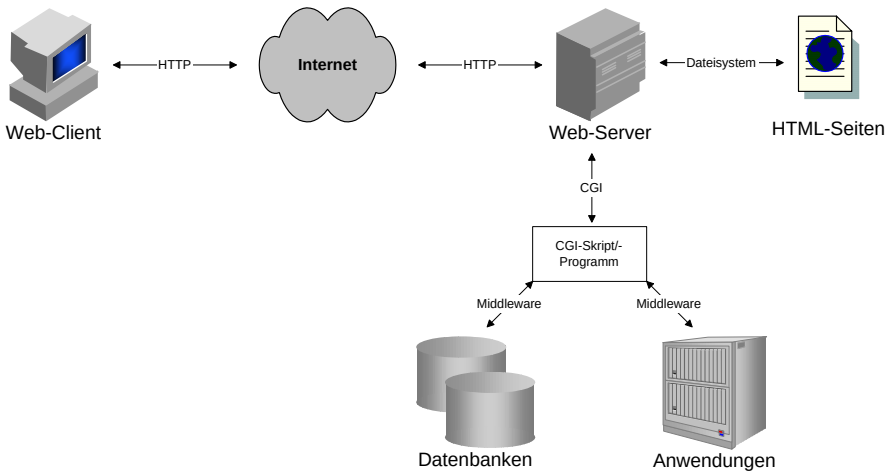

Fig. 3.7: Struktur von WWW-Anwendungen

Zur Erweiterung der Funktionen des WWW-Servers wurde das *Common Gateway Interface*[10] (CGI) definiert. Es ist ein Standard für die Übergabe einer Anfrage mit ihren Parametern vom Webserver an ein externes Anwendungsprogramm. Eine derartige Struktur ist in Abbildung 3.7 dargestellt.

Da die Browser einfache Funktionen ausführen können, ist diese einfachste Art der Verarbeitung[11] der Klasse der **verteilten Benuzterschnittstelle** zuzuordnen. Im Falle des Einsatzes von Java-Applets auf der Seite des Browsers handelt es sich um eine Einteilung in die **zweistufige, verteilte Verarbeitung**. Werden über den Internetserver Anwendungen anderer Server publiziert, so muß eine Zuordnung zur **mehrstufigen verteilten Verarbeitung** erfolgen.

Die größten Vorteile dieser Systeme sind die ausgereiften, standardisierten Schnittstellen und Protokolle. HTML schafft Portabilität, während HTTP als standardisiertes Protokoll Interoberabilität garantiert. Außerdem bieten sie eine zentrale Verwaltung der Anwendungen auf den Servern und damit geringe Administrationskosten.

Als nachteilig erweist sich die Tatsache, daß die meisten Systeme keine vernünftige Lastverteilung und damit kaum Skalierbarkeit bieten. Außerdem ist HTTP ein zustandsloses Protokoll, was die Überwachung von Transaktionen durch das Fehlen eines Sessionkonzeptes stark erschwert. Leistungsfähige Sicherheitskonzepte[12] sind erst vor kurzem entstanden. Zudem müssen Entwickler über Kenntnisse in einer Vielzahl von Sprachen wie HTML oder Skriptsprachen wie Perl oder JavaScript verfügen.

[10] siehe Abschnitt 5.2.3.
[11] das reine Publizieren von statischen HTML-Seiten sowie die Ausführung einfacher CGI-Anwendungen.
[12] Auf diese Konzepte werde ich in Kapitel 5 näher eingehen.

3.2.4 Verteilte Objektsysteme

In diesen Architekturen kommunizieren verteilte Objekte über ein gemeinsames Kommunikationsmedium, einen *Object Request Broker* (ORB)[13]. Dies ermöglicht es Entwicklern und Anwendern, Objekte unabhängig von ihrer Lokation, Implementierung und Programmiersprache zu nutzen. Diese Unabhängigkeit wird durch die Verwendung einer speziellen *Interface Definition Language* (IDL) für die Spezifikation der Schnittstellen erreicht. Damit können Objekte dynamisch auf jedem Server oder Arbeitsplatzrechner innerhalb der Infrastruktur plaziert werden, je nach Bedarf. Diese Art der Verarbeitung ist die maximale Fassung der **mehrstufigen, verteilten Verarbeitung**.

Der aktuelle Standard für ORBs ist die *Common Object Request Broker Architecture* (CORBA 2.2) [OMG98c][14] der *Object Management Group* (OMG). In Abbildung 3.8 ist die Architektur eines CORBA-ORBs dargestellt.

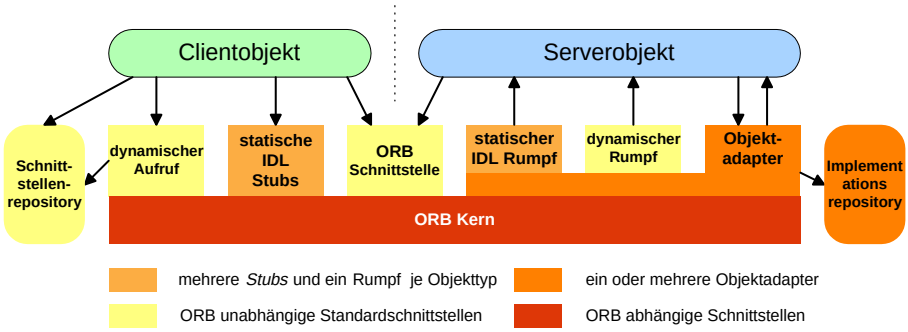

Fig. 3.8: Struktur eines CORBA 2 ORB

Ein Client hat die Möglichkeit über einen *IDL Stub*[15] oder über die dynamische Aufrufschnittstelle[16] einen Dienst eines anderen Objektes in Anspruch zu nehmen. Die dynamische Aufrufschnittstelle nutzt dabei Metainformationen über alle registrierten Objektschnittstellen, die in einem zentralen Schnittstellenrepository zur Verfügung stehen, um geignete Objekte zu finden.

Für den Server unterscheiden sich diese beiden Aufrufarten nicht. In beiden Fällen nutzt der ORB einen Objektadapter, um die Parameter des Aufrufs und die Kontrolle an eine Objektimplementation zu übergeben. Dies kann über einen statischen IDL Rumpf (*skeleton*) geschehen oder wieder über eine dynamische

[13] verfügbare Produkte sind zum Beispiel der VisiBroker von Visigenic/Inprise (verwendet von Netscape und Oracle) oder ORBIX von IONA.
[14] Diese Archtektur nennt sich *common*, weil sie die unterschiedlichen Ansätze der beiden Hauptbeiträge (Digital mit der dynamischen Schnittstelle und HP/SUN mit der statischen Schnittstelle) integriert.
[15] Diese *Stubs* bieten die statischen Schnittstellen zu den Objektdiensten. Sie agieren wie ein Stellvertreterobjekt (*proxy*) für ein enferntes Objekt.
[16] *Dynamic Invocation Interface* (DII)

Schnittstelle[17]. Dabei bietet der Objektadapter die Laufzeitumgebung für die Instantiierung und Ausführung der Serverobjekte, die die Implementation der Dienste realisieren. Er registriert alle unterstützten Klassen und deren Instanzen im Implementationsrepository.

Zusätzlich stehen den Clienten und den Objektimplementationen einige ORB-Infrastrukturdienste [18] über die ORB-Schnittstelle zur Verfügung.

Für die Kommunikation zwischen verteilten ORBs definiert die OMG ein abstraktes *General Inter-ORB Protocol* (GIOP). Es beschreibt eine Standard-Transfersyntax[19] und eine Menge von Nachrichtenformaten. Das *Internet Inter-ORB Protocol* (IIOP) ist die konkrete Realisierung des GIOP auf der Basis von TCP/IP. Seine Unterstützung ist eine notwendige Eigenschaft für die sogenannte CORBA 2.0 Kompatibilität eines ORBs, welche die Interoperabilität entsprechender ORBs sicherstellen soll. Dadurch kann ein Clientobjekt Dienste eines Serverobjektes eines entfernten ORBs in Anspruch nehmen.

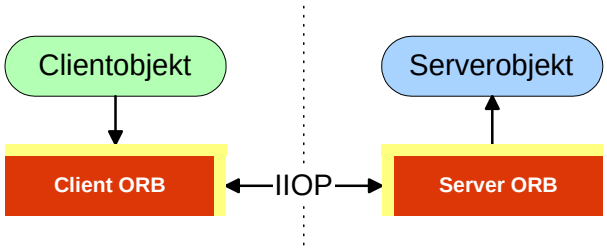

Fig. 3.9: Kommunikation zwischen Client- und Serverobjekt über IIOP

In Abbildung 3.9 wird gezeigt, wie ein Clientobjekt über seinen ORB Dienste eines Serverobjekts aufruft. Dabei befindet sich das Serverobjekt in einem anderen Adreßraum als das Clientobjekt. Der Client-ORB kommuniziert dafür mit dem Server-ORB über IIOP.

Ein Aspekt verteilter Objekte ist das Konzept einer Softwarekomponente. Eine Komponente ist ein wiederverwendbares, eigenständiges Stück Software, das unabhängig von bestimmten Anwendungen ist [OHE96] und seine Dienste über genormte Schnittstellen zur Verfügung stellt. Damit sind die verteilten Objekte per Definition schon Komponenten. Komponenten müssen generelle Abstraktionen darstellen, damit sie in unterschiedlichen Kontexten eingesetzt werden können und damit das primäre Ziel der Wiederverwendbarkeit realisiert werden kann [Jac92].

Basis des Einsatzes von Komponenten ist ein entsprechendes Komponentenmodell[20] wie die *JavaBeans* [Sun97a] im Rahmen der Programmiersprache Java [GJS96, Fla98]. Java Beans sind in erster Linie visuelle Komponenten. Sie können über eine Eigenschaftentabelle und Anpassungsmethoden an konkrete Anforderungen angepaßt werden.

[17] *Dynamic Skeleton Interface* (DSI)
[18] *Common Objects Services* (COS)
[19] *Common Data Representation* (CDR)
[20] siehe Abschnitt 5.2.3.

Ein weiteres, neueres Komponentenmodell sind die *Enterprise JavaBeans* (EJB) [Sun98a], die ein Server-Komponentenmodell für Java definieren. EJBs sind spezialisierte, nicht-visuelle JavaBeans, die auf einem Server in einer transaktionsorientierten Umgebung laufen. Über ein Mapping [Sun98b] kann eine Integration der EJBs in CORBA-Umgebungen stattfinden, so daß CORBA-Clienten EJBs in einem CORBA-basierten EJB-Server nutzen und EJBs und CORBA-Objekte an gemeinsamen Transaktionen teilhaben können.

Ultimatives Ziel ist die Entwicklung von sogenannten Geschäftsobjekten (*Business Objects*), die über Parameter, also ohne Zugang zum Quellcode der Komponente, an spezifische Anforderungen angepaßt werden können. Ein Geschäftsobjekt beinhaltet die Repräsentation eines fachliches Konzepts der realen Welt, des Umgangs mit diesem Konzept und seiner Beziehungen zu anderen Konzepten [OMG97a]. Das Konzept der Geschäftsobjekte soll in die *Object Management Architecture* (OMA) [OMG97c] der OMG aufgenommen werden.

Über derartige, interagierende Komponenten Anwendungen zu konzipieren, stellt eine Manifestation des in [Weg98] beschriebenen Paradigmenwechsels von Algorithmen zu Interaktionen dar.

Allerdings ist diese Technologie noch relativ jung, die Erfahrungen einzelner Pilotprojekte fließen erst jetzt in die entsprechenden Spezifikationen mit ein.

3.3 Bewertung

Nach der Vorstellung der zur Zeit relevanten Verteilungsansätze und der Einordung der einzelnen Architekturen in die fünf Verteilungsklassen nach Abschnitt 3.1.2 werde ich nun die zum Teil bereits erwähnten Vor- und Nachteile der Ansätze anhand von zwölf Kriterien zusammenfassend betrachten. Diese Bewertung dient dann als Grundlage für die Auswahl von Technologien und Konzepten im Rahmen des Referenzmodells in Kapitel 5.

3.3.1 Kriterien

Eine mögliche Perspektive zur Untersuchung der Herausforderungen verteilter Verarbeitung ist die Betrachtung qualitativer Aspekte der Verteilung. Ich wähle hier folgende Aspekte, die der Bewertung der konkreten Verteilungsansätze im folgenden Abschnitt dienen:

- **Ausgereiftheit** definiert den Reifegrad der Verteilungsansätze in bezug auf Erfahrung der Entwickler und Stabilität der eingesetzten Technologien.
- **Bedienerfreundlichkeit** ist ein Maß für den Komfort und die Flexibilität der Benutzeroberfläche, wobei z.B. graphische Benutzeroberflächen als komfortabler gelten als textorientierte Systeme.
- **Flexibilität** beschreibt die Fähigkeit zur Anpassung der Anwendungen an sich verändernde Anforderungen oder Rahmenbedingungen.

- **Integrität** ist auch im Bereich verteilter Verarbeitung ein wichtiger Aspekt. Hierzu zählt die Integrität der verarbeiteten Daten ebenso wie die Integrität der übermittelten Nachrichten. So können z.B. Unterschiede in der Benennung von Objekten durch verschiedene verteilte Systemkomponenten im Verlust von Ressourcen münden. Aber auch die Unterstützung verteilter Transaktionen ist ein Maß für die Sicherung der Integrität in verteilten Umgebungen.

- **Interoperabilität** zwischen verschiedenen Technologien ist aufgrund der natürlichen Heterogenität verteilter Umgebungen ein wichtiger Aspekt. Es ist eine möglichst einfache Zusammenarbeit unterschiedlichster Systeme anzustreben.

- **Kosten**-Aspekte sind auch bei der Beurteilung von Verteilungskonzepten von Bedeutung. Hier ist, wie überall, eine Minimierung von Kosten der Einrichtung, aber auch im Betrieb der Systeme anzustreben.

- **Portabilität** ist hier in erster Linie im Hinblick auf die Clienten der verteilten Umgebung zu sehen. Ziel ist es, eine möglichst große Anzahl von unterschiedlichen Clienten in bezug auf Betriebssysteme und Hardware zu unterstützen. Aber auch die Möglichkeit der Verwendung unterschiedlicher Server-Systeme ist wünschenswert.

- **Produktivität** ist ein Maß für die Effizienz der Anwendungsentwickler bei der Entwicklung von Anwendungen im Rahmen der unterschiedlichen Verteilungsansätze. Hilfreich sind hier Unterstützungen wie integrierte Entwicklungsumgebungen[21] oder existierende Softwarekomponenten.

- **Sicherheit** ist in verteilten Umgebungen stets ein gefährdeter Bereich. Daher sollte die Übertragung von Informationen ebenso sichergestellt werden wie die zuverlässige Identifizierung der Kommunikationspartner - und zwar auch über unterschiedliche Technologien hinweg.

- **Skalierbarkeit** ist ein Maß für die Fähigkeit, eine fast beliebige Anzahl von Clienten im Rahmen der verteilten Umgebung bedienen zu können. Dies betrifft insbesondere die Fähigkeit, eine transparente Lastverteilung auf mehrere Server vornehmen zu können.

- **Verwaltbarkeit** ist in einer verteilten Umgebung ein besonders komplexer Aspekt. Generell sollten Anwendungskomponenten von einer zentralen Stelle installierbar und administrierbar sein. Durch die Konzentration von Funktionen auf zentralen Servern kann die Komplexität der Verwaltung deutlich verringert werden.

- **Zuverlässigkeit** beschreibt das Maß, in dem die Server und damit die bereit gestellten Dienste innerhalb der verteilten Umgebung dauerhaft zur Verfügung stehen. Je länger und häufiger die Ausfallzeiten einzelner Server sind, desto geringer ist die Verfügbarkeit. Die Verfügbarkeit kann durch Redundanz gefährdeter Komponenten erhöht werden.

[21] *Integrated Development Environment* (IDE)

3.3.2 Übersicht

Die folgende Tabelle 3.1 bietet eine Übersicht über die Zuordnung der einzelnen konkreten Verteilungsansätze zu den abstrakten Verteilungsansätzen aus Abschnitt 3.1.2.

	Zentral-rechner	Client / Server	WWW	Verteilte Objekte
entfernte Benutzerschnittstelle	X	X		
verteilte Benutzerschnittstelle		X	X	
entfernter Datenzugriff		X		
2-stufig verteilte Verarbeitung		X	X	
n-stufig verteilte Verarbeitung		X	X	X

Tab. 3.1: Zuordnung zu den abstrakten Verteilungsansätzen

In Tabelle 3.2 werden nun die, in Abschnitt 3.2 beschriebenen, einzelnen Vor- und Nachteile der jeweiligen Architekturen entsprechend den Kriterien des Abschnitts 3.3.1 zusammengefaßt dargestellt. Dabei verwende ich die Symbole + für eine positive, **O** für eine mittelmäßige sowie - für eine negative Einschätzung des Ansatzes in bezug auf das jeweilige Kriterium.

	Zentral-rechner	Client / Server	WWW	Verteilte Objekte
Ausgereiftheit	+	+	+	-
Bedienerfreundlichkeit	-	+	+	+
Flexibilität	-	O	O	+
Integrität	+	+	-	O
Interoperabilität	-	+	+	+
Kosten	-	-	+	O
Portabilität	-	O	+	O
Produktivität	O	+	O	+
Sicherheit	+	+	O	O
Skalierbarkeit	O	+	-	+
Verwaltbarkeit	+	-	+	O
Zuverlässigkeit	+	+	-	O

Tab. 3.2: Bewertung der Verteilungsansätze

Im Rahmen von Kapitel 5 werde ich auf die Ergebnisse dieses Kapitels zurückkommen und eine Architektur in der Form der mehrstufigen verteilten Verarbeitung vorschlagen, die eine möglichst optimale Kombination der einzelnen Aspekte darstellt. Dabei werden unter anderem die robusten Dienste der Zentralrechner- und Client/Server-Architekturen wie Sicherheitkonzepte, Skalierbarkeit, Verfügbarkeit und Integritätssicherung über Transaktionen mit dem universellen Zugriffskonzept der WWW-Architekturen und der Flexibilität und Produktivität der verteilten Objektstrukturen kombiniert.

Kapitel 4

WIRTSCHAFT

In diesem Kapitel untersuche ich Aspekte einer Wirtschaft, die auf offenen, verteilten Informationssystemen beruht. Dazu definiere ich im ersten Abschnitt diese digitale Wirtschaft als Konvergenz von elektronischer Datenverarbeitung, Telekommunikation und den Anbietern klassischer Produkte und Dienstleistungen. Über die Beschreibung beispielhafter Anwendungen wird ein Bild dieser Wirtschaft, ihrer Vertriebswege und des dazugehörigen Marktes geschaffen. Anschließend definiere ich fünf Ebenen der Involvierung eines Unternehmens in diesen Markt und stelle diese Ebenen allgemeinen, abstrakten Metaphern der Nutzung digitaler Medien gegenüber. Abschließend benenne ich zwei Grundprinzipen dieses Marktes und fordere ein einheitliches Verarbeitungsmodell als Grundlage dieser Prinzipen. Dieses werde ich in Kapitel 5 als *Network Computing*-Referenzmodell erarbeiten und zur Verfügung stellen.

Zuvor werde ich aber in Abschnitt 4.2 weitere aktuelle Punkte aus dem Bereich der betrieblichen Informationssysteme diskutieren, die die Anforderungen aus dem Abschnitt 4.1 ergänzen. Dann werde ich eine Menge von Diensten ermitteln, die zur Unterstützung der Konzepte der digitalen Wirtschaft benötigt werden. Diese stelle ich in Form eines Infrastrukturprofils, ähnlich dem Datenbank-Profil in Kapitel 2, zusammen. Als Basis für die Definition der Anforderungen an das Profil dienen dabei die Aktivitäten von Unternehmen auf den definierten Ebenen eins bis fünf.

4.1 Digitale Wirtschaft

In diesem Abschnitt beschreibe ich den aktuellen Stand der digitalen Wirtschaft, wie sie zum Teil in [Mar96, Neg96, TLT98, Tap97, Sey98, Har97, Ste96] wiederzufinden ist, und ergänze diese durch meine eigene Einschätzung der Veränderungen im Rahmen des Übergangs von der Industriegesellschaft in die Informationsgesellschaft.

In seinem Buch *The Digital Economy* [Tap97] beschreibt Don Tapscott das Phänomen, daß sich ein neues wirtschaftliches Paradigma formt, das nicht mehr auf physikalischen sondern auf digitalen Werten beruht - die digitale Wirtschaft.

Abbildung 4.1 zeigt die Entstehung dieses Wirtschaftszweiges aus der Konvergenz von drei kritischen Technologien - EDV (*Computing*), Kommunikation

(*Communications*) und Inhalt (*Content*). Auch in [Col98] wird das Verschwimmen der Grenzen zwischen Information, Kommunikation und Unterhaltung beschrieben und die Nutzung von entsprechenden Synergieeffekten gefordert. Zu berücksichtigen ist dabei, daß sich diese Bereiche bisher unterschiedlich entwickelt haben.

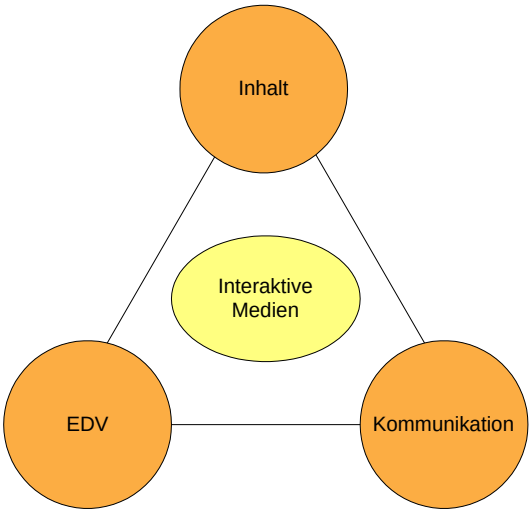

Fig. 4.1: Konvergierende Technologien

Die EDV-Branche ist eher an Produkten und Eigenschaften orientiert. Sie kreiert periodisch neue Versionen ihrer Produkte, die zwar meist abwärtskompatibel, aber nicht notwendigerweise interoperabel mit anderen Produkten, insbesondere anderer Hersteller sind. Interoperabilität wird in der Regel über Programmierschnittstellen (APIs) realisiert. Die Produkte sind meist monolithisch und als eigenständige Einheiten konzipiert. Netzwerke werden nur genutzt, um die eigenen Clienten mit den eigenen Anwendungen zu verbinden.

Die Telekommunikationsindustrie ist eher dienstorientiert ausgerichtet und betont besonders die Interoperabilität zwischen unabhängigen Herstellern. Die Komponenten sind natürlicherweise verteilt, so daß Wert auf eine angemessene Managementumgebung mit hoher Zuverlässigkeit und Verfügbarkeit gelegt wird. Standardisierung ist hier von enormer Bedeutung und wird oftmals in Form von Protokollen vorgenommen. Allerdings gibt es erst seit kurzem eine verstärkte Ausrichtung auf digitale Dienste und Anwendungen.

Auch die Produzenten der Inhalte müssen sich auf die veränderte Situation einstellen: Die Inhalte werden zunehmend interaktiv. Die Einbindung der Anforderer der Inhalte wird über die entstehenden neuen Techniken, zum Beispiel über interaktive Inhalte, immer umfassender.

So hat sich ein neuer Wirtschaftszweig entwickelt, der zunehmend an Bedeutung gewinnt: die interaktiven Medien[1]. In [Bec96] wird beschrieben, daß heute mehr Amerikaner Computer herstellen als Autos, mehr Arbeiter Halbleiter produzieren als Baumaschinen und mehr Menschen in der Datenverarbeitung arbeiten

[1] auch Neue Medien oder Online Medien genannt.

als in Raffinerien. So wie das Automobil das Aussehen der Welt physisch und auch sozial verändert hat, so können auch diese interaktiven Medien die Welt revolutionieren.

4.1.1 Individueller Massenmarkt

Im folgenden beschreibe ich, wie wir uns einer immer umfangreicheren Digitalisierung verschiedener Bereiche unseres täglichen Lebens nähern und sich daraus ein neuer Markt entwickelt. Die einzelnen Angebote reichen von der Kommunikation über *Electronic Mail* (eMail) bis hin zu komplexen Online-Bankgeschäften wie etwa dem Broker-Service der comdirect Bank[2], von wirtschaftlichen Beziehungen im *Business-to-Business*-Bereich wie den Händlersystemen von Computer2000[3] oder Cisco[4] bis hin zu Konsumermärkten wie im Fall des Versandhauses OTTO[5]. Auch der Abschluß von langfristigen Verträgen ist, wie das Beispiel der Deutschen Allgemeinen Versicherung[6] zeigt, bereits digital über das WWW möglich. Selbst die klassischen Medien, von Printmedien wie dem Hamburger Abendblatt[7] bis zu Fernsehsendern wie Pro7[8], etablieren eine Präsenz im Bereich der interaktiven Medien. Es gibt sogar Projekte, in denen Gebrauchsgegenstände des täglichen Lebens digitalisiert und an das Internet angebunden werden. So haben IBM, Netscape, SUN und General Motors ein *Networkvehicle*[9] konstruiert, das nicht nur die üblichen Internetdienste zur Verfügung stellt, sondern auch über das Internet überwacht werden kann.

Massenmedien und Massenproduktion werden in einem bisher nicht möglichen Grad personalisiert bzw. individualisiert. So bietet CNN[10] einen individuellen Nachrichtendienst, dem man an seine persönlichen Wünsche anpassen kann, an, und der Jeanshersteller Levi Strauss bietet mit seinem *Personal Pair* Dienst[11] kundenindividuelle Jeans an.

Aus den Angaben der Kunden bzw. der Interessenten lassen sich Profile erstellen und für Marketingzwecke nutzen. Je nach Gesetzeslage des einzelnen Landes ist dieser, *Profiling* genannte Vorgang, jedoch nur zulässig, wenn die entsprechende Person eingewilligt hat. In Deutschland etwa ist ein implizites *Profiling*, z.B. über besuchte Seiten in einem WWW-Angebot nicht zulässig. Explizites Profiling indes bietet jedoch Vorteile für Kunden und Unternehmen: Der Kunde erhält mit dem Konzept des *1-to-1-Marketing* eine auf ihn zugeschnittene Leistung und persönliche Betreuung, die Unternehmen gewinnen mehr Informationen über ihre aktuellen und potentiellen Kunden und können ihre Marktwirkung verstärken. So ist es zum Beispiel bei IBM möglich, durch Angabe der

[2] `http://www.comdirect.de`
[3] `http://www.intouch2000.de`
[4] `http://www.cisco.com/public/ordsum.html`
[5] Der Otto-Versand (`http://www.otto.de`) gab seinen Online-Umsatz im Jahr 1997 mit 437 Millionen Mark an.
[6] `http://www.deutsche-allgemeine.de`
[7] `http://www.abendblatt.de`
[8] `http://www.prosieben.de`
[9] `http://www.alphaworks.ibm.com/networkvehicle`
[10] `http://customnews.cnn.com`
[11] `http://www.levi.com/us/personal_pair`

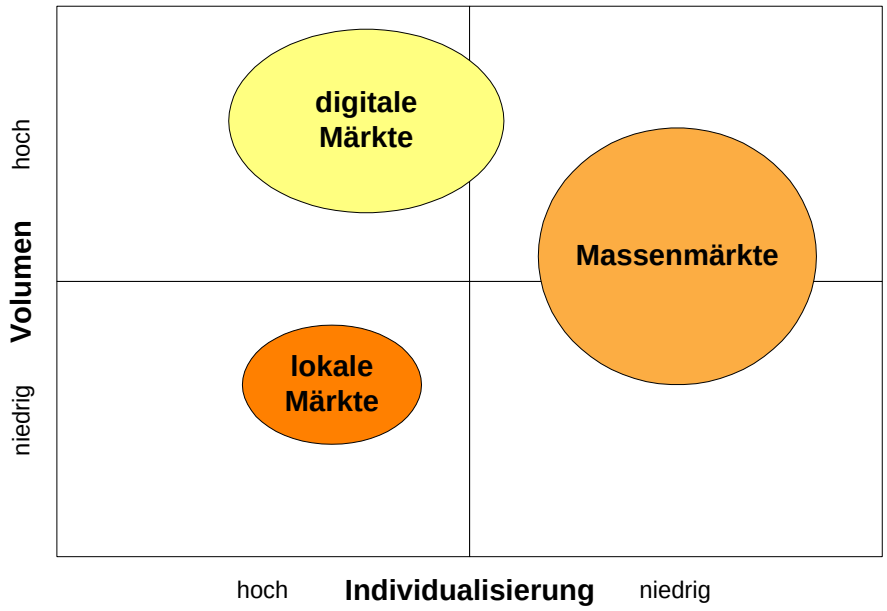

Fig. 4.2: Digitale Märkte

Seriennummern der erworbenen Geräte eine persönliche, individuelle Supportseite zu bekommen, auf der nur die aktuellen Informationen zu den erworbenen Produkten aufgeführt sind. Dies vereinfacht es für einen Kunden, die für ihn relevanten Informationen zu erhalten. Daß Menschen bei entsprechendem Gegenwert bereit sind, persönliche Informationen auch in großem Umfang preiszugeben, zeigt das Beispiel von Free-PC.com[12]. Hier haben mehr als eine halbe Million Menschen zugestimmt, ein umfangreiches persönliches Profil von sich erstellen zu lassen, um dann einen PC für zwei Jahre inklusive Internetzugang gestellt zu bekommen.

In Abbildung 4.2 ist diese Besonderheit des neuen Marktes dargestellt. Er verbindet die hohe Individualisierung eines lokalen Marktes[13] mit der breiten Streuung der klassischen Massenmärkte[14]. Laut einer KPMG Studie [KPM97a] bietet der digitale Markt im Konsumerbereich besondere Vorteile in puncto Kundenbindung, Markentreue und *Cross-Selling*-Potentiale durch die Nutzung des direkten Kontaktes zum Endkunden.

Zusammen mit der Individualisierung der Dienste und der Produkte gibt es damit einen zunehmenden Trend zur Kundenselbstbedienung unter Ausschaltung klassischer vermittelnder Einheiten wie Verkaufspersonal oder Einzelhandel. Dies wird als *Disintermediation* bezeichnet.

Neue Technologien haben stets entsprechende Veränderungen in den jeweiligen Branchen nach sich gezogen. So ersetzte etwa die Einführung automatischer

[12] `http://www.free-pc.com`
[13] z.B. persönliche Betreuung eines Friseurs
[14] z.B. anzutreffen im Bereich *Consumer Electronics*.

Vermittlungsstellen bei den Telefongesellschaften das *Fräulein vom Amt* und erhöhte gleichzeitig die Anzahl parallel zu berarbeitender Vermittlungen. Im Bankenbereich hat der Einsatz von Geldautomaten(*Automated Teller Machines (ATM)*) einen quasi weltweiten Zugriff der Kunden auf ihr Konto ermöglicht, sicher aber auch einigen Schalterangestellten einen anderen Arbeitsplatz beschert.

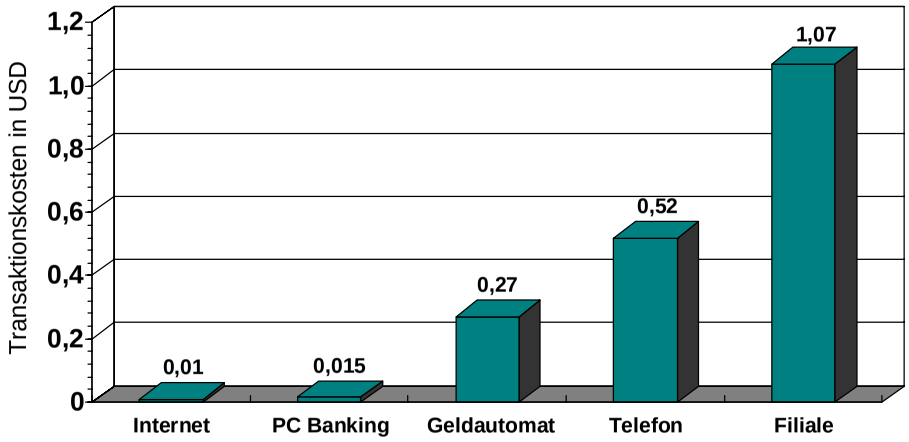

Fig. 4.3: Vergleich von Vertriebswegen im Bankenbereich

In Abbildung 4.3 wird entprechend [Rou98] ein Vergleich verschiedener Vertriebswege im Bankenbereich und ihrer Kosten je Transaktion in USD dargestellt. Man erkennt, daß die Kosten einer Internet-Transaktion weniger als 1% der Kosten einer Transaktion über eine Filiale ausmachen.

Das Potential dieses Unterschieds wird in Abbildung 4.4 durch Betrachtung der Wertschöpfungsketten erklärt. Der klassische Vertrieb der Produkte eines Herstellers erfolgt über die Wege des Groß- und Einzelhandels. Die verbleibende Wertschöpfung ist meist eher gering. Durch den Direktvertrieb der Produkte kann diese Wertschöpfung deutlich höher ausfallen. Bei materiellen Wirtschaftsgütern muß nur noch der Transport der Ware zum Kunden finanziert werden. Bei immateriellen Wirtschaftsgütern wie Software, Musik[15], Eintrittskarten oder Reisen kann sogar dieser Zwischenschritt entfallen.

Diese Ausschaltung vermittelnder Organisationen und die Nutzung des digitalen Absatzkanals zwischen Produzent und Konsument zeigt sich im Bereich der Touristik schon besonders ausgeprägt. So gibt die Holiday Inn Hotelkette[16] an, bereits 1995 im ersten halben Jahr ihrer WWW-Präsenz monatlich 11.000 Online-Reservierungen erhalten zu haben. Inzwischen haben die meisten größeren Fluggesellschaften, Hotelketten und Autovermietungen eine Möglichkeit des Direktvertriebs geschaffen.

Besonders affin ist auch die EDV-Industrie gegenüber dieser neuen Vertriebsstrategie. Der auf den Direktvertrieb spezialisierte Computer-Hersteller DELL[17] verzeichnete für das Jahr 1998 einen Online-Umsatz von über 4 Millionen USD

[15] z.B. im MPEG-Layer3-Format
[16] `http://www.holiday-inn.com`
[17] `http://www.dell.com`

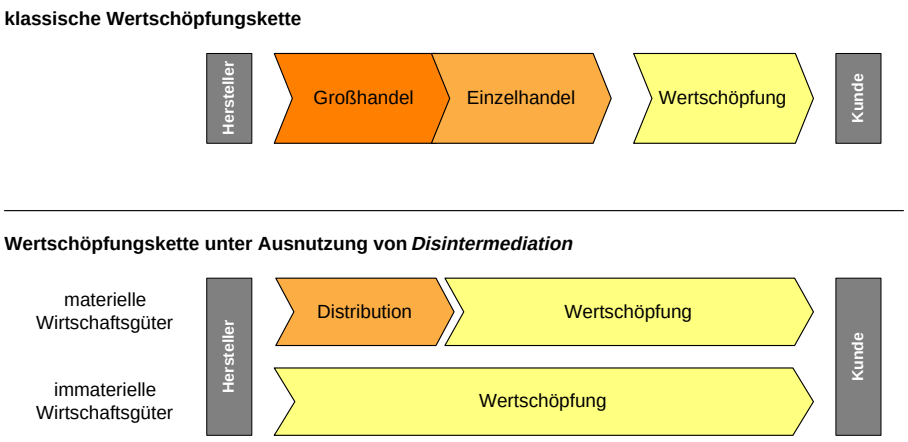

Fig. 4.4: Wertschöpfungspotentiale

pro Tag, und Cisco Systems erzielte laut [Rou98] mit 4 Milliarden USD bereits 50% seines Umsatzes online.

Es gibt allerdings auch Bestrebungen, die klassischen Vertriebskanäle in dieses neue Geschäftsparadigma zu integrieren. Anders als z.B. Amazon Books[18] bietet Libri[19] einen Online-Buchhandel unter Einbeziehung der Buchhandlungen vor Ort, bei denen man seine bestellten Bücher abholen kann. Ein optionaler Lieferservice wird durch die Buchhandlungen selbst zur Verfügung gestellt. Und Europas größtes Reisevertriebssystem Start Amadeus bietet über seinen Online-Auftritt[20] rund 120.000 Last-Minute- und Schnäppchen-Angebote an, Pauschalreisen von 31 Reiseveranstaltern zu ca. 1600 Zielorten, Linienflüge von 450 Airlines sowie Eintrittskarten von 250 Veranstaltern, wobei die Buchungen dann allerdings über eines von ca. 1.000 angeschlossenen Reisebüros weiter bearbeitet werden.

Es bleibt festzuhalten, daß der reine informations- und warenvermittelnde Teil der Wertschöpfungskette im Rahmen der digitalen Wirtschaft nur noch eine untergeordnete Rolle spielen wird, da diese Funktionen mittels des digitalen Mediums einfach vom Kunden selbst erfüllt werden können. Allerdings werden Dienstleister im Bereich der Bereitstellung sogenannter *Value Added Services*, die weder durch den Hersteller, noch durch den Kunden selbst zu erbringen sind, gefördert. Dazu zählen zum Beispiel die Dienste von PC Service Source[21], der die Garantieabwicklung für mehrere große PC Hersteller übernimmt oder CheckFree[22], der die Aufgaben im Rahmen der Rechnungsstellung und des Rechnungsausgleiches im Internet erfüllt. Es geht hier um die Ergänzung der Basisdienste der Hersteller um zusätzliche Dienste am Kunden.

Daneben eröffnen sich auch Möglichkeiten für eigentlich branchenfremde Unternehmen, durch die in Abbildung 4.1 dargestellte Konvergenz, sich neue Geschäfts-

[18] `http://www.amazon.com`
[19] `http://www.libri.de`
[20] `http://www.start.de`
[21] `http://www.pcservice.com`
[22] `http://www.checkfree.com`

bereiche zu erschließen. So verzeichnet das virtuelle Reisebüro des Softwareherstellers Microsoft[23] inzwischen einen wöchentlichen Umsatz von über einer Million USD.

Nach dieser Beschreibung von Erscheinungen und Möglichkeiten der digitalen Wirtschaft soll nun im folgenden Abschnitt eine Möglichkeit zur Kategorisierung der unterschiedlichen Angebote von Unternehmen in diesem Markt erarbeitet werden.

4.1.2 Fünf Stufen der digitalen Wirtschaft

Im folgenden definiere ich fünf Ebenen der Involvierung eines Unternehmens in die digitale Wirtschaft. Jede Aktivität eines Unternehmens im digitalen Markt läßt sich meiner Meinung nach in eine der in Abbildung 4.5 dargestellten fünf Ebenen einordnen. Dabei bezeichnet die Ebene I den Eintrittspunkt eines Unternehmens in die digitale Wirtschaft. Zusammen mit der Ebene II ergibt sich die Grundform des Engagements im Bereich der interaktiven Medien. Alle weiteren Ebenen bauen auf diesen beiden Ebenen auf. Eine Bereitstellung von Dienstleistungen der Ebenen III bis V ohne den grundlegenden Dienst der Information und einfachen Interaktion ist im Bereich der digitalen Wirtschaft unüblich.

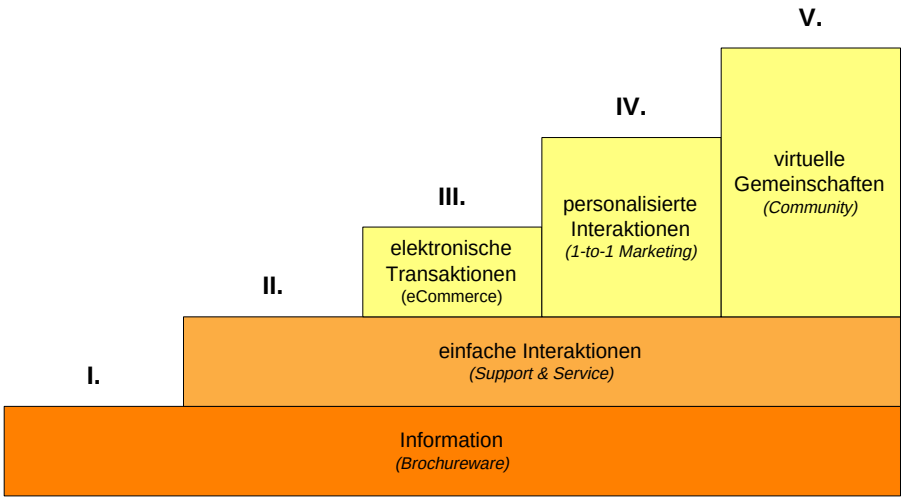

Fig. 4.5: Fünf Ebenen der digitalen Wirtschaft

Ebene I - Bereitstellung von Unternehmens- und Produktinformationen (*Brochureware*) Eine gebräuchliche Bezeichnung von Angeboten der ersten Ebene ist *Brochureware*. Diese beschreibt die bereitgestellte Dienstleistung recht gut. Sie stellt Unternehmens- und Marketinginformationen zur Verfügung, wie sie auch in Form von Prospekten und Broschüren existieren. Der Nutzen ist in erster Linie eine einfache Information von Kunden und Interessenten.

[23] `http://expedia.msn.com`

Ebene II - Einfache Interaktion mit dem Kunden (*Service & Support*) Die nächste Ebene ist die Unterstützung der Kunden durch Bereitstellung von Produktaktualisierungen (z.B. Patches) oder Zusatzprodukten (z.B. Zusatzlevel für Spiele). Auch Informationen über kundenspezifische Geschäftsvorgänge (z.B. Lieferstatus) stellen einen Wert dar, der einen Kunden stärker an ein Unternehmen binden kann. Natürlich gehört die Erreichbarkeit von einzelnen Abteilungen über eMail auch zu dieser Ebene.

Ebene III - Elektronische Transaktionen (*eCommerce*) Nachdem auf den ersten beiden Ebenen Informationen und Unterstützung über Produkte und Dienstleistungen des Unternehmens bereitgestellt werden, können mit dem nächsten Schritt diese Produkte und Dienstleistungen auch online erworben werden.

Ebene IV - Personalisierte Interaktionen mit dem Kunden (*1-to-1-Marketing*) In der vierten Ebene gilt es, den Kunden noch stärker an das Unternehmen zu binden. Möglich ist dies durch persönliches Profil des Kunden, um diesen so bei jedem Kontakt zum Unternehmen individuell und gezielt auf seine Vorlieben und Wünsche ansprechen zu können. Ist ein Kunde bereit, dem Unternehmen persönliche Informationen zu übermitteln, so ist das als Investition des Kunden in die Beziehung zum Unternehmen zu betrachten. Das so artikulierte Vertrauen in das Unternehmen ist die Basis für eine 1-zu-1-Beziehung zwischen Kunde und Unternehmen.

Ebene V - Bildung von virtuellen Gemeinschaften (*Communities*) Die letzte Ebene der Aktivitäten eines Unternehmens im Rahmen der digitalen Wirtschaft ist die Bildung einer Gemeinschaft von und mit den Kunden. Dabei geht es darum, die Kunden in die Dienste des Unternehmens zu integrieren. Ein Beispiel sind Diskussionsforen, in denen Kunden anderen Kunden mit ihrer Erfahrung bei Problemen weiterhelfen und somit eine neue Möglichkeit der Kundenunterstützung für das Unternehmen schaffen. Außerdem wird dem Kunden durch diese Plattform das Gefühl vermittelt, daß seine Meinung und Aktivitäten für das Unternehmen und andere Kunden von Bedeutung sind und er wird wiederkommen, um zu erfahren, was andere von seiner Meinung halten.

4.1.3 Grundprinzipien

Mit der Nutzung des Internets haben sich einige Metaphern etabliert, die den Umgang mit diesem neuen Medium kategorisieren. In [Ste96] werden folgende vier Metaphern erläutert, die ich hier den fünf Ebenen der digitalen Wirtschaft zuordne. Es läßt sich erkennen, daß trotz der unabhängigen Entwicklung der Metaphern und der wirtschaftlichen Entwicklungsstufen eine ziemlich hohe Übereinstimmung der zugrundeliegenden Konzepte vorliegt.

Die Digitale Bibliothek betont das Publizieren und Archivieren von gesammeltem Wissen, um es zu bewahren und der Gesellschaft zur Verfügung zu stellen. Dabei stellt sich unter anderem die Frage nach der Sicherstellung des

Urheberrechts an einzelnen Inhalten. Aktivitäten der ersten Ebene nach Abschnitt 4.1.2 entsprechen dieser Metapher.

Elektronische Post beschreibt die Nutzung als persönliches Kommunikationsmedium zwischen Individuen und als offenes Diskussionsforum für Gruppen mit gemeinsamen Themen. Die Erweiterung der Aktivitäten von Unternehmen auf die zweite Ebene der digitalen Wirtschaft basiert auf denselben Konzepten.

Der digitale Marktplatz bezieht sich auf den bereits beschriebenen Absatz von Gütern oder Dienstleistungen über das neue Medium. Zu dieser Metapher gehört der Ort des Handels genauso wie das Geld, die Produzenten, Händler und Anforderer von Gütern. Unternehmen, die elektronische Transaktionen über das Internet anbieten, handeln nach dieser Metapher.

Digitale Welten sind computergestützte Erfahrungsräume, die von Chat-Systemen[24] bis zu virtuellen Realitäten[25] reichen. Gemeinsame Grundlage aller unterschiedlichen Ausprägungen ist das gemeinschaftliche Erlebnis mit dem Schwerpunkt in der Kommunikation[26]. Wenn ein Unternehmen eine entsprechende digitale Gemeinschaft gebildet hat, befindet es sich auf der letzten Ebene der digitalen Wirtschaft.

Diese Metaphern versinnbildlichen jeweils unterschiedliche Perspektiven auf die digitale Wirtschaft. Basis dieser geschilderten Entwicklungen und Perspektiven sind meiner Meinung nach drei fundamentale Konzepte: der universelle Zugriff durch jedermann von jedem Ort, die Fähigkeit der verteilten Verarbeitung, also zum Beispiel die Produktion vom Konsum der Inhalte zu trennen und ein umfassendes, verbindendes Verarbeitungsmodell. Im folgenden erläutere ich diese Konzepte auf einer allgemeinen, abstrakten Ebene. Im anschließenden Abschnitt 4.2 werden diese Konzepte in bezug auf betriebliche Informationssysteme konkretisiert und bilden schließlich in Kapitel 5 zusammen mit den Kriterien und Erfahrungen aus Kapitel 3 die Grundlage für die Formulierung des *Network Computing*-Referenzmodells.

Universeller Zugriff Die wichtigste Idee ist die Realisierung eines universellen Zugriffs auf die digitale Welt. Dieser universelle Zugriff beseitigt die physikalischen und organisatorischen Grenzen zwischen einzelnen Systemen und ermöglicht es, Inhalte zu publizieren und Produkte und Dienstleistungen anzubieten, die sofort global zur Verfügung stehen und zwar mehr oder weniger unabhängig von der Lokation oder EDV-Umgebung potentieller Anforderer. Dies begründet auch eine neue Form der Freiheit[27] im Sinne der Redefreiheit. Die Fähigkeit, Inhalte und Angebote an eine große Anzahl von Menschen zu übermitteln, ist nicht mehr das Privileg bzw. Monopol einer Industrie, sondern ein

[24] zum Beispiel als Bestandteil des `http://www.investornet.de`.
[25] zum Beispiel komplette soziale Systeme wie in Ultima Online (`http://www.uo.com`)
[26] zur Vertiefung des *Community*-Gedanken finden sich interessante Anregungen in [HA97].
[27] nicht nur der Freiheit beliebige Inhalte zu konsumieren, sondern auch sie zu publizieren

persönliches Medium. Zudem können die Inhalte kaum von Regierungen oder anderen Organisationen kontrolliert werden[28] - auch eine Form der persönlichen Freiheit.

Verteilte Verarbeitung Mit verteilter Verarbeitung meine ich hier die Fähigkeit, Berechnungen entfernt von der Nutzung der Ergebnisse durchführen zu können. Dies umfaßt weit mehr als die klassischen zweistufigen, verteilten Verarbeitungsmodelle nach Abschnitt 3.1.2. Die Verteilung findet in geographisch deutlich ausgedehnteren Dimensionen statt. Im Rahmen ständig sinkender Kommunikationskosten nimmt die Bedeutung räumlich zusammengefaßter Datenverarbeitung kontinuierlich ab - eine weitere Analogie zum Übergang in das Industriezeitalter. Damals wurde durch die Einführung der Eisenbahntechnologie die räumliche Trennung der Produktion von Gütern vom Ort ihres Konsums ermöglicht. Die Produktion konnte aufgrund der günstigen Transportmöglichkeiten an Orte verlegt werden, die optimale Produktionsbedingungen boten. Ähnliches ist jetzt auch beim Übergang zum Informationszeitalter zu beobachten[29].

Einheitliches Verarbeitungsmodell Grundlage dieser zwei bisherigen Konzepte ist ein einheitliches Verarbeitungsmodell. Dieses wird über eine Menge von Standards und Basisdiensten die Vereinfachung der Erstellung und Nutzung von Hardware und Software unterstützen. Auch wenn diese Standards zunächst technischer Natur sind, werden sie um fachliche Standards erweitert werden müssen. Dies wird Organisationen in die Lage versetzen, sich auf fachliche Belange zu konzentrieren und diese einheitliche Infrastruktur zu nutzen. Im Rahmen dieses umfassenden Verarbeitungsmodells ist die Integration bestehender Anwendungen ein wichtiger Aspekt, da ein Großteil relevanter Geschäftsprozesse bereits in EDV-technischer Form vorliegt, mit denen die neuen Technologien in einer evolutionären Art und Weise kombiniert werden müssen.

Diese Grundprinzipen der neuen Informationsgesellschaft erfordern ein umfassendes Verständnis von Server- und Netzwerk-Technologien sowie ihrem globalen, allgegenwärtigen Zusammenspiel. Es ist erforderlich, eine neue Perspektive unabhängig von organisatorischen Grenzen zu entwickeln. Datenbanken und andere Datenquellen existieren nur noch im Rahmen des Netzwerkes. SUN Microsystem hatte schon in den 80er Jahren den Slogan *The Network is a Computer*. Anwendungen sind verteilt und in bisher noch nicht möglicher Art und Weise miteinander verbunden. Sie werden über heterogene Plattformen und unter Zugriff auf unterschiedlichste Datenquellen zusammenarbeiten. Der Computer wird zum Kommunikationsgerät[30]. Es gibt sogar Stimmen [Ste96], die eher vom Einzug des *Kommunikationszeitalters* sprechen als vom Informationszeitalter. Es wird eine netzwerk-zentrierte Perspektive in bezug auf die Realisierung von Informationssystemen benötigt.

[28] In Yugoslawien wurden inoffizielle Nachrichten nach Schließung der nicht-staatlichen Presse über das Internet veröffentlicht.

[29] Viele große Softwareprojekte werden z.B. in Indien durchgeführt, da dort die Personalkosten geringer sind. Auch die manuelle Erfassung von Massendaten wird in Ländern mit günstigem Lohnniveau organisiert.

[30] Die unter dem Begriff *Network Computer* (`http://www.opengroup.org/nc`) definierte Klasse von Geräten ist ein Beispiel für diese neue Sichtweise auf Computer.

Im folgenden Abschnitt werde ich unterschiedliche Aspekte von betrieblichen Informationssystemen diskutieren, die die Anforderung aus den Aktivitäten entsprechend den Ebenen aus Abschnitt 4.1.2 ergänzen und die Grundprinzipien aus Abschnitt 4.1.3 konkretisieren.

4.2 Betriebliche Informationssysteme

Nachdem ich im vorhergehenden Abschnitt das Konzept der digitalen Wirtschaft erarbeitet und strukturiert habe, wende ich mich in diesem Abschnitt besonderen Aspekten der informationstechnischen Infrastruktur von Organisationen zu. Im folgenden diskutiere ich dazu Punkte, die Gründe für eine Neuausrichtung der IT-Strategien von Unternehmen auf offene, verteilte Informationssysteme sein könnten. Diese Punkte ergänzen die Überlegungen aus Abschnitt 4.1.

4.2.1 Aspekte der Unternehmensstrategie

Für Unternehmen gibt es zur Zeit die klassischen Ziele der Effizienzsteigerung und Kostenreduzierung, andererseits ist auch eine Neuorientierung in bezug auf das in Abschnitt 4.1 geschilderte Geschäftsparadigma in Form von Kundenorientierung und vernetzten Geschäftsbeziehungen von nöten.

Effizienzsteigerung und Kostenreduzierung

In den 80er Jahren war das Instrument zur Erreichung von Wettbewerbsvorteilen das *Total Quality Management* (TQM) [SS98]. Über die stetige Verbesserung der Produktqualität sollte ein Vorteil gegenüber den Mitbewerbern erzielt werden. Die 90er Jahre sind die Zeit des *Business Process Reengineering* (BPR) [SS98]. Hiermit sollen weitere Vorteile durch eine Anpassung der organisatorischen Strukturen und Arbeitsprozesse an die aktuellen Geschäftsregeln herausgearbeitet werden. Theoretisch sollte dies vor allem in einer Verbesserung der Marktorientierung und damit des Kundendienstes münden, doch wird es bislang von den meisten Organisationen eher als Mittel zur Rationalisierung und Kosteneinsparung genutzt. Laut einer KPMG Studie [KPM97b] werden bis zum Jahr 2001 75% der größten englischen Firmen ein Intranet zu Verbesserung der internen Kommunikation und der Nutzung von Unternehmens-KnowHow einsetzen. Dabei kann die Wertschöpfung (*Return on Investment*) Größenordnungen zwischen 40% und 1000% erreichen, je nach Struktur des Unternehmens. Je größer die Unterstützung im Bereich der unternehmensinternen Kommunikation - gerade bei geografisch verteilten und sehr großen Unternehmen - , desto größer ist auch die Effizienzsteigerung. Aber auch im Bereich eines Extranets sind enorme Kosteneinsparungen möglich. So gibt der Netzwerkspezialist CISCO an [Rou98], durch sein *Cisco Connection Online* Einsparungen in Höhe von 474 Millionen USD erzielt zu haben, was einer Produktivitätssteigerung von 18 % entspricht.

Kundenorientierung

Nach dem BPR ist nun Kundenorientierung das aktuelle betriebswirtschaftliche Schlagwort, wenn es um die Neuausrichtung von Organisationen geht. Es zeichnet sich ab, daß Unternehmen, die sich einseitig auf die Optimierung der operativen Effizienz konzentrieren, einander immer ähnlicher und damit am Markt austauschbar werden. Laut [HW98] ist gesunde Profitabilität nur durch

ein Wachstum zu erreichen, das auf einer immer besseren Durchdringung der Geschäftsbeziehung mit den Kunden[31] basiert.

Durch die Ausweitung der Geschäftstätigkeit auf die digitalen Märkte mit dem direkten, teilweise interaktiven Kontakt zum Kunden läßt sich diese Unternehmensphilospohie besonders elegant realisieren. Aber auch die zunehmende Einführung von *Call Centern* für die Kunden- und Interessentenbetreuung mit der immer stärkeren *Computer Telephony Integration* (CTI) ist in diesem Zusammenhang von Bedeutung, da sie eine weitere Erscheinung im Rahmen der bereits besprochenen Konvergenz von Datenverarbeitung und Telekommunikationsbranche darstellt.

Vernetzte Geschäftsbeziehungen

Ein weiterer Aspekt ist die zunehmende Vernetzung von bisher autarken Organisationen. Im Bereich der Automobilindustrie ist die informationstechnische Anbindung der Zulieferer an die Hersteller schon länger Realität. Dies geschieht jedoch meist in Form von spezifischen Systemen und Protokollen. Eine Standardisierung ist, wenn überhaupt, nur im Bereich des Austausches von Geschäftsdaten über EDI vorzufinden.

Zukünftig werden sich das *World Wide Web* (WWW) und das Internet zunehmend zu der Infrastruktur für digitale Geschäftsbeziehungen entwickeln. Dies geschieht in Form des öffentlichen Internets, aber auch einer großen Anzahl von unternehmensinternen Netzen.

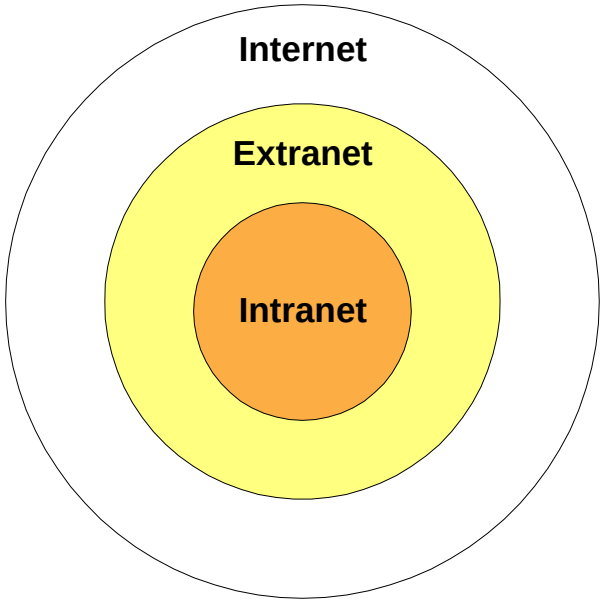

Fig. 4.6: Internet, Extranet und Intranet

[31] *Customer Relationship Management* (CRM)

Je nach Klassifizierung der beteiligten Partner und der Art der genutzten Inhalte können die in Abbildung 4.6 dargestellten und in Tabelle 4.1 beschriebenen unterschiedlichen Arten der Vernetzung identifiziert werden werden.

	Internet	**Extranet**	**Intranet**
Zugriff	Öffentlich	Beschränkt	Privat
Benutzer	Gelegenheitskunden, Interessenten, meist unbekannt	kooperierende Gruppe von Organisationen, bekannt	Begrenzt auf Mitarbeiter der Organisation, bekannt
Inhalt	öffentliche Informationen	vertrauliche Informationen mit Bezug auf die Kooperation	vertrauliche Informationen
Sicherheit	niedrige bis hohe Anforderungen, schwer sicherzustellen	hohe Anforderungen, schwer sicherzustellen	hohe Anforderungen, einfach sicherzustellen
Leistung	eher niedrige Anforderungen	hohe Anforderungen	hohe Anforderungen
Beispiele	Aktienkurse, Produktinformationen, öffentliche Ausschreibungen	Preislisten, Bestandsinformationen, Supportdaten	interne Stellenauschreibungen Urlaubsplanung, Kundeninformationen für die Hotline

Tab. 4.1: Arten der Vernetzung

4.2.2 Problembereiche

Im folgenden stelle ich die wichtigsten Problembereiche der oben geschilderten Neuausrichtung in der betrieblichen Informationsinfrastruktur mit ihren unterschiedlichen Aspekten vor.

Infrastrukturkosten

Der Betrieb der informationstechnischen Infrastruktur verschlingt einen ständig wachsenden Teil der Unternehmensressourcen. Die Einführung von klassischen Client/Server-Systemen hat nicht die erwünschten Kosteneinsparungen im Vergleich zu den zentralrechner-basierten Systemen gebracht (vgl. Gartner Group Inc. Report 1996). Die laufenden Betriebskosten übersteigen die Anschaffungskosten bei weitem. Laut dem *NC Manifesto* [Goo97] betragen die jährlichen Kosten für einen PC im Durchschnitt 13.000,- USD[32]. Diese Kosten entstehen

[32] Die Ergebnisse der Ermittlung der *Total cost of ownership* (TCO) von PC-Systemen durch Beratungshäuser wie Zona Research, IDC, Gartner Group, Forrester Research, Ovum und Giga Information Group bewegen sich zwischen 7.000,- USD und 20.000,- USD. Diese doch erheblichen Abweichungen basieren auf dem unterschiedlichen Umfang der verwendeten Modelle und berücksichtigten Aspekte. Trotz des Fehlens einer verbindlichen Formel für das TCO eines PC-Systems kann der in [Goo97] verwendete Mittelwert als realistisch angesehen werden.

zum größten Teil im Bereich der Verwaltung der Client-Maschinen in großen Netzwerken und beinhalten zum Beispiel Kosten der Softwareverteilung oder der Hardwarewartung. Nicht zu unterschätzen sind aber auch Produktivitätsverluste durch mangelhaften PC-Support und die gängige Praxis in Unternehmen, daß bei Problemen zunächst der Kollege vom Nachbartisch bemüht wird, bevor man die Fachabteilung hinzuzieht[33] In diesen Problemen manifestiert sich die Komplexität heute vorherrschender, PC-basierter Systeme. Das Fehlen einer einheitlichen Managementumgebung und die Notwendigkeit, eine Vielzahl von unterschiedlichen Technologien und Plattformen betreuen zu müssen, mündet in unhandlichen, kaum planbaren Systemen. Ziel müßte es daher sein, die Vorteile der Verwaltung zentralisierter Hostsysteme mit der Flexibilität der Client/Server-PCs zu verbinden.

Mangelnde Flexibilität und Integration

In heutigen EDV-Landschaften finden sich häufig verschiedene operationale Systeme, die kaum integriert sind. Sie entstanden durch die Entwicklung von produktspezifischen[34] oder funktionsspezifischen[35] Anwendungen. Dies führt zu einer Reihe von allgemein bekannten Problemen. Zum einen gibt es im Rahmen der bereits beschriebenen Kundenorientierung den Bedarf an einer umfassenden Sicht auf die Beziehungen von Kunden zum Unternehmen. Diese ist beim Einsatz von separaten Teilsystemen[36] für die unterschiedlichen Bereiche eines Unternehmens kaum zu realisieren.

Auch der Trend zum *Data Warehousing*, wie in Abschnitt 2.2.4 beschrieben, bedingt eine Integration der Daten unterschiedlichster Anwendungen. Dies bedeutet in der Regel den Zugriff auf operationale Daten in einer nicht vorgesehenen Form. Gerade bei den meist hostbasierten, sogenannten *Legacy*-Systemen führt die mangelnde Flexibilität zu Schwierigkeiten der zweckfremden Nutzung von Daten.

Ein weiterer Grund für die Integration unterschiedlicher Systeme ist die zunehmende Konzentration von Unternehmen durch Fusionen oder Übernahmen. Ohne eine angemessene Verbindung der bisher separaten Systeme der beteiligten Unternehmen können die gewünschten positiven Effekte dieser Vorgänge eingeschränkt werden.

Anwendungsstau

Unter diesem Begriff versteht man die Erscheinung, daß aufgrund der bereits erwähnten Ressourcenbindung im Bereich der Aufrechterhaltung des laufenden Betriebs dringend benötigte Anpassungen oder Neuentwicklungen nicht realisiert werden können. Das wirkt sich insbesondere auf die Einführung neuer Produkte oder Dienstleistungen aus. Ohne eine ausreichende EDV-Unterstützung sind derartige marktrelevante Aktionen kaum noch möglich. Somit kann die

[33] Auch die standardmäßig mitgelieferten Spiele der MS Windows Betriebssysteme bewirken entsprechende Opportunitätskosten.
[34] z.B. verschiedene Versicherungssparten
[35] z.B. Finanzbuchführung, Logistik
[36] auch unter dem Begriff *Insellösungen* bekannt.

sogenannte *Time to Market* als Funktion der Effizienz und Anpaßbarkeit der informationstechnischen Infrastruktur gesehen werden.

Benutzererwartungen

In einer Zeit, in der der PC in den Privathaushalten langsam den Status eines Gerätes wie Fernseher oder Telefon bekommt, steigen auch die Erwartungen der Benutzer an die Ausstattung ihres Arbeitsplatzes im Unternehmen. Während in den Wohn- bzw. Kinderzimmern der Familien die aktuellste Multimedia-Ausstattung vorzufinden ist, arbeiten die Mitarbeiter in den Unternehmen meist mit deutlich älterer Technik und unansprechenden Benutzeroberflächen, was nicht gerade zur Motivation qualifizierter Mitarbeiter beiträgt. Auch der Bereich der externen Unternehmenskommunikation, etwa WWW-Präsenzen, muß den Erwartungen anspruchsvoller Adressaten gerecht werden.

4.2.3 Grundprinzipen

Nachdem nun aktuelle Problembereiche von Unternehmen skizziert wurden, setze ich sie mit den Grundprinzipen aus Abschnitt 4.1 in Beziehung, die dadurch in bezug auf die Fragestellung der Arbeit konkretisiert werden.

Universeller Zugriff

Das Prinzip des universellen Zugriffs steht in diesem Zusammenhang für die Möglichkeit, mit beliebigen Endgeräten, basierend auf einer minimalen Anforderung an deren Funktionsumfang[37], auf die Informationen und Anwendungen eines Unternehmens zugreifen zu können. Dadurch wird die Nutzung deutlich günstigerer Endgeräte als PCs ermöglicht und entsprechende Einsparungen werden realisierbar.

Ein wichtiger Schritt in Richtung Kundenorientierung ist in diesem Zusammenhang eine Öffnung der unternehmensinternen Systeme für die Einbindung der Kunden mit den gleichen minimalen Anforderungen an deren technische Ausstattung. Durch den direkten Kontakt zum Kunden lassen sich seine Erwartungen und Reaktionen besser auswerten und entsprechende Marketingaktionen besser steuern. Durch die Möglichkeit zur Auswertung der Kontakthistorie können dem Kunden individuelle Dienstleistungen angeboten werden.

Zudem ermöglicht eine Standardisierung der Anforderungen an Endgeräte die Kooperation mit anderen Unternehmen, um so den Kunden zusätzliche Dienste anbieten zu können. Ein Beispiel ist die Kooperation des Suchdienstes `http://www.fireball.de` mit der Online-Vertretung der deutschen Buchhandlungen
`http://www.libri.de`. Nach Eingabe eines Suchbegriffs wird dem Kunden zusätzlich zu den gefundenen WWW-Adressen angeboten, sich relevante Bücher zu dem Thema anzuschauen.

[37] sogenannten *Thin-Clients*.

Verteilte Verarbeitung

Durch die Möglichkeit der Verteilung von Verarbeitungskapazitäten können entsprechende Ressourcen besser genutzt werden. Außerdem besteht die Möglichkeit des *Outsourcing* von Teilbereichen an entsprechende Dienstleister. Die eben beschriebene Zusammenarbeit von Fireball und Libri ist ein Beispiel für diese Art der verteilten Verarbeitung. Außerdem ist die Integration bestehender Altanwendungen über die gleichen Prinzipien vorzunehmen.

Einen weiteren Aspekt bildet die Zusammenfassung von Anwendungsfunktionalität auf ausgezeichneten Anwendungsservern. Im Zusammenhang mit den beschriebenen minimalen Endgeräteanforderungen lassen sich Anwendungen für eine große Anzahl heterogener Clients betreiben. Dies vermindert unter anderem den Aufwand der Softwareverteilung. Im Zusammenhang mit den oben aufgeführten Endgeräten lassen sich damit die Kosten der informationstechnischen Infrastruktur deutlich senken.

Außerdem wird die Skalierbarkeit der Anwendungen durch die Möglichkeit der Erweiterung der jeweiligen Verarbeitungskapazitäten erhöht. Auch die Verwaltung der Anwendungen auf den Anwendungsservern gestaltet sich weniger aufwendig als auf einer entsprechenden Anzahl von Endgeräten. Es wird der potentielle Gefahrenbereich von vielen, unsicheren und verteilten PCs auf wenige zentrale Server minimiert, für die angemessene Sicherheitslösungen realisiert werden können. Außerdem kann ein zentrales, qualifiziertes Team von Operateuren eingesetzt werden, das die Verfügbarkeit der Server-Systeme besser garantieren kann, als eine große Anzahl von Betreuern für die verteilten PCs.

Die Clients dieser Architektur sind nur noch für die Bereitstellung der Benutzerschnittstelle und die Zwischenablage von Informationen für den mobilen Einsatz zuständig.[38]

Einheitliches Verarbeitungsmodell

Grundlage für die Realisierung der zwei bisher besprochenen Prinzipen ist ein einheitliches Verarbeitungsmodell, das eine übergreifende Infrastruktur für den Betrieb und die Verwaltung der Anwendungen zur Verfügung stellt. Dieses Modell soll auf umfangreichen, standardisierten Diensten und einem leistungsfähigen Komponentenmodell [Gri98] im Sinne von Abschnitt 3.2.4 basieren. Entwickler werden damit in die Lage versetzt, sich auf die Lösung der fachlichen Probleme zu konzentrieren und von den technischen Details zu abstrahieren.

Außerdem besteht die Möglichkeit, fertige Komponenten für bestimmte technische oder fachliche Funktionen zu kaufen. Ein Beispiel hierfür wäre eine Zahlungsverkehrskomponente, die die Online-Abwicklung von Kreditkartenzahlungen realisiert und deren Dienste über die Infrastruktur allen Anwendungen gemäß dem einheitlichen Verarbeitungsmodell zu Verfügung stehen.

Über die gemeinsame Infrastruktur wird auch die Integration von unterschiedlichen Anwendungen unterstützt. Selbst Altanwendungen können, über entsprechende Komponenten gekapselt, ihre Funktionalität allen beteiligten Systemen

[38] Diese Art der Verarbeitung läßt sich mit folgender Metapher verdeutlichen - unserer Nutzung von Geld. Der Großteil unseres Geldes liegt bei sicheren und effizienten Finanzinstituten; wir hingegen tragen nur einen kleinen Betrag an Bargeld mit uns.

zur Verfügung stellen. Zudem wird durch die ausschließliche Verwendung von akzeptierten Standards die Interoperabilität zwischen Systemen unterschiedlicher Organisationen garantiert.

Desweiteren vereinfacht ein übergreifendes, eingebettetes Verwaltungskonzept den Betrieb der komplexen Systeme von den Arbeitsplätzen über die Netzwerk-Infrastruktur bis zu den Servern.

Auch der Austausch von Informationen muß über standardisierte Formate erfolgen und nicht über proprietäre Strukturen wie bei vielen aktuell verfügbaren EDI-Systemen.

Auch wenn es scheint, als wenn hier die Rückkehr zu den alten zentralrechnerbasierten Systemen postuliert wird, geht die Vision doch viel weiter. Die Betonung liegt zwar auf einer Verschiebung vom *Fat-Client* zum *Thin-Client* mit den entsprechenden Vorteilen einer zentralen Verwaltung, aber im Zusammenhang mit einer netzwerk-zentrierten Perspektive für die Gestaltung von offenen Systemen.

Der folgende Abschnitt faßt die bisher gesammelten Erfahrungen zusammen und ermittelt aus diesen ein Anforderungsprofil, das die Basis für die Entwicklung des *Network Computing*-Referenzmodells sein wird.

4.3 Profil

In diesem Abschnitt werde ich ein Infrastrukturprofil als eine Zusammenstellung von Diensten definieren, die auf der Basis der Konzepte aus den Abschnitten 4.1 und 4.2 ermittelt werden.

4.3.1 Ermittlung der Dienste

Dazu verwende ich eine Vorgehensweise, bei der Infrastrukturbestandteile anhand von Eigenschaften einzelner Geschäftsprozesse erarbeitet werden. Dieses Vorgehen unterscheidet sich von anderen dadurch, daß nicht primär technische Gründe für die Auswahl von Infrastrukturdiensten herangezogen werden, sondern die Unternehmensziele auf Infrastrukturentscheidungen abgebildet werden.

Ich verwende dazu Aktivitäten der in Abschnitt 4.1.2 definierten fünf Ebenen der digitalen Wirtschaft, um Dienste zu ermitteln, die diese Aktivitäten unterstützen können.

Information

Für die Bereitstellung von Unternehmens- und Produktinformationen muß zunächst eine Möglichkeit geschaffen werden, diese Inhalte in einer angemessenen Form zu definieren und darzustellen. Dies wird durch einen **Präsentationsdienst** ermöglicht, der die Darstellung an der Bedienerschnittstelle und die Steuerung der Interaktion übernimmt. Außerdem müssen die Informationen auch langfristig zur Verfügung stehen und nach Bedarf abgerufen werden können, was die Aufgabe eines **Persistenzdienstes** ist. Weiterhin müssen die einzelnen Angebote und Informationen auffindbar sein. Die Lokalisierung von Ressourcen der digitalen Welt ist dabei über einen **Verzeichnisdienst** zu realisieren. Schließlich müssen die Informationen auch vom Anbieter zum Konsumenten gelangen. Diese Aufgabe wird von einem **Transportdienst** erfüllt.

Einfache Interaktion

Auf dieser Ebene wird eine einfache Kommuikation zum Kunden aufgebaut. Diese basiert auf asynchroner Kommunikation in Form elektronischer Post, synchroner Kommunikation in Form eines *Chat* und der Übertragung von Datendateien. Hierzu werden entsprechende Angebote eines **Kollaborationsdienstes** genutzt werden müssen.

Elektronische Transaktionen

Sobald ein Unternehmen seine Dienstleistungen und Produkte am digitalen Markt auf der Basis elektronischer Geschäftstransaktionen anbietet, muß es in erster Linie die Sicherheit dieser Transaktionen über einen entsprechenden **Sicherheitsdienst** gewährleisten können. Eine solche Transaktion muß selbstverständlich die üblichen Eigenschaften einer Transaktion entsprechend Abschnitt 2.3.2 aufweisen, was durch einen entsprechenden **Transaktionsdienst**

zu realisieren ist. Da ein Teil der entsprechenden Geschäftsprozesse meist bereits in einzelnen Informationssystemen existiert, ist die Integration dieser Systeme zur Abwicklung der Transaktionen über einen **Integrationsdienst** sinnvoll.

Personalisierte Interaktionen und virtuelle Gemeinschaften

Die beiden letzten Ebenen der digitalen Wirtschaft benötigen keine zusätzlichen Dienste, über die bereits in den Ebenen davor geforderten Dienste hinaus, sondern nur einzelne Erweiterungen. Im Bereich der virtuellen Gemeinschaften sind eventuell erweiterte Dienste in den Bereichen Kollaboration (z.B. schwarze Bretter) oder neue Inhaltstypen im Bereich des Präsentationsdienstes (z.B. drei-dimensionale Darstellung von Räumen) hilfreich.

Betriebliche Informationssysteme

Der Betrieb und die Entwicklung betrieblicher Informationssysteme erfordert eine Reihe von Diensten, die zusätzlich zu den bisher ermittelten Dienste bereitgestellt werden müssen. Für die Entwicklung der Informationssysteme werden **Programmiersprachen** benötigt, die durch **Komponentenmodelle** in bezug auf Produktivität und Flexibilität optimiert werden können. Außerdem muß die oft komplexe Verwaltung der Informationssysteme und der darunterliegenden Infrastruktur über **Managementdienste** unterstützt werden.

Nach dieser problemorientierten Analyse von Anforderungen an unterstützende Dienste stelle ich diese nun in einem Profil zusammen.

4.3.2 Infrastrukturprofil

Für das Profil gliedere ich die ermittelten Dienste in die drei in Abschnitt 4.1 beschriebenen konvergierenden Kategorien **Content**, **Communication** und **Computing** sowie eine **Meta**-Kategorie. Das so erhaltene Profil ist in Tabelle 4.2 dargestellt.

CONTENT	
Präsentation	Die Präsentationsdienste ermöglichen die Darstellung der Inhalte an der Bedienerschnittstelle und die Steuerung der Interaktionen.
Persistenz	Dieser Dienst bietet die dauerhafte Ablage und bedarfsgerechte Wiedergewinnung der Inhalte sowie der Informationen, die für den laufenden Betrieb der Anwendungen benötigt werden.
COMMUNICATION	
Transport	Der Transportdienst dient der Übermittlung von Inhalten und Interaktionen zwischen den Kommunikationspartnern. Er muß mit dem Sicherheitsdienst kompatibel sein, um den entsprechenden Anforderungen genügen zu können.

Kollaboration	Zu dem Bereich der Kollaborationdienste zählen alle Dienste, die das gemeinsame Arbeiten von Personen unterstützen, wie elektronische Post, schwarze Bretter und der Austausch von Dateien.
Verzeichnis	In verteilten Umgebungen bedarf es standardisierter Verzeichnisse, um sämtlichen Kommunikationspartnern in Systemen der digitalen Wirtschaft über einheitliche Schnittstellen das Lokalisieren von Personen, Diensten und Ressourcen zu ermöglichen.
Sicherheit	Dieser Dienst sichert, daß die Vertraulichkeit und Authentizität der verarbeiteten und übertragenen Information gewahrt bleibt und die Infrastruktur vor unautorisierten Zugriffen geschützt ist. Dazu ermöglicht er in erster Linie die Identifizierung von Kommunikationspartnern und die verschlüsselte Übertragung von Inhalten unter Sicherstellung der Integrität der übermittelten Informationen.
COMPUTING	
Integration	Der Integrationsdienst stellt Möglichkeiten zur Anbindung existierender Informationssysteme an Systeme der digitalen Wirtschaft zur Verfügung. In diesen Dienst fallen Möglichkeiten zur synchronen und asynchronen Kopplung von Informationssystemen.
Transaktion	Der Transaktionsdienst bietet eine robuste, skalierbare und transaktionale Verarbeitungsumgebung für die Anwendungen der digitalen Wirtschaft. Dazu gehört unter anderem die Bereitstellung von verteilten Transaktionen und Möglichkeiten zur transparenten Lastverteilung.
Programmierung	Der Programmierdienst stellt Programmier- und Skriptsprachen zur Implementierung von einzelnen problemorientierten Anwendungen zur Verfügung.
Komponenten	Der Komponentendienst ermöglicht den Einsatz von Komponenten für die Erfüllung von problemorientierten, aber auch systemnahen Aufgaben. Er soll die Wiederverwendung vorgefertigter Bausteine unterstützen und damit die Anwendungsentwicklung beschleunigen und dabei gleichzeitig die Qualität der Softwaresysteme steigern.
META	
Management	Der Bereich der Managementdienste umfaßt die Verwaltung, Überwachung und Kontrolle aller Elemente der Infrastruktur.

Tab. 4.2: Infrastukturprofil

Dieses Infrastrukturprofil dient in Kapitel 6 einer ersten Prüfung des Umfangs der Unterstützung des *Network Computing* durch Datenbanksysteme. Allerdings reicht ein reiner Abgleich von bereitgestellten Diensten für eine abschließende Beurteilung nicht aus. Es müssen insbesondere auch der Umfang der einzelnen Dienste sowie die Form des Zugriffs auf diese Dienste analysiert werden. Aus diesem Grund werde ich unter Berücksichtigung der Ergebnisse des Kapitels 3 im folgenden Kapitel 5 ein *Network Computing*-Referenzmodell entwerfen, das die einzelnen Dienste des hier erstellten Profils konkretisiert.

Teil II

SYNTHESE

Kapitel 5

NETWORK COMPUTING

Mit diesem Kapitel beginnt die Synthese der Ergebnisse aus der Analyse in den ersten drei Kapiteln. Zunächst definiere ich den Begriff *Network Computing* als technische Repräsentation des Konzeptes des **gemeinsamen Verarbeitungsmodells** aus Kapitel 4.

Ich werde die Ergebnisse aus den Kapiteln 3 und 4 verwenden, um ein *Network Computing*-Referenzmodell zu entwicklen, das unter Bereitstellung der Dienste des Infrastrukturprofils die Vorteile der einzelnen beschriebenen Verteilungsansätze optimiert und die jeweiligen Nachteile minimiert. In diesem Rahmen wähle ich für die einzelnen Dienste des Infrastrukturprofils relevante Technologien aus, die die Konzepte der digitalen Wirtschaft realisieren und die Vorteile der ausgewählten Verteilungsansätze umsetzen können. Abschließend wird ein Beispielszenario beschrieben, in welchem eine Anwendung der digitalen Wirtschaft realisiert wird.

5.1 Einordnung

Zur Vorbereitung der Erstellung des Referenzmodells beschreibe ich zunächst den Zweck des Modells und die Kriterien für die Auswahl von Technologien für das Modell. Anschließend stelle ich jeweils eine verteilungsorientierte und eine dienstorientierte Sicht auf das *Network Computing*-Referenzmodell dar.

5.1.1 Kriterien und Ziele

Das Infrastrukturprofil aus Kapitel 4 beschreibt in erster Linie eine Reihe von Dienstbereichen, die benötigt werden, um Anwendungen der digitalen Wirtschaft zu realisieren. Um ein abgerundetes Bild des *Network Computing* zu erhalten, reicht dies jedoch nicht aus. Es müssen darüber hinaus der Umfang der einzelnen Dienste sowie die Form des Zugriffs auf diese Dienste beschrieben werden.

Aus diesem Grund erstelle ich im folgenden ein *Network Computing*-Referenzmodell, das die einzelnen Dienste des Infrastrukturprofils konkretisiert. Der **Zweck** des Referenzmodells ist dabei:

- die Schaffung eines Rahmens, in dem konkrete *Network Computing*-Plattformen eingeordnet und verglichen werden können,

- das Aufzeigen von Diensten und Dienstgruppen und ihren möglichen Abhängigkeiten und
- die Förderung des Verständnis und des Gedankenaustauschs von bzw. über das *Network Computing* .

Für die Auswahl von Technologien für das Referenzmodell werde ich folgende **Kriterien** anwenden. Im Rahmen dieses Referenzmodells werde ich grundsätzlich nur

- herstellerübergreifende,
- plattformübergreifende,
- standardisierte[1] und
- allgemein akzeptierte

Dienstausprägungen als Alternativen für die einzelnen Dienste berücksichtigen[2].

Ziel ist es, eine Synthese aus den in Kapitel 3 beschriebenen Verteilungsalternativen zu finden, die Synergieeffekte nutzt, um die gemeinsamen Vorteile zu maximieren und die beschriebenen Nachteile soweit wie möglich zu reduzieren[3]. Dabei gilt es im einzelnen

- die Verwaltbarkeit und Zuverlässigkeit der Zentralrechner-Architekturen,
- mit der Leistungsfähigkeit und den umfangreichen Middleware-Diensten der Client/Server-Welt,
- der hohen Interoperabilität und Portabilität der WWW-Anwendungen und
- der Flexibilität und Skalierbarkeit der verteilten Objekte

zu vereinen und dabei die jeweils mit den einzelnen Architekturen verbundenen Probleme zu minimieren. Dazu verwende ich die in Tabelle 5.1 angegebene Zuordnung von Kriterien aus Abschnitt 3.3.1 zu Diensten aus Abschnitt 4.3.2.

Diese Tabelle bietet einen Überblick über weitere Kriterien, die implizit in die Auswahl einzelner Technologien einfließen. Dabei lassen sich einzelne Cluster entdecken, die die Schwerpunkte bei der Auswahl prägen. So ist zu erkennen, daß der Bereich der Persistenzdienste sowie Dienste der Kategorie *Computing* auf Technologien aus der Zentralrechner- und Client/Server-Welt basieren. Die Dienste im Bereich *Communication* werden hingegen eher Internet-Technologien sein.

[1] *de jure* sowie *de facto*

[2] Natürlich engagiert sich auch Microsoft in diesem Markt - allerdings mit eigenen Standards. Die Versuche der Konkurrenten werden als Störgeräusche (*NOISE* - Netscape, Oracle, IBM, Sun and Everyone else) bezeichnet. Obwohl viele der Microsoft-Standards (z.B. Active/X) allgemein akzeptiert sind, schließe ich sie wegen mangelnder Herstellerunabhängigkeit aus der Auswahl aus.

[3] vgl. Tabelle 3.2.

Dienste		Kriterien / Sicherheit	Integrität	Verwaltbarkeit	Zuverlässigkeit	Ausgereiftheit	Produktivität	Benutzerfreundlichkeit	Interoperabilität	Kosten	Portabilität	Flexibilität	Skalierbarkeit
CONTENT	Präsentation					X		X	X	X	X	X	
	Persistenz	X	X	X	X	X							
COMMUNICATION	Kollaboration	X				X		X	X	X	X		
	Verzeichnisse	X				X			X	X	X		
	Sicherheit	X	X			X			X	X			
	Transport	X				X			X	X	X	X	
COMPUTING	Transaktionen	X	X	X	X	X							X
	Integration		X	X	X	X			X	X		X	X
	Komponenten				X	X	X				X	X	X
	Programmierung					X	X			X	X	X	
META	Management			X		X			X	X			

Legende: Zentralrechner; Client/Server; WWW; Verteilte Objekte

Tab. 5.1: Dienste zu Kriterien

5.1.2 Verteilungssicht

Als abstrakten Verteilungsansatz wähle ich eine Variante der *mehrstufigen, verteilten Verarbeitung* entsprechend Abschnitt 3.1.2, die in Abbildung 5.1 dargestellt ist. Diese Darstellung läßt durch ihre Ähnlichkeit mit Abbildung 3.7 den Ursprung des *Network Computing* in den WWW-Architekturen nach Abschnitt 3.2.3 erkennen. Dies wird in der historischen Beschreibung der Entwicklung der Zugriffs-Ebene und der Anwendungs-Ebene noch deutlicher werden. Allerdings wende ich bei diesem Modell auch Konzepte aus der Programmiersprachenwelt an. So verwende ich für die Bezeichnung der einzelnen Ebenen Begriffe entsprechend den Konzepten aus dem Anwendungsrahmenwerk der Programmiersprache *Smalltalk* [Vis95].

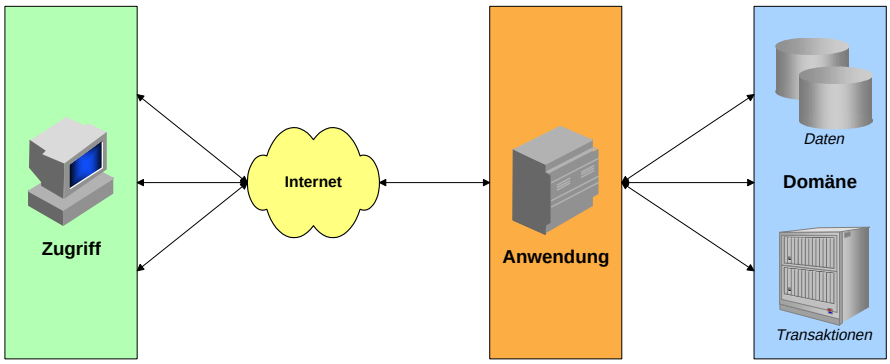

Fig. 5.1: *Network Computing*-Architekturmodell

Die einzelnen Ebenen erfüllen dabei die folgenden Aufgaben:

- **Zugriff**
  Die Zugriffs-Ebene realisiert die Benutzerschnittstelle zu den Anwendungen der digitalen Wirtschaft. Sie bietet den Zugriff auf die Anwendungen sowie auf grundlegende Kollaborationsdienste. Die Zugriffs-Ebene ermöglicht die Darstellung und Nutzung des Contents. Diese Funktion wird durch die *Browser* genannten Endgeräte zugänglich gemacht.

- **Anwendung**
  Die Anwendungs-Ebene realisiert die einzelnen Anwendungen der digitalen Wirtschaft. Sie nutzt dafür die Zugriffs-Ebene für die Interaktion mit den Anwendern und bedient sich Datenhaltungsfunktionen oder Transaktionen der Systeme aus der Domänenebene. Damit ist sie auch der Integrationspunkt für Systeme, die vor der Vision der digitalen Wirtschaft entstanden sind. Sie erfüllt dabei zu einem großen Teil integrierende und vermittelnde Aufgaben. Die dafür benötigten Dienste werden meist durch sogenannte *Anwendungsserver* bereitgestellt.

- **Domain**
  Die Domain-Ebene verwaltet einerseits den Content und stellt andererseits die Verbindung zu existierenden Daten und Transaktionen in anderen Systemen dar. Die entsprechenden Dienste werden durch Datenbankserver oder sogenannte *Legacy*-Systeme erbracht.

Die Dienste des Infrastrukturprofils spielen in den verschiedenen Verteilungs--Ebenen unterschiedliche Rollen, die in der folgenden Dienstsicht deutlich werden.

5.1.3 Dienstsicht

In Abbildung 5.2 ist die Aufteilung der Dienste auf die verschiedenen Verteilungsebenen dargestellt. Ich verwende hier die gleiche Gliederung der Dienste in die drei konvergierenden Wirtschaftsbereiche wie in Abschnitt 4.3.2.

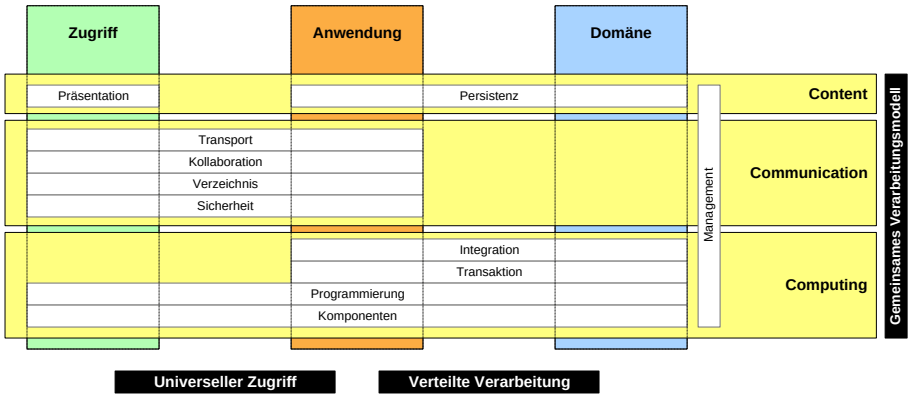

Fig. 5.2: *Network Computing*-Dienstverteilung

Der Bezug zu den Grundprinzipien aus Abschnitt 4.1.3 ist dabei folgender: Die Beziehung zwischen *Zugriff* und *Anwendung* ist relevant für das Konzept des **universellen Zugriffs**. Der Zusammenhang von *Anwendung* und *Domain* repräsentiert das Konzept der **verteilten Verarbeitung**. In der Summe dieser Dienste auf den unterschiedlichen Ebenen manifestiert sich das **gemeinsame Verarbeitungsmodell**.

Für den **universellen Zugriff** sind dabei Dienste aus den Bereichen:

- Präsentation,
- Kollaboration,
- Sicherheit,
- Verzeichnisse und
- Transport

relevant. Das Konzept der **verteilten Verarbeitung** wird durch folgende Dienste realisiert:

- Persistenz,
- Integration und
- Transaktion.

Programmierung und Komponenten sind in dieser Perspektive nicht eindeutig einem Konzept zuzuordnen, sondern bilden die Basis für die Realisierung von Anwendungen der digitalen Wirtschaft im Rahmen dieser Konzepte. Unter Berücksichtigung dieser Dienstverteilung ist *Network Computing* für mich ein Verarbeitungsparadigma, das die Infrastruktur der digitalen Wirtschaft primär auf der Basis von Internet-Technologien repräsentiert.

5.2 Auswahl der Technologien

Im folgenden Abschnitt werde ich die, für jeden der 11 Dienstbereiche von mir ausgewählten Technologien vorstellen, die die Vision des **gemeinsamen Verarbeitungsmodells** für die digitale Wirtschaft erfüllen. Bei der Auswahl fanden die in Abschnitt 5.1 definierten Kriterien Anwendung. Innerhalb der einzelnen Dienstbereiche werde ich zunächst deren Bedeutung in bezug auf die Grundprinzipien erläutern und dann die ausgewählten Technologien kurz vorstellen.

Ziel dieses Kapitels ist es, damit auch einen Überblick über die unglaubliche Vielfalt von relevanten Technologien, Konzepten, Protokollen und Schnittstellen zu geben und die grundsätzlichen Zusammenhänge zwischen den einzelnen Bereichen transparent zu machen. Dabei kann natürlich im Rahmen dieser Arbeit nicht jedes Konzept bis ins Detail beleuchtet werden. Vielmehr ist beabsichtigt, die jeweiligen Prinzipien in der Form zu erläutern, daß die Bedeutungen der einzelnen Technologien in meinem Gesamtkonzept des *Network Computing* verständlich werden.

Doch können vertiefende Informationen in den jeweils angegebenen Standard-Dokumenten gefunden werden.

5.2.1 Content-Dienste

Präsentation

Der Bereich der Präsentationsdienste ist von zentraler Bedeutung für das Prinzip des **universellen Zugriffs**. Durch die Verwendung von akzeptierten Standards für die Darstellung einer Benutzerschnittstelle kann dieses Prinzip auf der Zugriffsebene verwirklicht werden. Entsprechend Tabelle 5.1 ist die Verwendung von Internet-Standards sinnvoll.

Als Basis für die Darstellung von Informationen verwende ich daher in diesem Referenzmodell eine Identifizierung des Typs des Inhaltes entsprechend des *Multipurpose Internet Mail Extensions* (MIME) [FB96] Konzeptes des `content-type`. MIME-Typen werden nach folgendem Schema angegeben: `Kategorie/Unterkategorie`. Kategorien sind z.B. `text`, `image`, `audio`. Unterkategorien von `text` sind beispielsweise `plain` (Das Dokument ist eine reine Textdatei) oder `html` (Das Dokument ist eine HTML-Datei). Eine Unterkategorie von `image` ist beispielsweise `gif` (Das Dokument ist eine Grafik im GIF-Format).

In Tabelle 5.2 führe ich die Dokumenttypen auf, die mindestens unterstützt werden müssen.

`text/plain`	Darstellung als normaler Text
`text/html`	Darstellung als Hypertextseite entsprechend der Spezifikation der *Hypertext Markup Language* (HTML) [W3C98b].
`image/gif`	Darstellung als Grafik im *Graphics Interchange Format* (GIF) [Com90].

`image/jpeg`	Darstellung als Grafik im *JPEG-Format* (JPEG) [ISO94].
`application/pdf`	Darstellung des Dokuments entsprechend der Spezifikation des *Portable Document Format* (PDF) [BCM96].
`multipart/mixed`	Interpretation der einzelnen Teile der zusammengesetzten Information nach [FB96] und Darstellung entsprechend ihrer jeweiligen Typen.

Tab. 5.2: Inhaltstypen des Präsentationsdienstes

Um andere Dokumenttypen als die hier definierten darstellen zu können, werden diese Typen über einen entsprechenden anderen MIME-Typ identifiziert. Daten dieser Typen können dann über entsprechende Hilfsanwendungen (*Helper Applications*) oder Module (*PlugIn*) interpretiert und dargestellt werden. Im folgenden werde ich einzelne Inhaltstypen und den Zusammenhang zwischen ihnen näher beschreiben.

HTML Für die Definition der Benutzerschnittstelle ist die Seitenbeschreibungssprache *Hypertext Markup Language* (HTML) [W3C98b] der kleinste gemeinsame Nenner aller Endgeräte des *Network Computing* . HTML ist eine Menge von *markup* Symbolen[4], die einem Endgerät beschreiben, wie eine anzuzeigende Seite aufgebaut ist. Sie ist die primäre Sprache für das Publizieren von Hypertext und basiert auf der *Standard Generalized Markup Language* (SGML), dem Standard für die Definition von Dokumentstrukturen. Sie beschreibt eine Seite mit Angaben zu Schriftarten, Farben, Listen, Tabellen, Grafiken und Formularen.

In HTML-Dokumenten können Grafiken über das `<IMAGE>`-Tag integriert werden. Entsprechend Tabelle 5.2 werden GIF- und JPEG-Grafiken unterstützt.

Mittels des `<FORM>`-Tags können einfache Formulare definiert werden, die folgende Interaktionstypen unterstützen:

- Textfelder (auch mehrzeilig),
- Knöpfe (*Buttons*),
- Ankreuzfelder (*Check-Boxes*),
- Radioknöpfe (*Radiobuttons*),
- Listfelder (*Listboxes*),
- Kombinationsfelder (*Drop-Down-Listboxes*).

[4] auch *tags* genannt.

Dabei entspricht die Handhabung dem alten Konzept der *Block-Mode*-Terminals. Es kann immer nur eine komplette Seite an den Server übertragen und erst dort ausgewertet werden. Diese Einschränkung kann durch den Einsatz von *JavaScript* umgangen werden.

JavaScript ist eine plattform-unabhängige, objektorientierte Skriptsprache. Sie besteht aus einer Menge von Basisobjekten wie Zahlen oder Zeichenketten und umgebungsabhängigen Objekten. Auf der Zugriffsebene sind dies Objekte wie Fenster oder Dokumente. Über diese Skriptsprache kann eine Auswertung einzelner Felder bereits im Endgerät erfolgen. Diese Skripte können über das Tag <SCRIPT> in HTML-Dokumente eingebettet werden. Mit ihnen sind zum Beispiel einfache Feldprüfungen zu realisieren. In Abbildung 5.3 ist eine solche Prüfung für ein Postleitzahlfeld dargestellt.

```
<SCRIPT LANGUAGE="JavaScript">

function validatePLZ(textfield) {
    plzOK = false;  // plzOK is a GLOBAL variable
    // Check that the PLZ is the right length
    if ( textfield.value.length != 5 ) {
        alert("Postleitzahlen müssen exakt fünf Ziffern haben!");
        textfield.focus();
        textfield.select();
        return false;
     }
     // don't accept non-numeric characters
     if (!numbersOnly(textfield.value)) {
        alert("Postleitzahlen bestehen nur aus Ziffern!");
        textfield.focus();
        textfield.select();
        return false;
     }
     return (plzOK=true);
}

</SCRIPT>

<FORM>
...
<INPUT TYPE="TEXT" NAME="PLZ" SIZE="5" MAXLENGTH="5" onChange="validatePLZ(this)">
...
</FORM>
```

Fig. 5.3: ECMA Script Prüfung eines PLZ-Feldes

Mittels JavaScript kann jedoch deutlich mehr als nur die Validierung von Feldinhalten durchgeführt werden. Es können der Inhalt, die Struktur und die Präsentation von Dokumenten über Manipulation der entsprechenden Objekte dynamisch verändert werden.

Es gibt mehrere Varianten von JavaScript, die durch die verschiedenen Browser-Hersteller entstanden sind. *ECMA Script* [ECM98] ist der aktuelle Standard von JavaScript und basiert auf der JavaScript Version 1.1. Um eine Herstelleru-

nabhängigkeit zu gewährleisten, sollten JavaScript-Anwendungen auf der Basis von ECMA Script erstellt werden.

Falls die Möglichkeiten von HTML und JavaScript für die Visualisierung des Content nicht ausreichen, könnten noch *Java Applets* eingesetzt werden.

Im Rahmen von HTML können über das `<APPLET>`-Tag *Java Applets* eingebettet werden, die in der *Java Virtual Machine* (JVM) des Endgerätes ausgeführt werden. Das Konzept der *Java Applets* wird in Abschnitt 5.2.3 näher erläutert. Java Applets sind die flexibelste Möglichkeit zur Definition von Bedienerschnittstellen im Rahmen des *Network Computing* . Im *Abstract Windowing Toolkit* (AWT) der Programmiersprache Java sind eine Reihe von klassischen Interaktionstypen definiert, die im Rahmen von Java Applet-Oberflächen verwendet werden können.

Abbildung 5.4 zeigt beispielhaft die Einbindung eines Java Applets für die Darstellung eines drei-dimensionalen Chat-Raums. Die einzelnen Parameter des Applets werden über Name/Wert-Paare an das Applet übergeben.

```
<APPLET CODE = "blacksun.clife.ui.bsiCG1.class" WIDTH=120 HEIGHT=350>
<PARAM NAME = imageBasePath VALUE = "http://www.bacardi.de/night/bar/images/">
<PARAM NAME = imageNumber VALUE = "1">
<PARAM NAME = firstImage VALUE ="1">
<PARAM NAME = image1 VALUE = "control.gif">
</APPLET>
```

Fig. 5.4: Einbettung eines Java Applets in ein HTML-Dokument

XML Die *Extensible Markup Language* (XML) [W3C98a] ist ein textbasiertes Format für strukturierte Dokumente. Ähnlich wie bei HTML werden auch hier *Markup*-Symbole (*Tags*) verwendet. Allerdings erlaubt XML die Definition eigener Tags und bietet eine Metasprache namens *Document Type Definition* (DTD) zur Definition eigener *Markup*-Sprachen für spezielle Anwendungsbereiche, die durch alle XML-fähigen Systeme[5] verstanden werden können.

Anders als bei HTML wird durch XML nicht die Darstellung der Informationen beschrieben. Dies erfolgt durch eine Transformation von XML-Dokumenten in eine andere Form, normalerweise HTML. Diese Transformation wird durch sogenannte *style sheets* vorgegeben, die in einer *Extensible Style Language* (XSL) formuliert werden. XML ist wie HTML eine Untermenge der SGML.

In Abbildung 5.5 ist als Beispiel die XML-Repräsentation einer vCard [DH98a] entsprechend der DTD in [DH98b] dargestellt.

XML ist damit eine flexible Möglichkeit, Informationsstrukturen zu definieren und diese Strukturen sowie die in ihnen enthaltenen Daten gemeinsam zu nutzen. . Durch sie kann ein Teil der in Abschnitt 4.1.3 geforderten fachlichen Standardisierung erfolgen. Daher wird XML meiner Ansicht nach künftig HTML zumindest für den Teil des Content ablösen, der auf dem Austausch von Geschäftsdaten basiert.

[5] Zur Zeit bietet nur der Internet Explorer 4.0 eine XML Unterstützung. Netscape will dies mit der Navigator Version 5.0 nachholen.

```
<vCard version="3.0">
<fn>Karim H. Attia</fn>
<n>  <family>Attia</family>
     <given>Karim</given>
<email email.type="INTERNET">attia@acm.org</email>
</vCard>
```

Fig. 5.5: XML-Repräsentation einer vCard

PDF Für alle anderen Arten von Dokumenten wähle ich das *Portable Document Format* (PDF)[BCM96] von Adobe[6]. Es ist ein neutrales, plattformübergreifendes Datenformat, welches Daten unterschiedlichen Typs (z.B. Texte und Grafiken) enthalten kann. Mit Hilfe eines sogenannten *Acrobat Reader*s werden die Daten visualisiert und Suchfunktionen bereitgestellt. Dieses Format ist als Dokumentformat zwar durch einen Hersteller definiert, jedoch allgemein akzeptiert[7] und offen verfügbar[8]. Viele der klassischen Publikationen stehen bereits als PDF-Version online zur Verfügung[9]. Weitere Gründe für das PDF-Format sind folgende Eigenschaften von PDF-Dokumenten: Sie sind

- plattform-übergreifend - PDF-Anzeigeprogramme sind für viele Betriebssysteme verfügbar[10].
- druckbar - PDF basiert auf Postscript. Dies ermöglicht den exakten Ausdruck auf einer Vielzahl von Druckern.
- klein - PDF-Dateien sind meist kleiner als vergleichbare Dateien in anderen Formaten.

Für das *Network Computing*-Referenzmodell wähle ich für den Bereich der Präsentationsdienste HTML, MIME, XML und PDF sowie das AWT, GIF und JPEG als technologische Plattform.

Persistenz

Der Bereich der Persistenz im Rahmen des *Network Computing* dient vor allem der Verwaltung des Content in all seinen unterschiedlichen Ausprägungen. Dazu gehören auch Multimediadaten oder Dokumente, z.B. im HTML- oder XML-Format. Es werden standardisierte Mechanismen für die Ablage und Wiedergewinnung des Content zur Verfügung stehen und zwar auch im Sinne der **verteilten Verarbeitung** unter Nutzung von Systemen der Domänenebene.

[6] `http://www.adobe.com`

[7] Laut Adobe nutzen bereits über 20 Millionen Menschen den Acrobat Reader.

[8] Diese Arbeit wird über einen TEX-Konverter im PDF-Format zur Verfügung stehen.

[9] So stellt Panasonic (`http://www.panasonic.de`) für seine Office-Produktpalette alle Prospekte online als PDF-Version zur Verfügung. Die Stadt Oberursel (`http://www.oberursel.de`) bietet ihren Bürgern alle gängigen Formulare im PDF-Format, und die italienische Tageszeitung *La Stampa* präsentiert unter `http://www.lastampa.it/Dayfax/dayfax.html` eine 3-Seiten- Version der Zeitung im Internet.

[10] z.B. auch über Ghostscript, einem Freeware Postscript-Interpreter

Entsprechend Tabelle 5.1 ist die Verwendung von Standards aus den Bereichen Zentralrechner und Client/Server sinnvoll.

Aus diesesm Grund wähle ich als Basis der Persistenzdienste die *Structured Query Language* (SQL) [ISO92]. Diese ist meiner Ansicht nach als einziger Standard relevant. Der Standard der *Object Database Management Group* (ODMG 2.0) [Cat97] hat sich bisher nicht im Bereich betrieblicher Informationssysteme etablieren können, und der *Persistent Object Service* (POS) [OMG97b] der OMG ist offiziell gescheitert[OMG97d, App98]. Der Nachfolger *Persistent State Service* (PSS) [OMG98b] ist noch nicht in einem Stadium, welches eine Einschätzung erlaubt[11].

Außerdem nutzen die meisten aktuellen Top-eCommerce Angebote sehr erfolgreich relationale Datenbanksysteme. So basiert *Amazon.com* (`http://www.amazon.com`) mit über 3,5 Millionen Kunden auf einer Oracle-Datenbank, das *Charles Schwab*-Online Brokersystem (`http://www.schwab.com`) mit über sechs Millionen Seitenabrufen pro Tag auf DB2 Datenbanken von IBM [Sey98].

Es gibt zwei grundsätzliche Verfahren für die Nutzung von SQL. Eines basiert auf einer Aufrufschnittstelle: Die *SQL Access Group* (SAG) - jetzt Teil des X/Open Konsortiums - definiert ein solches unabhängiges und einheitliches *Call Level Interface*[12] (CLI) für den Zugriff auf SQL Datenbanken. Dieses Verfahren erlaubt die Anwendung von dynamischen SQL. Der zweite Ansatz bettet die SQL Kommandos in die Programmiersprache ein[13] und übersetzt diese durch einen Precompiler in Aufrufe eines CLI. Dies wird *Embedded SQL* (ESQL) genannt und steht für die Nutzung von statischen SQL zur Verfügung.

ODBC Die *Open Database Connectivity* [Mic97] ist eine erweiterte Version der ISO/IEC [ISO95a] und X/Open [X/O95a] CLI Standards, die zum Zeitpunkt der Veröffentlichung der ODBC Version 1.0 im Jahr 1992 noch nicht verabschiedet waren. Auch wenn ODBC ursprünglich für Microsoft-Umgebungen entwickelt wurde, ist ODBC mittlerweile auch für Apple Macintosh und verschiedene UNIX Systeme verfügbar[14] und damit ein Standard für den Zugriff auf Datenbanksysteme. Die Architektur der ODBC-Schnittstelle ist in Abbildung 5.6 dargestellt.

Die zentralen ODBC-Komponenten sind der *Driver Manager* und die *Driver*. Sie dienen als Bindeglieder zwischen der Anwendung und einem oder mehreren Datenbanksystemen. Beide Komponenten befinden sich grundsätzlich auf dem System der Anwendungsebene und werden jeweils als *Dynamic Link Library* (DLL) dynamisch zur Anwendung hinzugebunden. Der *Driver Manager* ist für die Anwendung völlig transparent und übernimmt das dynamische Laden von Datenbank-Treibern (*Driver*). Datenbank-Treiber werden vom *Driver Manager*

[11] Allerdings wurde der entsprechende Vorschlag gemeinsam von Ardent, IBM, INPRISE, IONA, Objectivity Object Design, Oracle, Persistence, Sun, Versant und anderen eingereicht, was hoffen läßt, zudem durch die Reduzierung des Funktionsumfangs die Realisierbarkeit wahrscheinlicher ist, als es beim POS der Fall war.

[12] Der Begriff *Call Level Interface* (CLI) ist historisch entstanden und bezeichnet eigentlich ein *Application Programming Interface* (API) - in der Datenbankwelt ist die API jedoch SQL selbst.

[13] über das Schlüsselwort `EXEC SQL`.

[14] etwa über Hersteller wie Intersolv oder Visigenic.

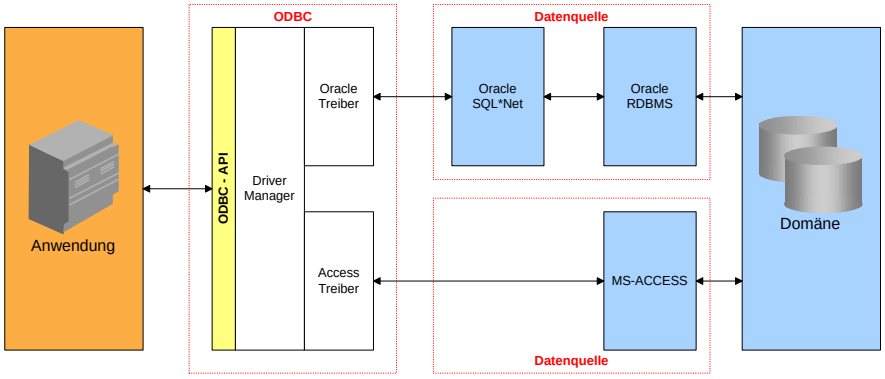

Fig. 5.6: ODBC Architektur

geladen, wenn eine Verbindung zu einer Datenbank hergestellt werden soll. Der Treiber übernimmt dann den Verbindungsaufbau zu den Datenquellen (*Data Source*), die Übermittlung der Anfragen und zugehörigen Ergebnisse, die Übersetzung der Datenformate und die Formatierung von Fehlercodes. Nachteil bei der Verwendung von ODBC ist allerdings, daß durch die vielen Schichten ein Verwaltungsoverhead entsteht, der die Anwendung verlangsamt.

JDBC Für die Nutzung von Datenbanksystemen durch die Programmiersprache Java[15] wurde von SUN die *Java Database Connectivity* (JDBC) [Sun97b] entwickelt. JDBC basiert auf den Konzepten von ODBC, ist jedoch objektorientiert. Ein JDBC-Treiber muß mindestens die `java.sql` Interfaces und Klassen implementieren und dabei mindestens den *ANSI SQL92 - Entry Level* an Funktionalität bereitstellen.

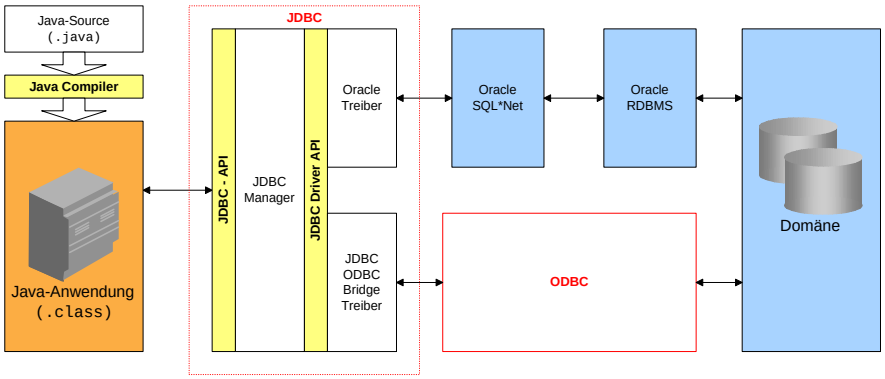

Fig. 5.7: JDBC Architektur

Wie in Abbildung 5.7 dargestellt, kann über eine *JDBC-ODBC-Bridge* die Anbindung beliebiger ODBC-fähiger Datenbanken an JDBC und damit an Java er-

[15] siehe Abschnitt 5.2.3.

reicht werden. Eine entsprechende URL[16] wäre dann `jdbc:odbc://as.xenion.de/OraDB`, also der Zugriff auf eine Datenbank auf dem Server `as.xenion.de` über ODBC und anschließend über die *JDBC-ODBC-Bridge*.

SQLJ Der Standard für *Embedded-SQL* für Java ist SQLJ [ISO98]. Dieser bietet eine einfache Schnittstelle für die Integration von statischem SQL in die Programmiersprache Java. In Abbildung 5.8 ist das Konzept einer solchen SQLJ-Anwendung dargestellt.

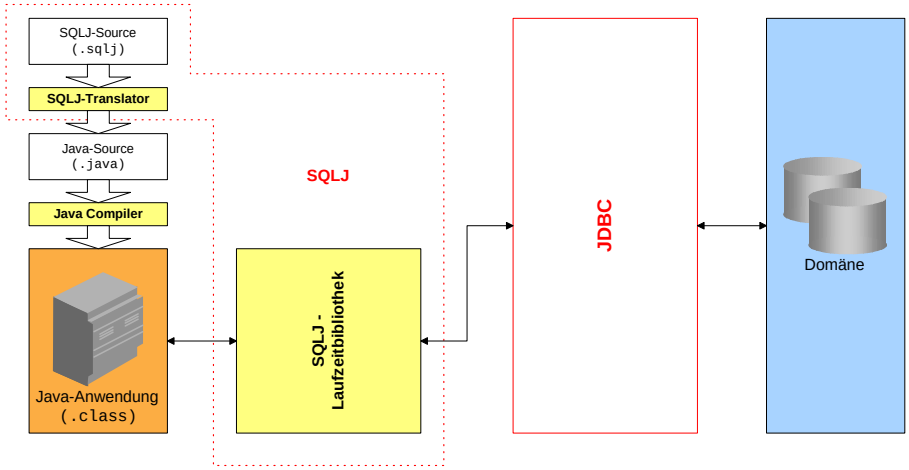

Fig. 5.8: SQLJ Architektur

Ein Precompiler übersetzt die SQLJ-Kommandos - gekennzeichnet durch das Schlüsselwort `#sql` - in Aufrufe der SQLJ-Laufzeitbibliothek. Diese nutzt die JDBC-Schnittstelle für die Abwicklung der Anfragen an das Datenbanksystem - als *host-variablen* sind alle JDBC-Datentypen zulässig.

Abbildung 5.9 zeigt einen einfachen Vergleich von SQLJ und JDBC. Abhängig von den Parametern `y` und `z` werden die Werte `w` und `x` aus der Tabelle `TAB` gelesen. Wie zu erkennen, ist die SQLJ-Variante deutlich einfacher und bietet bereits zur Compile-Zeit eine Typ-Prüfung der Host-Variablen. Allerdings ist nur die Verwendung von statischen SQL möglich.

Als Standards für die Persistenzdienste wähle ich daher ODBC, JDBC und SQLJ.

5.2.2 Communication-Dienste

Transport

Der Transportdienst bildet die Grundlage des **universellen Zugriffs** auf die Anwendungen der Digitalen Wirtschaft. Entsprechend Tabelle 5.1 ist auch hier

[16] siehe Abschnitt 5.2.2.

```
float w; java.sql.Date x; int y; String z;

// ***** SQLJ *****

#sql { SELECT C1, C2 INTO :w, :x FROM TAB WHERE C3 = :y AND C4 = :z };

// ***** JDBC *****

PreparedStatement s = connection.prepareStatement
("SELECT C1, C2 FROM TAB WHERE C3 = ? AND C4 = ?");
s.setInt(1, y);
s.setString(2, z);
ResultSet r = s.executeQuery();
r.next();
w = r.getFloat(1);
x = r.getDate(2);
r.close();
s.close();
```

Fig. 5.9: Vergleich von SQLJ und JDBC

die Verwendung von Internet-Standards sinnvoll.

TCP/IP Basis jeder Kommunikation im Rahmen des *Network Computing* ist die Transportprotokollsuite TCP/IP [IET81b, IET81a, IET80]. Sie besitzt eine Reihe von Eigenschaften, die sie als Grundlage für das Konzept des **universellen Zugriff** des *Network Computing* prädestiniert:

- Das Protokoll ist unabhängig von bestimmten physikalischen Netzwerkeigenschaften. Es kann unter anderem über Ethernet oder ISDN eingesetzt werden.
- Der Standard ist frei verfügbar und unabhängig von bestimmten Hard- und Softwareplattformen und -herstellern und damit ideal für die Verbindung unterschiedlichster Systeme geeignet.
- Das einheitliche Adressierungsschema, die IP-Adresse, erlaubt die eindeutige Identifizierung eines jeden TCP/IP-Systems, selbst im weltweiten Internet[17].

TCP/IP ist ein Protokoll-Stack, also eine Struktur, bei der mehrere Protokolle übereinander gestapelt werden. In der Spezifikation werden vier Schichten[18] beschrieben, die den TCP/IP-Protokoll-Stack ausmachen. In Abbildung 5.10 sind diese vier Schichten dargestellt:

[17] Auch wenn durch den Boom des World Wide Web hier der Adreßraum langsam knapp wird.

[18] Im Rahmen des ISO/OSI 7-Schichtenmodells [Ker89] deckt TCP/IP ungefähr die Aufgaben der Schichten eins bis vier ab. Allerdings werden für die Sicherungsschicht (*data link layer*) und die Bitübertragungsschicht (*physical layer*) existierende Standards verwendet, Aufgaben der Darstellungschicht (*presentation layer*) werden zum Teil über das MIME-Protkoll erfüllt, und Teile der Aufgaben der Kommunikationssteuerungsschicht (*session layer*) von der TCP/IP-Transportschicht übernommen.

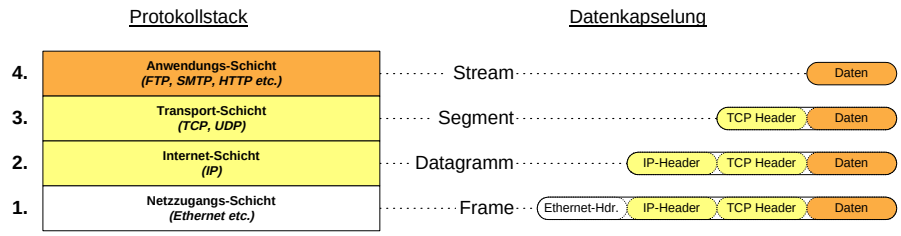

Fig. 5.10: TCP/IP Protokollstack und Datenkapselung

Die *Netzzugangs-Schicht* enthält dabei die Routinen für die Nutzung physikalischer Netzwerke wie Ethernet oder ISDN und die Übertragung der Rahmen (*frames*). Die *Internet-Schicht* übernimmt unter Verwendung des *Internet Protocol* (IP) das Routen der Datenblöcke (*datagram*) zu ihren Zielendpunkten. Die *Transport-Schicht* stellt die End-zu-End-Verbindungen zwischen Absender und Empfänger bereit. Die Transportschicht bietet unter Nutzung des *Transmission Control Protocol* (TCP) ein zuverlässiges verbindungsorientiertes Bytestream-Protokoll bzw. bei der Nutzung des *User Datagram Protocol* (UDP) ein unzuverlässiges verbindungsfreies, aber effizienteres Protokoll[19]. Auf der Transportschicht sitzt die eigentliche *Anwendungs-Schicht*, auf der sich Anwendungsprotokolle wie SMTP, FTP oder HTTP befinden, die sich der Dienste der Transportschicht bedienen.

Jede einzelne Schicht fügt nun den an sie übergebenen Daten Kontrollinformationen in Form eines *Headers* hinzu. Dabei betrachtet jede Schicht die gesamten, von der oberen Schicht empfangenen Informationen als Daten und stellt diesen ihren eigenen Header voran.

TCP/IP ist auch dafür verantwortlich, daß die von IP übernommenen Daten an die richtige Anwendung ausgeliefert werden. Die Anwendung, für die die Daten gedacht sind, wird über eine 16-Bit-Zahl identifiziert, die als *Port*-Nummer bezeichnet wird. Die folgenden zwei Protokolle HTTP und IIOP bilden, aufbauend auf TCP/IP, die Grundlage für die generelle Kommunikation zwischen Zugriffs- und Anwendungsebene.

HTTP Das *Hypertext Transfer Protokoll* (HTTP) [FGM+97, BLFF96] ist ein generisches, zustandsloses Anwendungsprotokoll für verteilte Hypermedia Informationssysteme. Es nutzt das Konzept des *Uniform Resource Locator*[20] (URL) für die Lokalisierung einer Ressource, auf die eine Methode angewendet werden soll. Die Nachrichten werden dabei in einem Format entsprechend der *Multipurpose Internet Mail Extensions*[21] (MIME) übertragen.

In Abbilding 5.11 ist eine solche HTTP Client/Server-Interaktion dargestellt. Sie ist eine Art von *Remote Procedure Call* (RPC) und besteht aus einem einfachen *Request/Response* Austausch. Aufgrund des zustandslosen Charakters wird für

[19] Für eine genaue Diskussion der Unterschiede sei auf die jeweiligen Standard-Dokumente verwiesen.
[20] siehe Abschnitt 5.2.2.
[21] siehe Abschnitt 5.2.1.

jede Interaktion eine TCP/IP-Verbindung aufgebaut, der Request vom Client zum Server geschickt, die Response vom Server zurück zum Client übermittelt und dann die TCP/IP-Verbindung wieder abgebaut.

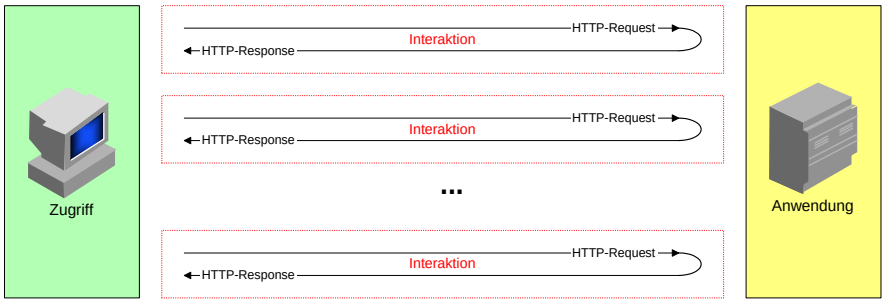

Fig. 5.11: HTTP Interaktionsmodell

Dies hat den Vorteil, daß die Server sehr effizient arbeiten können. Ein Problem ist allerdings, daß es zwischen den einzelnen Requests keine Verbindung gibt. Da die Verbindung von einzelnen Requests zu einer Session jedoch für die verschiedensten Anwendungsbereiche[22] wichtig ist, bieten sich mehrere unterschiedliche Konzepte an, dieses Manko zu umgehen:

- *Cookies* [KM97] - Mittels Daten, die lokal auf den Arbeitsstationen der Zugriffsebene abgelegt und bei jedem Request an den Server der Anwendungsebene geschickt werden, der ihre Speicherung initiiert hat, kann über eine selbstdefinierte Session-ID die Verbindung zwischen den einzelen Requests zu einer Session hergestellt werden. Allerdings haben *Cookies* einen schlechten Ruf, da mit ihnen immer die Möglichkeit einer Ausspionierung der einzelnen Anwender verbunden wird. Die meisten Browser bieten daher die Möglichkeit, eine solche Speicherung zu verhindern. Aus diesem Grund sollten Cookies nicht verwendet werden.
- *URL-Encoding* - In dieser Variante wird die Session-ID als Teil der URL[23] mitgeführt. Indes eignet sich dieses Verfahren selbstverständlich nur bei dynamisch erzeugten HTML-Dokumenten und hat außerdem den Nachteil, daß die Session-ID für jedermann sichtbar ist.
- *Hidden Fields* - Diese dritte Alternative versteckt die Session-ID in einem unsichtbaren HTML-Feld. Auch diese Variante eignet sich nur bei dynamisch erzeugten HTML-Dokumenten, die zusätzlich zumindest ein Formular enthalten müssen. Allerdings ist die Session-ID nun nicht mehr offensichtlich, sondern vom Anwender nur über den Dokumentquelltext zu ermitteln.

Ein HTTP-Request besteht aus einer *Request Line*, mehreren optionalen *Request Header Fields* und einem optionalen *Entity Body*. In Abbildung 5.12 ist ein einfacher Request dargestellt:

[22] wie z.B. auch für requestübergreifende Transaktionen.
[23] z.B. bei der Suchmaschine `http://www.web.de`.

Syntax

```
<method><resource identifier><HTTP version><crlf>          request line
[<Header> : <value><crlf>]                                 request header
      ...                                                  fields
[<Header> : <value><crlf>]
<crlf>
[Entity Body]                                              entity body
```

Beispiel

```
GET /external/bacardi/index.html HTTP/1.0                  request line
User-agent: Mozilla/3.01 [de] (Win95; I)                   request header
Referer: http://www.xenion.de/external/index.html          fields
```

Fig. 5.12: HTTP Request

Das Beispiel beschreibt einen HTTP-Request, der mittels der Methode `GET` die Datei `/external/bacardi/index.html` vom Server verlangt, unter Nutzung des Protokolls `HTTP/1.0`. Desweiteren übermittelt dieser Request die Information, daß er von einem Browser (*User Agent*) namens `Mozilla/3.01 [de] (Win95; I)`[24] initiiert wurde und zwar ausgehend (*Referer*) von einer HTML-Seite mit der URL `http://www.xenion.de/external/index.html`. Die hier dargestellte Methode `GET` ist eine der zwei wichtigsten HTTP-Methoden:

- *GET* - Hole die Ressource mit der angegebenen URL vom Server.
- *POST* - Sende die Daten des *Entity Body* an den Server.

Allerdings sollte von der Verwendung von `GET` als HTTP-Methode zum Versand der Feldinhalte eines HTML-Formulars abgesehen werden. Denn im Gegensatz zu `POST` werden bei einem `GET` die Parameter in der Form `?parameter1=wert1&parameter2=wert2...` an die URL angehängt. Dies kann bei der Verwendung des *Common Gateway Interface*[25] (CGI) zu Problemen mit der maximalen Länge von Umgebungsvariablen führen. Die Parameterliste wird nämlich der CGI-Umgebungsvariable `query_string` zugeordnet, die je nach Betriebssystem eine Länge zwischen 256 und 1024 Zeichen haben kann. Bei Verwendung von `POST` hingegen wird die Parameterliste an die HTML-Nachricht angehängt und über die Standardeingabe (STDIN) an das CGI-Programm übergeben.

Analog zu einem HTTP-Request besteht eine HTTP-Response aus einer *Status Line*, mehreren optionalen *Header Fields* und einem optionalen *Entity Body*.

Dieses Beispiel beschreibt eine HTTP-Response eines Servers `Apache/1.2` über das Protokoll `HTTP/1.0` mit dem Statuscode `200`, dessen Bedeutung `OK` ist. Als Beschreibung des Entity Body wird der `Content_type: text/html` angegeben und anschließend die HTML-Seite übermittelt.

[24] Ein Netscape 3.01 Browser in der deutschen Version unter MS-Windows 95.
[25] siehe Abschnitt 5.2.3.

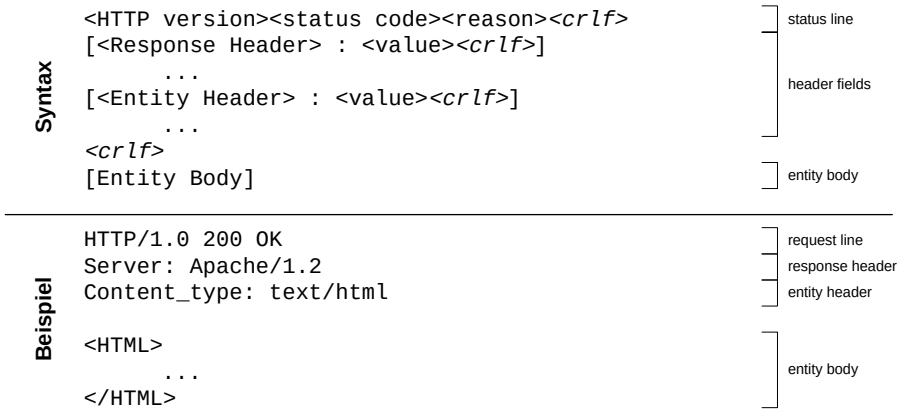

Fig. 5.13: HTTP Response

Die weiteren Methoden, Statuscodes und *Header Fields* sind der aktuellen Spezifikation [FGM+97] zu entnehmen. Ein Vorteil von HTTP ist, daß ähnlich wie bei HTML unbekannte Schlüsselworte in den Response/Request-Daten einfach ignoriert werden. Dies ermöglicht die Einführung neuer *Header Fields*, ohne daß bestehende Anwendungen beeinflußt werden. Neben dem fehlenden Session-Konzept ist die Parameterübergabe mittels URL oder *Entity Body* ein Grund für Probleme bei der Anwendung von HTTP im Rahmen von komplexen *Network Computing*-Anwendungen.

IIOP Bei Anwendungen, die über den Einsatz von HTML und CGI hinausgehen und dafür die Programmiersprache Java [26] nutzen, bietet sich die Verwendung von *Objekt Request Broker*[27] (ORBs) und des zugehörigen *Internet Inter-ORB Protocol* (IIOP) an. Hierbei wird die Kommunikation zwischen Zugriffs- und Anwendungsebene über miteinander kommunizierende ORBs abgewickelt. Der aktuelle Standard für ORBs ist die *Common Object Request Broker Architecture* (CORBA) [OMG98c] der *Object Management Group* (OMG).

Für die Kommunikation zwischen verteilten ORBs definiert die OMG ein abstraktes *General Inter-ORB Protocol* (GIOP) [OMG98c]. Es beschreibt eine Standard-Transfersyntax[28] und sieben Nachrichtenformate, die die komplette ORB-Request/Reply-Semantik definieren. Die CDR beschreibt die Transformation der Datentypen der OMG *Interface Definition Language* (IDL) in ein Format zur Übertragung der Daten. Die sieben Nachrichtenformate sind in Abbildung 5.3 aufgeführt.

`Request`	Ein Client greift mittels eines `Request` auf Attribute oder Methoden eines CORBA-Objektes zu.

[26] siehe Abschnitt 5.2.3.
[27] siehe Abschnitt 3.2.4.
[28] *Common Data Representation* (CDR)

`Reply`	`Reply`-Nachrichten werden von einem Server als Antwort auf einen `Request` an den Client geschickt. Sie enthalten die Parameter, das Ergebnis und evtl. vorhandene Ausnahmewerte.
`CancelRequest`	informiert einen Server, daß ein Client nicht mehr auf ein `Reply` für ein `Request` oder `LocationRequest` wartet.
`LocateRequest`	dient der Ermittlung, : (a) ob eine Referenz auf ein CORBA-Objekt gültig ist, (b) ob der angesprochene Server in der Lage ist, `Requests` an das referenzierte CORBA-Objekt weiterzuleiten oder, falls nicht, (c) an welche Adresse ein `Request` an das referenzierte CORBA-Objekt zu schicken ist.
`LocateReply`	liefert die Antwort des Servers auf ein `LocateRequest`.
`CloseConnection`	informiert die Clienten eines Servers, daß die Verbindung vom Server geschlossen wird. Alle ausstehenden `Requests` werden nicht mehr erfüllt und müssen erneut angefordert werden.
`MessageError`	informiert das jeweilige Gegenüber einer Kommunikation, daß die letzte Nachricht nicht verstanden werden konnte.

Tab. 5.3: GIOP Nachrichtenformate

IIOP ist eine konkrete Realisierung des abstrakten *General Inter-ORB Protocol* (GIOP) der OMG auf der Basis von TCP/IP.

Allerdings ist der Einsatz von IIOP über eine Firewall[29] problematisch. Die meisten eingesetzten Firewall-Systeme sind nicht dafür eingerichtet, IIOP-Requests zu interpretieren und entsprechend zu behandeln. Daher gibt es im wesentlichen zwei Ansätze, um auch IIOP-Requests über Firewalls hinweg zu übertragen:

- Einbettung des IIOP-Request in einen HTTP-Request (*Tunneling*)[30] - Vorteil dieser Variante ist, daß dieses Verfahren mit allen gängigen Firewallsystemen zusammenarbeitet, jedoch proprietäre Kodierungsschichten auf den beteiligten Systemen vorhanden sein müssen.
- Nutzung eines IIOP-Proxies in der Firewall[31] - Diese Firewall ist in der Lage, den IIOP-Request zu interpretieren. Das jeweilige Proxy-Objekt prüft dann anhand einer *Access Control List* (ACL) die Berechtigung des Zugriffs und leitet diesen an das eigentliche Objekt weiter. Vorteil dieser Variante ist die Verwendung von reinem IIOP ohne Tunneling und damit eine höhere Effizienz des Zugriffs.

[29] siehe Abschnitt 5.2.2.
[30] z.B. Visigenic's GateKeeper.
[31] z.B. Orbix Wonderwall.

Im Rahmen des *Network Computing*-Referenzmodells wähle ich daher für den Bereich der Transportdienste die Protokolle TCP/IP, HTTP und IIOP als technologische Plattform.

Kollaboration

Durch Teamarbeit in Gruppen entsteht Information mit überwiegend geringem Strukturierungspotential. Die Arbeit selbst findet oft in relativ unstrukturierten Prozessen statt. Eine Eigenschaft der hier anfallenden Informationen ist, daß sie meist nur durch zugehörige Anwendungsprogramme (z.B. Tabellenkalkulation, Präsentationen) oder durch einen Menschen (z.B. elektronische Briefe, Nachrichten in elektronischen Diskussionsforen) interpretiert werden können. Die Kollaborationsdienste müssen dafür die asynchrone und synchrone Kommunikation zwischen Einzelpersonen oder Gruppen realisieren können.

Entsprechend der Tabelle 5.1 bieten sich auch hier Standards aus dem Bereich des WWW an. Ich wähle folgende Dienste für die verschiedenen Ausprägungen der Kollaboration:

- elektronische Post (*eMail*) : SMTP/POP3
- Diskussionsforen (*NetNews*) : NNTP
- Dateiübertragung (*Filetransfer*) : FTP
- Tele-Konferenzen (*Chat*) : IRC

Im folgenden beschreibe ich diese Standards nun näher:

SMTP/POP3 Das *Simple Mail Transfer Protokoll* (SMTP) [Pos82] ist das Protokoll zur Auslieferung von eMails. Es nutzt den zuverlässigen, verbindungsorientierten Dienst des TCP[32] zur Auslieferung der eMail in einem Punkt-zu-Punkt-Verfahren[33]. Die direkte Auslieferung erlaubt es, Mail auszuliefern, ohne von dazwischengeschalteten Rechnern abhängig zu sein. Anhand des *Domain Name Systems*[34] [35] kann der lokale Mailer feststellen, welches Zielsystem für die Auslieferung der Mail gewählt werden muß[36]. Die wichtigsten Befehle des SMTP sind in Tabelle 5.4 angegeben.

`HELO <Absendersystem>`	Identifiziert das sendende System
`MAIL FROM: <Absenderadresse>`	eMail-Adresse des Absenders
`RCPT TP: <Empfängeradresse>`	eMail-Adresse des Empfängers
`DATA`	Beginn der Nachricht
`RSET`	Abbrechen der Nachricht
`QUIT`	Beenden der Verbindung

Tab. 5.4: SMTP-Kommandos

[32] Port 25
[33] anders als z.B. UUCP oder X.400, die nach dem *Store-and-Forward*-Prinzip arbeiten.
[34] siehe Abschnitt 5.2.2.
[35] über den `MX`-Eintrag der Domäne.
[36] Es gibt meist mehrere, mit unterschiedlichen Prioriäten versehene Zielsysteme für eine Maildomäne, um den Ausfall eines Systems überbrücken zu können.

Das *Post Office Protocol - Version 3* (POP3)[37] [MR96] wird verwendet, um die Nachrichten von dem Postfach des Zielssystems zum Arbeitsplatz des Empfängers zu übermitteln[38]. Die wichtigsten POP3-Kommandos sind in Tabelle 5.5 aufgelistet.

`USER <Benutzername>`	Identifiziert den Empfänger
`PASS <Kennwort>`	Authentisiert den Empfänger
`STAT`	Fragt nach der Anzahl ungelesener Nachrichten
`RETR <n>`	Empfängt Nachricht ¡n¿
`DELE <n>`	Löscht Nachricht ¡n¿
`LAST`	Fragt nach der letzten abgefragten Nachricht
`QUIT`	Beendet die Verbindung

Tab. 5.5: POP3-Kommandos

Die Struktur der übertragenen Nachrichten entspricht dabei dem bereits im Abschnitt 5.2.1 beschriebenen MIME-Format [FB96] .

NNTP Das *Network News Transfer Protocol* (NNTP)[39] [KL86] spezifiziert ein Protokoll für die Verteilung, die Abfrage und das Versenden von Artikeln für elektronische schwarze Bretter. Diese Artikel werden in einer zentralen Datenbank gespeichert. Ein Anwender braucht so nur die Artikel von dem zentralen Server zu abrufen, die ihn interessieren. Außerdem werden Möglichkeiten zur Indizierung, Querverweise und das Ablaufen von Gültigkeiten geboten. Insgesamt stehen dabei 23 Kommandos für die Verwaltung der Artikel zur Verfügung.

IRC Das *Internet Relay Chat Protocol* (IRC)[40] [OR93] ist ein Tele-Konferenzsystem (*Chat*) auf der Basis eines Client/Server-Modells. Die Server bilden die Grundlage des IRC. Sie sind in einem Netzwerk angeordnet, in dem jeder Server für sich als zentraler Knoten für den Rest des Netzwerkes gilt.

Die Clienten nehmen die Verbindung zu einem Server auf und melden sich an einem Kanal an. Ein Kanal ist eine benannte Gruppe von Clienten, die alle die gleichen Nachrichten erhalten. Der Kanal wird implizit erzeugt, wenn der erste Client ihn betritt und gelöscht, wenn der letzte Client ihn verläßt. Dialoge sind in den Kanälen in Echtzeit möglich, die Verwaltung der Kanäle und der Konferenzteilnehmer wird von den Servern übernommen.

[37] TCP/IP Port 110.
[38] Dies ist nur nötig, wenn der Arbeitsplatz nicht selbst das Zielsystem ist. Dies ist z.B. bei fast allen MS-Windows Arbeitsplätzen der Fall.
[39] TCP/IP Port 119.
[40] in der Regel TCP/IP Port 6667.

FTP Das *File Transfer Protocol* (FTP)[41] [PR85] ist ein Standardprotokoll für die Übertragung von Dateien zwischen Systemen im Internet. Mittels FTP lassen sich auch Dateien auf einem Server aktualisieren[42]. Für die Verwendung eines FTP-Servers muß man sich beim ihm anmelden. Um einen allgemeinen, öffentlichen Zugang zu ermöglichen, gibt es einen anonymen Standardbenutzer. Die wichtigsten FTP-Kommandos sind in Tabelle 5.6 aufgelistet.

`USER <Benutzername>`	Identifiziert den Benutzer
`PASS <Kennwort>`	Authentisiert den Benutzer
`CWD <Verzeichnis>`	Wechselt zum angegebenen Verzeichnis
`RETR <Dateiname>`	Lädt die Datei mit dem angegebenen Dateinamen vom Server
`STOR <Dateiname>`	Speichert die Datei mit dem angegebenen Dateinamen auf dem Server
`DELE <Dateiname>`	Löscht die Datei mit dem angegebenen Dateinamen auf dem Server
`LIST`	Übermittelt eine Liste der Dateien im aktuellen Verzeichnis
`QUIT`	Beendet die Verbindung

Tab. 5.6: FTP-Kommandos

Für die Unterstützung kooperativer Arbeit von Personen im Rahmen der digitalen Wirtschaft wähle ich also SMTP/POP3, NNTP, IRC und FTP als Technologien für das *Network Computing*.

Verzeichnisse

In verteilten heterogenen Umgebungen bedarf es eines Dienstes, um allen Anwendungen über einheitliche Schnittstellen das Lokalisieren von Ressourcen zu ermöglichen. Dies ist eine Grundlage des **universellen Zugriffs**. Im Rahmen des *Network Computing* benötige ich daher einen standardisierten Verzeichnisdienst, der die eindeutige Benennung und Identifikation von Objekten über alle beteiligten Systeme hinweg sichert und nach Tabelle 5.1 eher aus dem Bereich des WWW kommen sollte. Damit eine lokale Autonomie bei der Vergabe von Objektnamen gewährleistet werden kann, gleichzeitig jedoch die weltweite Eindeutigkeit sichergestellt wird, muß eine hierarchische Baumstruktur eingeführt werden. Dieses Konzept wird beim *Domain Name System* verwendet, das die Basis der Lokalisierung von Ressourcen im Internet über deren *Uniform Resource Locator* bildet.

Domain Name System Das DNS [Moc87] ist der Mechanismus zur eindeutigen Benennung von Systemen im Internet über einen hierarchisch strukturierten Namen. Die oberste Ebene wird durch sogenannte Top-Level-Domänen gebildet, die über ein Kürzel ein Land identifizieren[43]. Diesen untergeordnet sind die

[41] TCP/IP Port 21.

[42] verschieben, löschen, umbenennen und kopieren.

[43] mit Ausnahme der historischen Top-Level-Domänen `com`, `org`, `edu` und `mil`, die noch aus der Zeit stammen, als das Internet noch auf den Bereich der USA beschänkt war.

Namen der Organisationen, zu denen die Systeme gehören[44]. Jeder Top-Level-Domäne ist eine Institution zugeordnet, die die Vergabe von Namen unterhalb dieser Domäne überwacht[45].

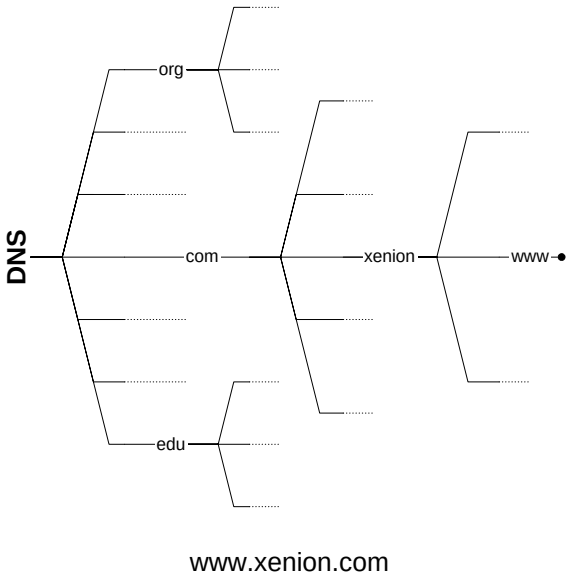

Fig. 5.14: Namensräume im Domain Name System

In Abbildung 5.14 ist ein Ausschnitt aus dem DNS dargestellt, in dem die Zusammensetzung des Namens eines Systems einer Organisation namens XENION unter der Top-Level-Domäne `com` erkennbar ist. Der Name des Systems ist `www.xenion.com`. Das DNS stellt damit ein weltweites Verzeichnis von Systemen im Internet dar.

Uniform Resource Locator Eine URL [BLMM94] lokalisiert eine Ressource im Internet über eine abstrakte Identifikation der Lokation der Ressource. Eine URL hat immer folgende Struktur: `<Schema>:<schemaspezifische Informationen>`. Die schemaspezifischen Informationen beinhalten dabei den DNS-Namen des Systems, das die gewünschte Ressource bereitstellt. URLs sind damit eine spezielle Form der *Uniform Resource Identifiers* (URI) [BLFIM98], bei der die Ressource über ihren primären Zugriffsmechanismus identifiziert wird.

Beispiele für URLs sind:

- `http://www.xenion.de/welcome.html`
  referenziert eine Datei namens `welcome.html`, die über das HTTP-Protokoll von einem System mit dem DNS-Namen `www.xenion.de` bereitgestellt wird.

[44] auch hier gibt es Ausnahmen, wie z.B. in Groß Britannien, wo unter der Top-Level-Domain `uk` noch Domänen wie `co` definiert wurden, die die US-Einteilung nachahmen.
[45] In Deutschland ist dies das DeNIC `http://www.nic.de`.

- `mailto:attia@acm.org`
  initiiert den Versand einer eMail an die Adresse `attia@acm.org` über das SMTP-Protokoll.

Lightweight Directory Access Protocol Für die Lokalisierung von Informationen, die nicht im DNS verwaltet werden, gibt es eine Vielzahl von Verzeichnis-Systemen. Der Standard für derartige Verzeichnisdienste ist der X.500-Standard für *Directory Services* der ITU[ITU97a] . Er definiert eine, in Abbildung 5.15 beispielhaft dargestellte, hierarchisch organisierte Verzeichnisstruktur, ähnlich dem DNS.

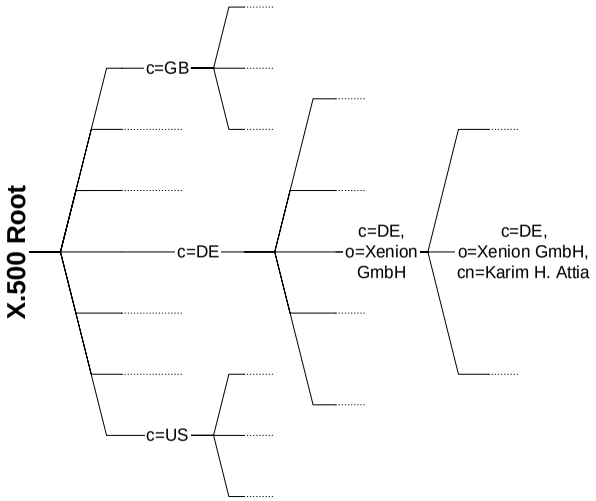

Fig. 5.15: Namensräume im X.500 System

Jedes verwaltete Objekt hat einen *Distinguished Name* (DN) wie z.B. `c=DE, o=Xenion GmbH, cn=Karim H. Attia`, der die Person mit dem *Common Name* (CN) Karim H. Attia innerhalb der Organisation (O) Xenion GmbH im Land (C) Deutschland identifiziert. Dabei kann es sich um reale Objekte wie Personen oder Geräte, aber auch um logische Objekte, wie Benutzergruppen handeln. Zu diesen Objekten werden Attribute verwaltet, wie z.B. ein Name, eine Telefonnummer oder ein Standort.

Ein DNS-Name wie `www.xenion.de` kann auch als X.500 *Distinguished Name* ausgedrückt werden: `o=Internet, dc=de, dc=xenion, dc=www`[46].

Dieser X.500 Standard basiert auf den OSI-Protokollen und stellt daher hohe Anforderungen an die beteiligten Systeme. Aus diesem Grund und auch durch die gestiegene Bedeutung der Internet-Systeme wurde das *Lightweight Directory Access Protocol* (LDAP)[47] [YHK95] entwickelt. Es basiert konzeptionell auf den X.500-Directory Services, bietet aber wesentliche Vereinfachungen:

[46] `dc` ist die Abkürzung für *Domain Component*.
[47] TCP/IP Port 389.

- Als Transportprotokoll wird TCP/IP direkt unterstützt.
- Es werden nur die meistgenutzten Funktionen von X.500 bereitgestellt.
- Die Daten werden in Form von einfachen Zeichenketten übermittelt.

Das LDAP hat sich inzwischen als Standardprotokoll für den Zugriff auf unterschiedlichste Verzeichnisdienste etabliert. Abbildung 5.16 zeigt, wie über die LDAP-Schnittstelle im Sinne des universellen Zugriffs die Nutzung von Verzeichnissystemen verschiedener Hersteller ermöglicht werden kann.

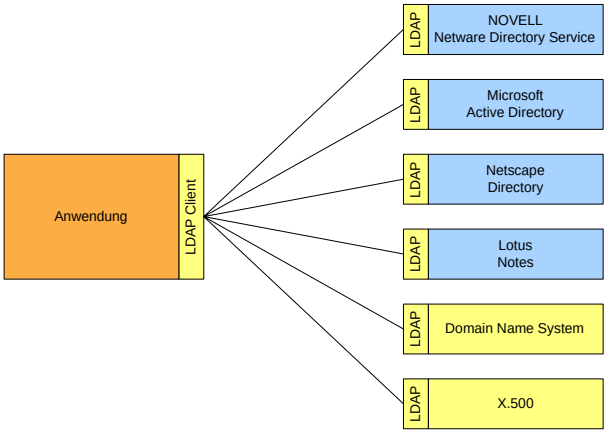

Fig. 5.16: Verzeichnis-Integration durch LDAP

Durch LDAP-fähige Verzeichnisse im Zusammenhang mit den im nächsten Abschnitt beschriebenen persönlichen Zertifikaten für die eindeutige Identifizierung von Personen oder Organisationen ist die Nutzung von stark personalisierten Anwendungen über LDAP-fähige Standardendgeräte denkbar. Die jeweilig relevanten Attribute können dann über die LDAP-Schnittstelle standardisiert ausgetauscht werden. Außerdem ist eine redundante Verwaltung von Benutzerinformationen in heterogenen Systemen nicht mehr nötig. So können zum Beispiel Zugriffskontroll-Listen in LDAP-fähigen Verzeichnissen abgelegt werden.

Auch im Bereich des System-Management können LDAP-fähige Verzeichnisse hilfreich sein. Ein über Standardmechanismen abfragbares Verzeichnis von Druckern oder Servern verbessert etwa die Verwaltung, da auf diese Weise systemübergreifende Verwaltungswerkzeuge eingesetzt werden können.

Im Bereich der Verzeichnisdienste wähle ich daher für das *Network Computing*-Referenzmodell DNS, URL und LDAP als technische Konzepte.

Sicherheit

In einer verteilten Umgebung der digitalen Wirtschaft ist das Thema Sicherheit von enormer Bedeutung. Dabei steht nicht in erster Linie der Aspekt der Ausfallsicherheit im Vordergrund, sondern der Aspekt der mißbräuchlichen Nutzung von Anwendungen und Daten zum Schaden der beteiligten Organisationen und

Personen: Durch den **universellen Zugriff** von jederman von jedem Ort liegt hier ein enormes Gefahrenpotential.

Die zunehmend stärkere Einbindung von Geschäftspartern in die betrieblichen Geschäftsprozesse wirft zwangsläufig die Frage auf, wie betriebliche Ressourcen wirksam gegen unautorisierte oder mißbräuchliche Nutzung geschützt werden können. Beispiele für entsprechende Gefahren sind:

- Personen, die sich als eine andere Person ausgeben, um deren erweiterte Zugriffsprivilegien nutzen zu können,
- die nachträgliche Leugnung[48] von Aktionen oder Zugriffen, falls Personen nicht eindeutig identifiziert werden können,
- das Abhören von Kommunikationsverbindungen, wobei vertrauliche Informationen an unberechtigte Personen gelangen können und
- die Verfälschung von Informationen während der Übertragung vom Sender zum Empfänger.

Im folgenden werde ich eine Auswahl von wichtigen Konzepten beschreiben, die die Sicherheit von Anwendungen der digitalen Wirtschaft unterstützen.

Firewall Die betriebsinternen Netzwerke stehen unter relativ umfassender Kontrolle und können daher als recht sicher betrachtet werden. Kommunikationsverbindungen nach außen hingegen sind a priori unsicher. Die Verbindung der sicheren internen Systeme mit der unsicheren Außenwelt kann dabei über eine *Firewall* abgesichert werden. Als Firewall fungiert ein Computersystem, über das die gesamte Kommunikation mit der Außenwelt abgewickelt wird. Es sollte nur mit minimaler Anwendungsfunktionalität ausgestattet sein, um die Anzahl möglicher Angriffspunkte zu minimieren. Wie in Abbildung 5.17 dargestellt, hat eine Firewall die Aufgabe nur berechtigte Zugriffe auf Systeme im sicheren Netzwerk zuzulassen und unautorisierte Zugriffe zurückzuweisen.

Eine Firewall filtert dafür den Netzwerkverkehr und läßt nur Zugriffe von autorisierten Maschinen auf freigegebene Dienste zu. In Verbindung mit Verschlüsselungstechnologien können sogenannte Virtuelle Private Netzwerke[49] aufgebaut werden. Dazu bauen zwei oder mehr Firewalls eine verschlüsselte, sichere Inter-Firewall-Verbindung über unsichere Netzwerke auf, um so ein privates, sicheres Netzwerk auf der Basis einer öffentlichen, unsicheren Netzwerkinfrastruktur zu betreiben.

X.509 Es gibt für den Bereich der Authentisierung und Verschlüsselung zwei grundlegende Technologien auf der Basis von Schlüsseln: geheime Schlüssel (symmetrische Schlüssel) und öffentliche Schlüssel[50] (asymmetrische Schlüssel).

Die Verschlüsselung auf der Basis von geheimen Schlüsseln basiert auf einem gemeinsamen Geheimnis der Kommunkationspartner. Ein Dienstanforderer identifiziert sich hierbei über das gemeinsame Geheimnis seines Kennwortes bei dem

[48] *repudiation*
[49] *virtual private networks* (VPN)
[50] *public key*

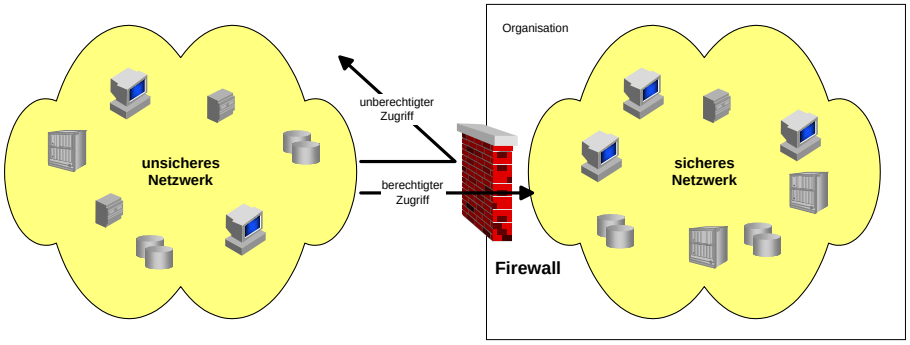

Fig. 5.17: Sicherheit durch eine Firewall

Diensterbringer. Die Informationen können dann zum Beispiel mittels dieses Kennwortes verschlüsselt übertragen werden. Für die Ver- und Entschlüsselung wird derselbe Schlüssel verwendet. Daher wird diese Variante auch symmetrische Verschlüsselung genannt.

Die grundlegende Eigenschaft der *Public Key*-Technologie ist ein asymmetrisches Schlüsselpaar. Hier sind die beiden Schlüssel der private Schlüssel (*private key*), der nur seinem Besitzer bekannt ist, und der öffentliche Schlüssel (*public key*), der der Allgemeinheit bekannt ist. Informationen, die mit dem privaten Schlüssel verschlüsselt wurden, können nur mit dem dazugehörigen öffentlichen Schlüssel wieder entschlüsselt werden und umgekehrt. So kann der private Schlüssel als digitale Unterschrift und damit zur Identifizierung des Unterschreibenden verwendet werden. Der dazugehörige öffentliche Schlüssel dient in diesem Fall der Überprüfung der digitalen Unterschrift.

Um bei dieser Technologie dem Sicherheitsanspruch gerecht zu werden, muß eine eindeutige und sichere Zuordnung von öffentlichem Schlüssel und dem Besitzer des zugehörigen privaten Schlüssels erfolgen. Diese Zuordnung darf auf keinen Fall von einem potentiellen Angreifer verändert oder vorgetäuscht werden können. So eine sichere Zuordnung von öffentlichem Schlüssel und seinem Besitzer wird durch sogenannte Zertifikate (*certificate*) realisiert, die durch Zertifizierungsstellen[51] ausgestellt werden. Diese haben einen ähnlichen Status wie Notare oder öffentliche Behörden[52].

Der Standard für die Authentifizierung in verteilten Umgebungen ist der X.509-Standard der ITU [ITU97b], der von der IETF als Basis für die Definition einer an die Anforderungen des Internets angepaßten Version X.509v3 [HFPS98] genutzt wurde.

Ein Zertifikatnehmer wird innerhalb einer Zertifizierungsstelle (CA) durch sei-

[51] *certificate authorities* (CA), wie z.B. das TC TrustCenter (`http://www.trustcenter.de`) oder VeriSign (`http://www.verisign.com`).

[52] *Pretty Good Privacy* (PGP), ein privat entwickeltes System für die Verschlüsselung von eMail-Nachrichten, setzt im Gegensatz dazu auf ein *Web of Trust*, in dem sich dessen Benutzer gegenseitig zertifizieren.

nen *distinguished name* (DN) eindeutig identifiziert. Sein Zertifikat enthält neben dem DN der ausstellenden CA und dem DN seinen öffentlichen Schlüssel, die Unterschrift der CA, den Gültigkeitszeitraum sowie die Seriennummer des Zertifikates und ist von der CA digital unterschrieben. Die Vertrauenswürdigkeit der CA-Unterschrift wird durch eine Zertifizierungshierarchie sichergestellt, an deren Spitze die *Internet Policy Registration Authority* (IPRA) steht. Diese untersteht direkt der *Internet Society* und vergibt nur Zertifikate an die nächste Hierarchieebene, den *Policy Certification Authorities* (PCAs). Diese vergeben Zertifikate an die eigentlichen Zertifizierungsstellen, den *Certification Authorities* (CAs).

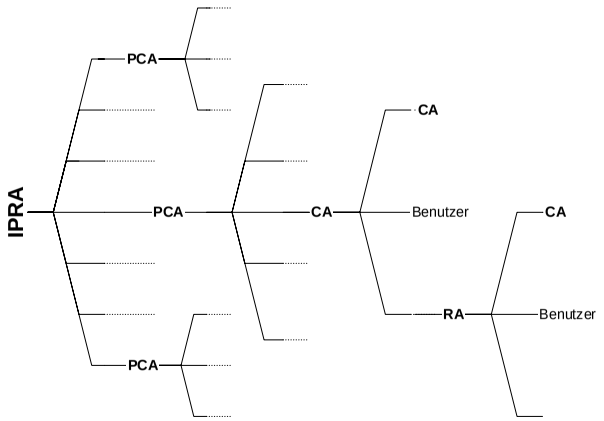

Fig. 5.18: Zertifizierungshierachie in X.509v3

Diese Beziehung ist in Abbildung 5.18 dargestellt. Zusätzlich gibt es *Registration Authorities* (RAs), die selbst keine Zertifikate ausgeben dürfen, aber die Identitätsprüfung[53] eines Zertifikatsbeantragenden vornehmen können. CAs dürfen nur Zertifikate innerhalb ihres Zweiges des X.500-Verzeichnisbaums vergeben. Unbekannte Zertifikate werden bei der nächst höheren Instanz angefragt und weitergeleitet. Einzelne Zertifikate können auch ungültig werden, etwa durch Ausscheiden eines Mitarbeiters. Für diesen Fall verwalten die CAs sogenannte *Certificate Revocation Lists* (CRL), in denen die Seriennummern abgelaufener Zertifikate aufgeführt werden. Zudem hat jede Prüfung eines Zertifikates auch gegen die aktuelle CRL zu erfolgen. Die Zertifikate und CRLs sollten in LDAP-fähigen Verzeichnissen der CAs verwaltet werden.

SSL Der *Secure Socket Layer* (SSL) Version 3 [FKK96] der IETF[54] bietet eine transparente, sichere Verbindung auf der Ebene der TCP/IP-Socket-Schnittstelle[55], also auf Sitzungsebene. Er ist ein de-facto-Standard für die sichere Kommunikation zwischen Web-Browser und Web-Server. SSL nutzt dazu eine *Public*

[53] Die Post ist zum Beispiel eine entsprechende *Registration Authority*, die die Identitätsprüfung über den Personalausweis vornimmt.
[54] ursprünglich entwickelt von Netscape.
[55] SSL schiebt zwischen TCP und dem darüberliegenden Anwendungsprogramm eine Verschlüsselungsschicht ein.

Key-Verschlüsselung[56] auf der Basis von Algorithmen der RSA Data Security, Inc. Die Identifizierung der Kommunikationspartner erfolgt hierbei über digitale Zertifikate entsprechend X.509v3. Die URL zur Nutzung von Angeboten unter Verwendung des SSL beginnt mit dem Schema `https://`. Dabei wird dieses Schema genauso verwendet wie das `http://`-Schema. Die Verschlüsselung erfolgt transparent und beeinflußt die genutzten Anwendungen nicht. Daher kann die Funktionalität von SSL ohne Probleme nachträglich in bestehende Anwendungen integriert werden[57].

Für das *Network Computing*-Referenzmodell wähle ich im Bereich der Sicherheitsdienste X.509v3 und SSLv3 als technologische Grundkonzepte.

5.2.3 Computing-Dienste

Integration

Im Bereich der digitalen Wirtschaft gibt es diverse Gründe für die Integration von Alt-Anwendungen[58] in die Anwendungen des *Network Computing* . Im Fall von Intranetanwendungen gilt es, einer großen Zahl von Endanwendern einen komfortablen Zugang zu einer großen Bandbreite an Informationen und Anwendungen zu geben. Im Internet geht es um die Bereitstellung von universellen Zugriffsmöglichkeiten auf die Unternehmenssysteme. Im Bereich des Extranet müssen Informationen mit Lieferanten und anderen Geschäftspartnern geteilt werden. Dies betrifft die Prinzipien des **universellen Zugriffs** und der **verteilten Verarbeitung**. Alle diese Anwendungen basieren meist auf der Integration von bestehenden Informationssystemen. Im folgenden stelle ich daher drei Mechanismen vor, die eine Integration über jeweils unterschiedliche Konzepte möglich machen. Entsprechend Tabelle 5.1 bieten sich hier Technologien aus den Bereichen WWW, Zentralrechner und den verteilten Objekten an.

CGI Das *Common Gateway Interface* (CGI) [CR98] ist eine einfache Schnittstelle für den Aufruf externer Programme durch einen Anwendungsserver. Diese Schnittstelle wird seit 1993 im WWW verwendet und basiert auf den Ideen zur Integration externer Programme in den Web-Server des *U.S. National Centre for Supercomputing Applications* (NCSA).

Das CGI stützt sich dabei nicht auf eine Programmierschnittstelle (API) sondern auf Umgebungsvariablen sowie die Standardeingabe bzw. -ausgabe. Anwendungen können daher in beliebigen Programmiersprachen erstellt werden. Häufig wird die Skriptsprache Perl [WCS96] für die Realisierung von Anwendungen[59] eingesetzt. Auch C oder C++ sind häufig verwendete Programmiersprachen.

Der Ablauf des Aufrufs eines externen Programms über das CGI ist dabei in Abbildung 5.19 dargestellt.

[56] verwendet werden 128-bit-Schlüssel bzw. 40-bit-Schlüssel außerhalb der USA.

[57] Eine Alternative wäre die *Security Architecture for the Internet Protocol* (IPSec) [Atk95, Tha98], welches Authentifikation, Integrität und Vertraulichkeit auf der Basis von IP-Paketen unterstützt. Im Gegensatz zu SSL bietet IPSec eine Sicherheit auf Netzwerk-Ebene und nicht auf Session-Ebene. Auch IPSec verwendet Zertifikate für die Identifizierung der Kommunikationspartner und Verschlüsselung der Informationen.

[58] *legacy systems*

[59] z.B. ist die bekannte Online-Shopping-Software InterShop in Perl entwickelt worden.

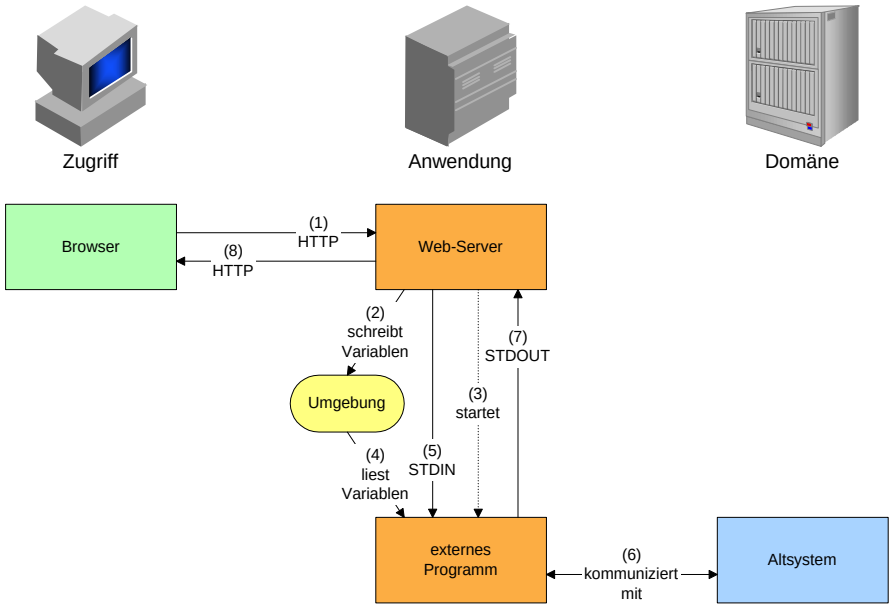

Fig. 5.19: Ablauf eines CGI-Aufrufs

Zunächst (1) wird von einem Browser ein HTTP-Request zu einem Webserver geschickt, der anhand der URL[60] erkennt, daß es sich um einen CGI-Aufruf handelt. Er schreibt eine Reihe von Informationen in Umgebungsvariablen (2) und startet das angegebene externe Programm (3). Dieses liest die Variablen aus der Umgebung (4) und die Parameter über die Standardeingabe (5). Anschließend erfüllt das Programm seine Aufgabe, z.B. unter Nutzung von Ressourcen aus der Domänenebene (6) und bereitet seine Ausgabe in Form eines gültigen MIME-Typs vor. Das Ergebnis wird über die Standardausgabe (7) an den Web-Server zurückübermittelt und von diesem, als Ergebnis des CGI-Aufrufs, per HTTP-Response wieder an den Browser zurückgegeben (8).

Dieses Verfahren basiert also auf einer synchronen Kopplung von Systemen und ist dabei völlig unabhängig von der verwendeten Programmiersprache bzw. dem aufgerufenen CGI-Programm. Der Einsatzbereich dieses Verfahrens reicht von der Generierung einfacher Feedbackseiten bis hin zu komplexen Systemen wie dem Federal Express-Sendungsverfolgungssystem.

MQ Das Konzept des *Message Queuing* basiert auf der asynchronen Kommunikation zwischen Sender und Empfänger von Anfragen. Die Zusammenarbeit erfolgt dabei nicht in der Form von Aufrufen, sondern über den Austausch von Nachrichten (*Messaging*). Die Übermittlung der Nachrichten erfolgt nach dem *Store-and-Forward*-Prinzip, ähnlich dem Versand von eMail-Nachrichten[61] (*Queuing*) im Zusammenhang mit POP3. Der Vorteil dieses Ver-

[60] In der Konfiguration des Servers wird definiert, welche URLs einen CGI-Aufruf bewirken sollen.
[61] siehe Abschnitt 5.2.2

fahrens ist in erster Linie die lose Kopplung der beteiligten Systeme über sogenannte Nachrichten-Warteschlangen *message queues*. Dadurch muß der Empfänger einer Nachricht nicht notwendigerweise zum Zeitpunkt des Absendens der Nachricht verfügbar sein. Auch ist es möglich, eine Art Rundruf (*broadcast*) an alle angeschlossenen Systeme zu schicken, bei dem nicht unbedingt eine Antwort erwartet wird. Auch können Prioritäten für die Nachrichten vergeben werden, die dann von den Queue Managern bei der Weiterleitung von Nachrichten berücksichtigt werden.

Das Nachrichtenformat basiert in der Regel auf einer beliebigen Anzahl von Bytes, die transparent übertragen werden, d.h. das MQ-System interpretiert diese Nachrichten nicht. Die jeweiligen Kommunikationspartner haben die interne Struktur, z.B. feste Satzlänge oder Feldtrennung durch spezielle Trennzeichen jedoch im Vorwege genau spezifiziert. Dies garantiert die Unabhängigkeit von Änderungen in der Architektur der unterschiedlichen Systeme, was besonders für sehr autonome Systeme, etwa in unterschiedlichen Unternehmen, relevant ist.

Daher ist diese Art der Integration von Anwendungen sehr flexibel[62]. Das Prinzip des *Message Queuing* ist in Abbildung 5.20 dargestellt:

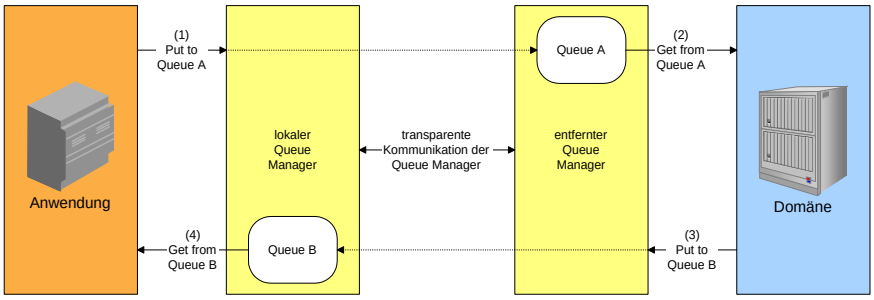

Fig. 5.20: Prinzip des *Message Queuing*

Es gibt nur zwei grundlegende Operationen - `MQPUT` und `MQGET` - für den Aufruf der Queue Manager. Die Anwendung auf der Anwendungsebene übermittelt ihre Nachricht (1) über ihren Queue Manager an die Message Queue einer Anwendung auf der Domänenebene. Der Transport der Nachricht erfolgt dabei über den Queue Manager an den zuständigen Queue Manager auf der Domänenebene. Für die Anwendung auf der Anwendungsebene ist nur die Bezeichnung der Queue ihres Partners auf der Domänen Ebene wichtig. Die Anwendung auf der Domänenebene holt regelmäßig Nachrichten (2) aus ihrer Queue über ihren lokalen Queue Manager und legt die Anworten in die Queue des Anforderers (3). Die Anwendung auf der Anwendungsebene holt auch regelmäßig Antworten (4) aus ihrer Queue und verarbeitet diese.

Leider gibt es im Bereich des *Message Queuing* keinen offziellen Standard. Aufgrund seiner Verbreitung und seines Status als Quasi-Industriestandard wähle

[62] Die *Message-Oriented Middleware Association* (`http://www.moma-inc.org`) formuliert es so: *Every DAD needs a MOM*, wobei DAD für *Distributed Application Development* und MOM für *Message-Oriented Middleware* steht.

ich daher das *Message Queue Interface* (MQI) [BHL95] des Produktes *MQSeries* [IBM95] der IBM als Technologie für das *Message Queuing*. Fast alle Produkte anderer führender Hersteller[63] bieten Schnittstellen zu diesem System an.

CORBA/IDL Auch im Bereich der Integration bietet ein CORBA-Broker nützliche Dienste. Die Beschreibung der Schnittstellen der vom ORB verwalteten Objekte wird unabhängig von der Implementationssprache durch die *Interface Definition Language* (IDL)[64] [OMG98c] vorgenommen. Die IDL ist eine rein deklarative Sprache. Sie kann nicht für die Spezifikation von Implementierungsdetails verwendet werden. Für die verschiedensten Programmiersprachen werden Abbildungen (*Bindings*) zur Verfügung gestellt, über die dem ORB die in der jeweiligen Programmiersprache erstellten Objekte beschrieben werden.

Wie in Abbildung 5.21 dargestellt, dient IDL hier als Integrationspunkt für die Anwendungen, die in unterschiedlichen Programmiersprachen entwickelt werden und wurden.

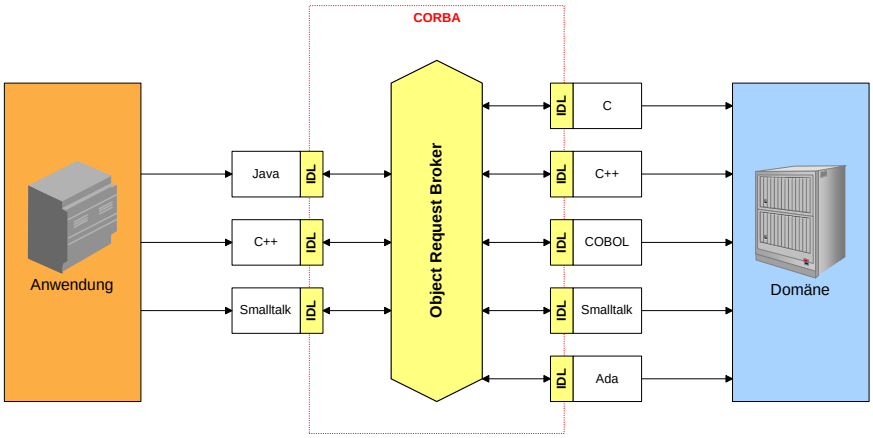

Fig. 5.21: Integration über die CORBA *Interface Definition Language*

Über diesen Weg kann zum Beispiel eine Java-Anwendung auf der Anwendungsebene mittels des ORBs die Dienste einer COBOL-Anwendung auf der Domänenebene für ihre Zwecke nutzen. Voraussetzung ist dabei nur das Vorhandensein eines ORBs auf Seiten der Anwendungs- und Domänenebene sowie die Spezifikation der Dienste der COBOL-Anwendung in Form der IDL.

Für den Bereich der Integrationsdienste wähle ich daher das CGI, das MQI sowie die CORBA/IDL.

Transaktion

Im Bereich der digitalen Wirtschaft entstehen Anwendungen mit sehr hohen Anforderungen an Skalierbarkeit und Transaktionssicherheit in verteilten Umgebungen. Die entsprechenden Aufgaben des Transaktionsdienstes werden dabei

[63] wie z.B. BEAs MessageQ.
[64] seit 1998 ein ISO Standard (14750).

traditionell von sogenannten *Transaction Processing*-Monitoren[65] (TP-Monitor) übernommen[66].

TP-Monitore erweitern das ACID-Prinzip von Datenbanktransaktionen[67] auf beliebige, sogenannte *Resource Manager*. Diese können Datenbanksysteme, Messaging Systeme und ähnliches sein. Eine entsprechende Transaktion sichert nicht nur die konsistente Aktualisierung der Datenbank, sondern auch den erfolgreichen Versand einer Benachrichtigungs-eMail sowie die Übermittlung eines Datensatzes in die Eingangsqueue eines über ein *Message Queuing*-System[68] angeschlossenen Fremdsystems.

Im Rahmen des *Network Computing* wähle ich die zwei folgenden Standards im Bereich Transaktionsdienste, wobei entsprechend Tabelle 5.1 der Schwerpunkt auf der Technologie der Zentralrechnerwelt liegen muß.

Distributed Transaction Processing (DTP) Der Standard für verteilte Transaktionsverarbeitung im Rahmen der Zentralrechnerwelt ist das *Distributed TP: Reference Model* (DTP) [X/O96] der *Open Group*. Bekannt ist dieser Standard eher unter dem Namen der Schnittstelle zwischen *Transaction Manager* und *Resource Manager* - der XA-Schnittstelle. Die meisten relevanten Systeme der Domänenebene bieten XA-konforme Schnittstellen für die Zusammenarbeit mit einem entsprechenden *Transaction Manager* an[69].

Das DTP Modell nutzt ein *Two-Phase-Commit*-Protokoll (2PC), um die ACID-Eigenschaften auch bei verteilten Transaktionen sicherzustellen. In der ersten Phase werden alle beteiligten *Resource Manager* gefragt, ob sie die Transaktion durchführen können. Nur falls alle unwiderruflich ihre entsprechende Bereitschaft gemeldet haben, werden sie in der zweiten Phase angewiesen, die Änderungen durchzuführen und damit ein systemweites *Commit* durchgeführt. Anderenfalls wird ein übergreifendes *Rollback* eingeleitet[70].

In Abbildung 5.22 ist das Konzept des *Distributed Transaction Processing* dargestellt.

Eine Anwendung nutzt dabei die Standard-API der Domänendienste[71], steuert jedoch die Transaktion mittels eines *Transaction Manager* über dessen TX-Schnittstelle [X/O95b]. Mittels der TX-Schnittstelle informiert eine Anwendung den *Transaction Manager* über den Beginn (`tx_begin`) und das Ende einer Transaktion und in diesem Zusammenhang, ob die Transaktion erfolgreich durchgeführt (`tx_commit`) werden kann oder abgebrochen (`tx_rollback`) werden muß.

[65] wie IBMs CICS oder BEAs Tuxedo.
[66] Ursprünglich stand die Bezeichnung TP-Monitor für *Teleprocessing Monitor* - eine Anwendung, die das Multiplexing von Terminalverbindungen zu einem Großrechner übernahm. Schon damals stand die Lastverteilung und damit Skalierbarkeit beim Einsatz eines TP-Monitors im Vordergrund. Mit dem Übergang zu transaktionsorientierter Verarbeitung (siehe Abschnitt 2.2.2) wurde diese Bedeutung dann geändert.
[67] siehe Abschnitt 2.1.
[68] siehe Abschnitt 5.2.3.
[69] So gibt es zum Beispiel eine XA-Bibliothek für das Oracle RDBMS.
[70] für weiterführende Informationen zu diesem Thema sei auf [Dat95] verwiesen.
[71] z.B. die in Abschnitt 5.2.1 beschriebenen Schnittstellen der Persistenzdienste wie ODBC.

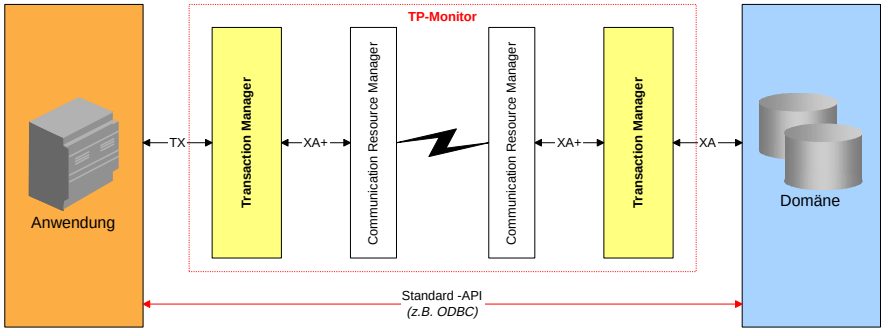

Fig. 5.22: Beziehungen im *Distributed Transaction Processing*

Der *Transaction Manager* dient zur Koordination und Kontrolle der genutzten *Resource Manager*[72]. Die Kommunikation der *Transaction-* und *Resource Manager* erfolgt über die XA-Schnittstelle [X/O92]. Der *Transaction Manager* nutzt den Aufrufe `xa_start`, um den Beginn einer Transaktion mitzuteilen. Mittels der `xa_prepare`, `xa_commit` und `xa_rollback` Aufrufe wird dann dem *Resource Manager* mitgeteilt, wie er sich im Rahmen des 2PC zu verhalten hat. Ein `xa_end` bezeichnet das Ende der Transaktion.

Für verteilte Transaktionen wird ein spezieller *Resource Manager* benötigt, der die Kommunikation zwischen den beteiligten *Transaction Managern* übernimmt. Die Zusammenarbeit zwischen *Communication Resource Managern* (CRM) wird über die XA+ Schnittstelle [X/O94] geregelt, die unter anderem die *Transaction Manager* über den Zustand der verteilten Transaktion informiert.

OTS In der Welt der Objekte ist das logische Äquivalent zum *Transaction Manager* des DTP-Modells der *Object Transaction Service* (OTS) [OMG97b] der OMG im Rahmen von CORBA[73] .

Ein an einer Transaktion beteiligtes Objekt kann eine von mehreren möglichen Rollen übernehmen. Es kann entweder ein *Transactional Client*, ein *Transactional Server* oder ein *Recoverable Server* sein. Ein Objekt ist transaktional, wenn es von der abstrakten `TransactionalObject`-Klasse erbt.

Ein *Transactional Client* ruft eine Reihe von Methoden auf, die innerhalb einer Transaktionsklammer eingeschlossen sind. Diese Aufrufe können an transaktionale und nicht-transaktionale Objekte gehen. Den Beginn-Aufruf fängt der ORB ab und leitet ihn an den OTS[74] weiter, der einen Transaktionskontext für diesen Client erzeugt. Anschließend ruft der Client Methoden entfernter Objekte auf. Der ORB propagiert den Transaktionskontext implizit bei jeder Kommunikation zwischen den an der Transaktion beteiligten Objekten. Falls der Client einen Commit- oder Rollback-Aufruf tätigt, informiert der ORB den OTS. Dies alles geschieht völlig transparent und ohne Zutun des Clients.

[72] z.B. Datenbank- oder Message Queuing-Systeme.
[73] Böse Zungen behaupten, daß sich die Leistungen von ORB-Transaktionsdiensten eher in Sekunden pro Transaktion als in Transaktionen pro Sekunde messen ließen.
[74] entspricht dem *Transaction Manager* des DTP-Modells.

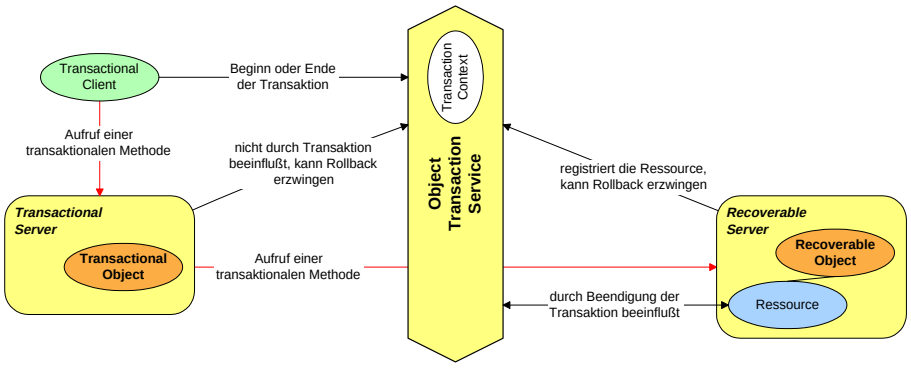

Fig. 5.23: Object Transaction Service

Ein *Transactional Server* ist eine Ansammlung von einem oder mehreren transaktionalen Objekten, deren Verhalten von der Transaktion zwar beeinflußt wird, die jedoch keine eigenen Ressourcen besitzen und ihre Zustände nicht persistent festschreiben. Die augenblicklichen Zustände werden lediglich im Hauptspeicher gehalten und sind deshalb nicht wiederherstellbar. Sie nutzen Ressource-Objekte eines *Recoverable Servers* im Rahmen einer Transaktion. Zwar kann ein *Transactional Server* mit den von ihm verwalteten Objekten nicht am Transaktionsabschluß teilnehmen, sehr wohl aber die Beendigung einer Transaktion mit Rollback erzwingen.

Ein wiederherstellbarer Server (*Recoverable Server*)[75] ist eine Ansammlung von einem oder mehreren transaktionalen Objekten, die eine Ressource verwalten und deren Zustände daher vom Abschluß oder Abbruch einer Transaktion betroffen sind. Entsprechende Beispiele sind Datenbanken oder Queues. Diese Objekte melden dem Transaktionsdienst ihre Teilnahme an einer Transaktion, deren Kontext sie beim Aufruf durch einen *Transactional Server* erhalten haben. Der Transaktionsdienst fungiert dabei gleichzeitig als Koordinator beim Transaktionsabschluß im Rahmen des 2PC.

Zudem unterstützt der OTS auch die TX- und XA-Schnittstellen des DTP-Modells für die Client und *Resource Manager* Kommunikation.

Für den Transaktionsdienst im Rahmen des *Network Computing* wähle ich daher das DTP-Modell (TX/XA) und den CORBA/OTS.

Programmierung

Anwendungen der digitalen Wirtschaft müssen genauso mittels Programmiersprachen implementiert werden, wie klassische Anwendungen. Es werden auch hier spezielle Programmiersprachen benötigt, um die einzelnen fachlichen und technischen Probleme zu lösen. Diese stehen auf der Zugriffs- sowie der Anwendungsebene zur Verfügung - jedoch in unterschiedlichen Ausprägungen - , die zum Teil bereits beschrieben wurden. Im Rahmen der folgenden Ausführungen

[75] entspricht dem *Resource Manager* des DTP-Modells.

werde ich nun jeweils die Unterschiede des Einsatzes der Lösungen auf der Zugriffsebene (clientseitige Programmierung) und auf der Anwendungsebene (serverseitige Programmierung) erläutern.

Zunächst wähle ich daher eine bestimmte Programmiersprache, die meine Anforderungen entsprechend den Kriterien zur Auswahl von Technologien des *Network Computing* erfüllt. Als Ergänzung dazu nehme ich noch eine Skriptsprache hinzu, die für einfachere Aufgaben zur Verfügung stehen soll. Abschließend erläutere ich noch ein weiteres Konzept zur Realisierung von Funktionalität auf der Anwendungsebene.

Java Als Programmiersprache wähle ich Java[76] [GJS96, Fla98]. Gründe für den Einsatz von Java als Programmiersprache im Rahmen des *Network Computing* sind für mich dabei folgende:

- Die Plattformunabhängigkeit ist Grundlage des universellen Zugriffs und der verteilten Verarbeitung.
- *Remote Method Invocation* (RMI) bildet die Fähigkeit zur Verteilung der Anwendungen und ist eine Möglichkeit zur Realisierung verteilter Verarbeitung.
- Eingebaute Sicherheitsmechanismen - etwa über Zertifikate signierte Applets - bieten die nötige Sicherheit.
- Java bietet eine Reihe von APIs für viele der bereits aufgeführten Bestandteile des Referenzmodells. So gibt es das *Java Naming and Directory Interface* (JNDI), die *Java Database Connectivity* (JDBC) oder den *Java Transaction Service* (JTS).
- Java bietet Komponentenmodelle für clientseitige Programmierung (JavaBeans) und serverseitige Programmierung (Enterprise JavaBeans).
- Wie bereits beschrieben, können Java-Programme als Applets direkt in HTML-Seiten eingebettet werden.
- Serverseitig können Java-Programme als Servlets die Funktionalität von Web-Servern erweitern.
- Durch das Interface-Konzept ist eine sinnvolle Trennung von Implementierung und Schnittstelle möglich.

Aus all diesen Gründen hat sich Java als strategische Technologie für die Entwicklung von Informationssystemen im allgemeinen etabliert. Eine Studie von Zona Research [Zon97] belegt dies mit entsprechenden Zahlen. Hauptsächlich wird diese Programmiersprache für die Entwicklung neuer Anwendungen bzw. die Erweiterung bestehender Anwendungen eingesetzt: Etwa 84 % der Befragten gaben an, Java für die Entwicklung neuer Systeme einsetzen zu wollen. Dies ist in Anbetracht der noch jungen Geschichte dieser Programmiersprache eine bemerkenswerte Zahl. Die Ablösung existierender Anwendungen ist nach der

[76] Bill Joel, Mitgründer von SUN, bezeichnet Java als C plus-plus-minus-minus.

Studie eher ein untergeordnetes Ziel. Dies steht im Einklang mit meiner Forderung nach einem integrativen Charakter des Verarbeitungsmodells und damit auch der zugrundeliegenden Infrastruktur und Programmierumgebung.

Ein wichtiges Argument für den Einsatz von Java scheint auch das Interesse der Programmierer an dieser Sprache zu sein und damit das entsprechende Angebot an qualifizierten Programmierern. Rund 62 % der Befragten entschieden sich aus diesem Grund für Java.

Die Nutzung von Java auf der Zugriffsebene wurde bereits im Bereich der Präsentationsdienste beschrieben. Abbildung 5.24 stellt den Lebenslauf eines Applets dar. Die Schnittstelle `java.applet` fordert unter anderem die Methoden `init`, `start`, `stop` und `destroy`, die von der Ausführungsumgebung aufgerufen werden und den Lebenslauf des Applets steuern.

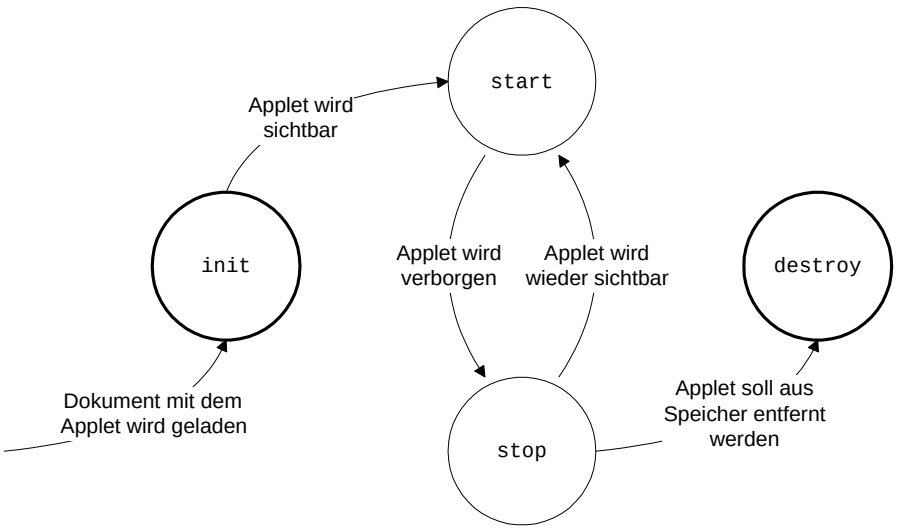

Fig. 5.24: Lebenslauf eines Applets

Das hier beschriebene Konzept der Applets steht auch auf der Ebene der Anwendung zur Verfügung. Dort wird eine entsprechende Java-Anwendung *Servlet* [DA98] genannt. Dies sind Java Klassen, die das `javax.servlet.*` Interface implementieren. Die Einbettung der Servlets erfolgt mit dem Tag `<SERVLET>`. Der Server parst das HTML-Dokument, führt die dort referenzierten Servlets aus und ersetzt die Stelle des Aufrufs mit dem Ergebnis der Ausführung[77].

In Abbildung 5.25 ist das Grundprinzip der Anwendung von Applets und Servlets dargestellt. In einer über HTTP aufgerufenen HTML-Seite befindet sich die Referenz auf ein Applet, welches ebenfalls über HTTP geladen wird. Es wird initialisiert und gestartet und kommuniziert mit einer serverseitigen Anwendung in Form eines Servlets über HTTP oder IIOP. Es sind natürlich auch einfachere

[77] Servlets sind damit eine Variante des CGI-Konzeptes, allerdings ohne dessen extremen Ressourcenbedarf. Dies erkauft man sich jedoch mit der Festlegung auf die Programmiersprache Java.

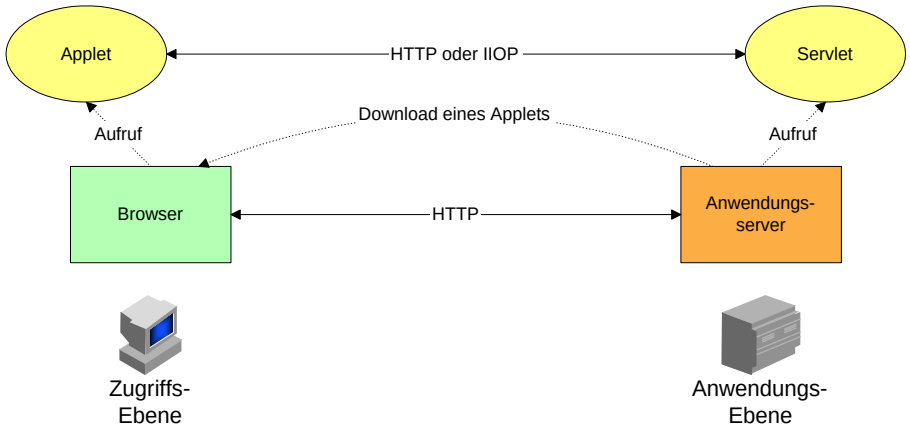

Fig. 5.25: Verteilte Anwendung mit Applets und Servlets

Varianten dieses Prinzips denkbar, in denen jeweils nur auf der Zugriffs- oder der Anwendungsebene Applets bzw. Servlets eingesetzt werden[78].

In bezug auf die Sicherheit auf der Zugriffsebene gibt es bei der Verwendung von Applets zwei Konzepte. Zunächst stehen allen Applets grundsätzlich nur stark eingeschränkte Möglichkeiten der Nutzung von lokalen Ressourcen wie Dateisystemen oder Netzwerkverbindungen zur Verfügung (Sandkastenprinzip). Wenn erweiterte Funktionalität in diesen Bereichen erforderlich ist, muß ein Applet seine Herkunft über ein Zertifikat seines Autors nachweisen können - dieses Applet nennt man ein signiertes Applet. Einem Applet eines als vertrauenswürdig eingestuften Autors können dann auf der Zugriffsebene besondere Berechtigungen bezüglich des Zugriffs auf lokale Ressourcen erteilt werden.Applets stellen eine Art von Komponenten dar, allerdings ohne die Fähigkeit zur Introspektion und Anpassung[81].

Bei der Verwendung der Programmiersprache Java stehen eine Reihe von APIs zur Verfügung, die die Nutzung von diversen, bereits beschriebenen Diensten ermöglichen:

Persistenz	Java Database Connectivity (JDBC)
Verzeichnisse	Java Naming and Directory Interface (JNDI)
Transport	JDK java.net
Message Queuing	Java Message Service (JMS)
CORBA	Java IDL

[78] Bei der Verwendung von Java Applets auf der Zugriffsebene bietet sich auch die Nutzung von *Remote Method Invocation* (RMI) [Sun98c] mittels des `java.rmi.*` Interface als Kommunikationsmechanismus an[79]. Dies funktioniert jedoch nur, wenn der Kommunikationspartner auf der Anwendungsebene ebenfalls eine Java-Anwendung bzw. ein Servlet[80] ist. Für Verbindungen über eine Firewall versucht RMI zunächst eine direkte Socketverbindung zum Server aufzubauen und bei Fehlschlag den Request in ein HTTP-Request zu verpacken (*Tunneling*). Ein Nachteil der Nutzung von RMI ist die Beschränkung auf Java als Implementationsplattform für die Anwendungen der Anwendungsebene.

[81] siehe Abschnitt 5.2.3.

XA-Transaktionen Object Transaction Service	Java Transaction API (JTA) Java Transaction Service (JTS)
Management[82]	Java Management API (JMAPI)

Tab. 5.7: Java Enterprise APIs

JavaScript Für einfachere Aufgaben stelle ich im Rahmen des *Network Computing*-Referenzmodells eine Skriptsprache zur Verfügung. Eine Skriptsprache ist nach [ECM98] eine Programmiersprache, die verwendet werden kann, um Möglichkeiten eines existierenden Systems zu manipulieren, anzupassen und zu automatisieren. In solchen Systemen ist bereits Funktionalität über eine Bedienerschnittstelle zugänglich. Eine Skriptsprache eröffnet in diesem Rahmen die Möglichkeit, diese Funktionalität unter Programmkontrolle zu stellen. Dabei bietet das existierende System eine Umgebung mit einer Menge von Objekten und Funktionen, die die Fähigkeiten der Skriptsprache komplettieren. Wie bereits im Bereich der Präsentationsdienste beschrieben, wähle ich JavaSkript als die Skriptsprache im Rahmen des *Network Computing* .

JavaSkript ist objektbasiert: die Basissprache und die Möglichkeiten der Umgebung werden durch Objekte bereitgestellt. Je nach Einsatz, auf der Zugriffs- oder der Anwendungsebene, werden unterschiedliche Umgebungsobjekte bereitgestellt. Dies ist in Abbildung 5.26 deutlich gemacht. *ECMA Script* definiert dabei den *Core*-Bereich von JavaScript.

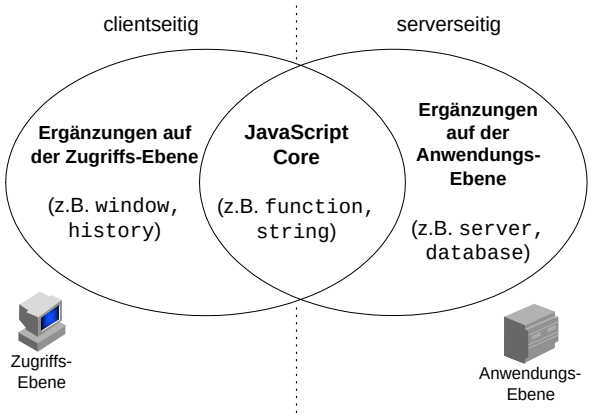

Fig. 5.26: Unterschiedliche Ausprägungen von JavaScript

Mit Hilfe von JavaScript auf der Zugriffsebene lassen sich einfache Aufgaben wie Feldprüfungen, Layoutsteuerungen oder Animationen clientseitig realisieren.

Auch auf der Anwendungsebene kann mit Hilfe von serverseitigem JavaScript [Net97] einfache Anwendungsfunktionalität erreicht werden. So basiert Netscape LiveWire-Datenbankanbindung auf der Programmierung mit serverseitigem

[82] siehe Abschnitt 5.2.4.

JavaScript. Auch hier parst der Server die HTML-Dokumente, interpretiert die entsprechenden Skripte innerhalb des `<SERVER>`-Tags und fügt die Ergebnisse der Skriptausführung in die Dokumente ein.

SSI Die letzte Möglichkeit zur Programmierung serverseitiger Anwendungen sind die sogenannten *Server Side Includes* (SSI) . Hier werden Ausdrücke in der Form `<!--#include file="header.html"-->` verwendet, um zum Beispiel Teildokumente in andere Dokumente einzubetten. Dies kann unter anderem dazu genutzt werden, einheitliche Pro- und Epiloge wie Copyright-Vermerke o.ä. auf allen Dokumenten über ein gemeinsames HTML-Fragment zu realisieren und damit die Pflege einer großen Anzahl von Dokumenten zu vereinfachen.

Als Programmiersprachen für das *Network Computing*-Referenzmodell wähle ich Java und JavaScript.

Komponenten

Komponenten sind eigenständige, wiederverwendbare Softwarebausteine, die mit anderen Komponenten kombiniert werden können, um ganze Anwendungen zu bilden. Sie können in unvorhersehbaren Konstellationen verwendet werden und bieten ihre Dienste über wohl definierte Schnittstellen an [Gri98]. Komponenten sind oft verteilte Objekte[83] mit ausgeprägten Verwaltungsmöglichkeiten. Sie können über ihre Eigenschaften (*Properties*) an die jeweiligen spezifischen Bedürfnisse der Anwendung, in der sie eingesetzt werden sollen, angepaßt werden[84].

Ein *Komponentenmodell* beschreibt das Ausführungsumfeld von wiederverwendbaren Komponenten und definiert zwei wesentliche Elemente: die Komponenten an sich und die Behälter (*Container*), in deren Kontext sie ausgeführt werden. Es spezifiziert die Mechanismen und Schnittstellen, mit denen eine Komponente mit ihrem Container interagiert.

Container stellen nicht nur den Kontext für die Ausführung, sondern auch für die Interaktion von Komponenten zur Verfügung. Beispiele für Container visueller Komponenten der Zugriffsebene sind Fenster (*Windows*) oder Rahmen (*Frames*). Auf der Anwendungsebene sind es z.B. Webserver oder Transaktionsmonitore. Container können ineinander geschachtelt werden, wenn ein Container zur Komponente innerhalb eines anderen Containers wird.

Durch die Spezifikation eines Komponentenmodells wird erreicht, daß die Ergebnisse der Entwickler von Komponenten auch in den Containern anderer Hersteller lauffähig sind. Diese Interoperabilität ist die Grundvoraussetzung für die Vorteile von Komponenten und eine Manifestation des **gemeinsamen Verarbeitungsmodells**.

Im folgenden werde ich drei Komponentenmodelle vorstellen. Das erste ist ein java-basiertes Modell für die Zugriffsebene, das zweite ein java-basiertes Modell

[83] siehe Abschnitt 3.2.4.

[84] So kann z.B. über die Eigenschaft `Name` die angezeigte Bezeichnung einer *Button*-Komponente verändert werden oder über eine Eigenschaft einer Buchhaltungskomponente die zu nutzende Datenbank spezifiziert werden.

für die Anwendungsebene des *Network Computing* und das letzte ein allgemeines, ebenenübergreifendes Modell. Entsprechend Tabelle 5.1 stehen hier nur Alternativen aus der objektorientierten Welt zur Verfügung.

JavaBeans [Sun97a] ist ein von SUN definiertes Komponentenmodell, das für Softwareentwickler als Leitlinie für die Herstellung von Komponenten auf der Grundlage von Java dienen soll. Ein *Bean* ist definiert als eine wiederverwendbare Softwarekomponente, die mit Hilfe eines Entwicklungswerkzeuges[85] visuell manipuliert werden kann. Jedes JavaBean hat folgende Eigenschaften:

- **Introspection**: Die Funktionsweise einer *Bean* kann analysiert werden.
- **Customization**: Die Eigenschaften (*Properties*)[86], die das Erscheinungsbild und das Verhalten steuern, können angepaßt werden.
- **Events**: Eine *Bean* reagiert auf Ereignisse[87] und kann anderen Komponenten Ereignisse signalisieren.
- **Persistence**: Der Zustand einer *Bean* kann gesichert und später wieder geladen werden.

JavaBeans verwende ich in erster Linie als visuelle Komponenten auf der Zugriffsebene, auch wenn es grundsätzlich möglich ist, unsichtbare JavaBeans zu erstellen.

Enterprise JavaBeans Das Pendant zu JavaBeans auf der Anwendungsebene sind SUNs *Enterprise JavaBeans* (EJB) [Sun98a]. Sie erweitern das JavaBean Komponentenmodell um die Unterstützung serverseitiger, transaktionaler Komponenten.

Ein EJB Container ist für die Verwaltung einer Klasse von EJB Objekten (*Enterprise Beans*) zuständig. Er verwaltet den Objektlebenslauf, implementiert die Objektsicherheit und koordiniert verteilte Transaktionen[88]. Außerdem liefert er den Namensraum für das Objekt[89]. Der Container verwaltet auch den Zustand des Objektes. Dabei unterstützt das EJB-Modell transiente Objekte (*Session Beans*) und persistente Objekte (*Entity Beans*).

Ein *Session Object* wird von einem Client erzeugt und existiert meist nur für die Dauer einer Sitzung. Es korrespondiert also zu einer Client-Session und wird z.B. verwendet, um den Weg eines Users in einer Website zu verfolgen. Nach Beendigung der Sitzung wird es gelöscht. *Session Objects* können transaktional sein, jedoch sind sie nicht *recoverable*[90]. Sie können zustandslos oder

[85] z.B. IBM Visual Age, Inprise JBuilder oder Symantec Visual Cafe.

[86] *Properties* einer *Bean* sind benannte Attribute, die über entsprechende GET/SET-Methoden gelesen bzw. geändert werden können.

[87] Ereignisse erlauben im Gegensatz zu direkten Methodenaufrufen die asynchrone Nutzung von Methoden.

[88] Die Nutzung des Transaktionsdienstes erfolgt über das `javax.transaction`-Package der JTA.

[89] unter Nutzung des JNDI.

[90] siehe Abschnitt 5.2.3.

zustandsbehaftet sein, sind in diesem Fall jedoch selbst für die persistente Verwaltung ihrer Daten zuständig[91]. Dabei repräsentieren sie nie Informationen in einer Datenbank, sondern eben einen Client, der Aktionen auf dem EJB-Server durchführt.

Ein *Entity Bean* repräsentiert persistente Daten, die in einer Datenbank abgelegt sind. Sie sind vergleichbar mit einem Datensatz in einer Tabelle, sind transaktional und *recoverable*. Außerdem sind sie implizit persistent. Ein *Entity Bean* kann seine Persistenz entweder selbst verwalten oder an seinen Container delegieren.

Jedes *Bean* hat einen eindeutigen Identifikator - vergleichbar einem Primärschlüssel. Für *Entity Beans* ist dies ein Schlüssel bezüglich der persistenten Informationen, die es repräsentiert. Für *Session Beans* ist es ein, von der Art des *Beans* abhängiger Schlüssel wie z.B. eine IP-Adresse.

In Abbildung 5.27 ist die Architektur eines *Enterprise JavaBeans* Servers dargestellt:

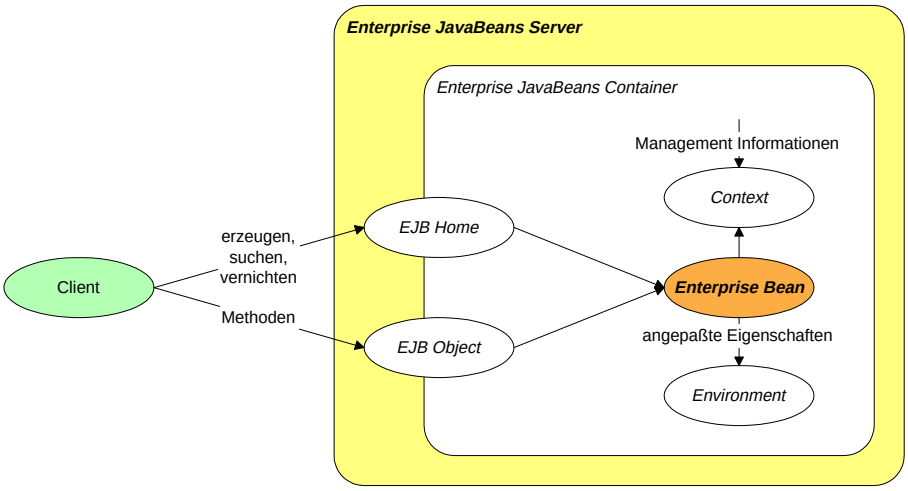

Fig. 5.27: Architektur eines *Enterprise JavaBeans*-Servers

Enterprise Beans werden in einem EJB Container innerhalb eines EJB-Servers verwendet. Der Container fungiert dabei als Verbindung zwischen Client und *Enterprise Bean*. Er erzeugt automatisch eine *EJB Home*-Schnittstelle für die *Enterprise Bean*-Klasse und für jede Instanz ein *EJB Object*. Die EJBHome-Schnittstelle verwaltet den kompletten Lebenszyklus - Erzeugung, Suche und Vernichtung der Enterprise JavaBean Instanz - und dient zur Identifizierung des Enterprise JavaBean. Der Zugriff auf die *EJB Home*-Schnittstelle wird von dem Container über das JNDI bereitgestellt. Über die *EJB Object*-Schnittstelle wird der Zugriff auf die Methoden des *Enterprise Bean* ermöglicht. Alle Zugriffe über diese beiden Schnittstellen werden zunächst vom Container abgefangen, um seine Verwaltungsaufgaben in den Bereichen Lebenszyklus, Sicherheit, Transaktionen und Persistenz zu erfüllen. Die Clienten agieren also nie direkt mit dem

[91] z.B. über JDBC oder SQLJ

Enterprise Bean, sondern nur über die beiden, vom Container bereitgestellten *Wrapper*-Schnittstellen *EJB Home* und *EJB Object*.

Für jede aktive *Enterprise Bean*-Instanz erzeugt der Container ein *Context*-Objekt, das den aktuellen Zustand der Instanz sowie Managementinformationen hält. Dieses *Context*-Objekt wird gemeinsam vom Container und der *Enterprise Bean*-Instanz genutzt. Zusätzlich wird ein *Environment*-Objekt erzeugt, das die angepaßten *Property*-Werte des *Enterprise Beans* hält.

Über ein Mapping [Sun98b] kann eine Integration der EJBs in CORBA-Umgebungen stattfinden, so daß CORBA-Clienten EJBs in einem CORBA-basierten EJB-Server nutzen und EJBs und CORBA-Objekte an gemeinsamen Transaktionen teilhaben können.

CORBA Für die Verwendung von Nicht-Java-Komponenten wähle ich die *Common Object Request Broker Architecture* (CORBA) [OMG98c] der *Object Management Group* (OMG) im Rahmen der *Object Management Architecture* (OMA) [OMG92, OMG97c] als Komponentenmodell[92]. Wie bereits erwähnt, sind verteilte Objekte im Sinne von CORBA per se schon Komponenten[93]. In Abbildung 5.28 sind die Bestandteile der *Object Management Architecture* dargestellt.

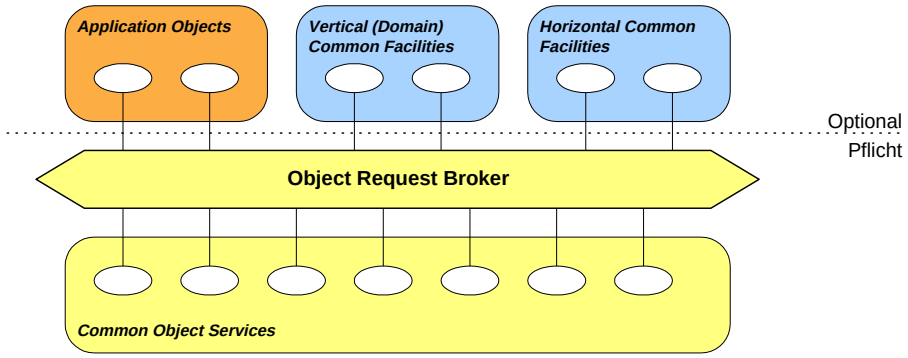

Fig. 5.28: *Object Management Architecture*

Der *Object Request Broker* (ORB)[94] als Kommunikationsinstrument ist der wichtigste Bestandteil der Architektur. Er ermöglicht den Zugriff auf die Komponenten, verwaltet deren Schnittstellen, übermittelt die Anfragen und leitet die Antworten weiter. Damit erfüllt er die Aufgaben eines Containers. Für das Management der von ihm verwalteten Komponenten stehen ihm eine Reihe von *Common Object Services* [OMG97b] zur Verfügung, die seine Aufgaben in den Bereichen Lebenszyklus, Benennung, Transaktionen, Sicherheit u.s.w. unterstützen. Sie erweitern die Fähigkeiten des ORB um die entsprechenden Dienste.

[92] Hier werde ich nur die Eigenschaften dieses Ansatzes in bezug auf das Komponentenmodell erläutern. Teilweise sind Techniken und Konzepte von CORBA bereits in anderen Abschnitten erklärt worden.
[93] vgl. Abschnitt 3.2.4.
[94] siehe Abschnitt 3.2.4.

Die *Common Facilities* [OMG95] sind wie die *Common Object Services* horizontal orientiert. Allerdings sind sie eher anwendungsorientiert und bieten generische Dienste aus den Bereichen *User Interface Management*, *Information Management*, *System Management* und *Task Management*. Außerdem sind sie wie die *Domain Facilities* optionale Bestandteile der Architektur. *Domain Facilities* konzentrieren sich als vertikale Dienste auf Anwendungsbereiche wie etwa Aufgabenstellungen im Finanzbereich [OMG98a]. Die Anwendungsobjekte (*Application Objects*) realisieren die eigentliche Anwendung und bauen auf den bereits genannten Komponenten auf.

Als Komponenten-Modelle für das *Network Computing* wähle ich also *JavaBeans*, *Enterprise JavaBeans* und CORBA.

5.2.4 Meta-Dienste

Management

In Abschnitt 4.2 wurde deutlich, wie wichtig die Möglichkeit zur effizienten und übergreifenden Verwaltung von Anwendungen und Systemen ist. Diese Möglichkeit muß somit Bestandteil des **gemeinsamen Verarbeitungsmodells** sein und ist daher im Bereich der Meta-Dienste anzusiedeln.

Unter Berücksichtigung von Tabelle 5.1 ist es sinnvoll, Technologien aus den Bereichen WWW und Zentralrechner zu wählen. Allerdings gibt es nur ein einziges standardisiertes Protokoll, das den in Abschnitt 5.1 aufgeführten Kriterien genügt:

SNMP Dieses *Simple Network Management Protocol* (SNMP) gründet auf mehreren Dokumenten, die zum einen Management-Informationen [RM90], zum anderen ein Management-Protokoll [CFSD90] definieren. Es ist angelehnt an das OSI-Netzwerk-Management Modell im Rahmen des OSI-Referenzmodells [Ker89].

Das Management-Rahmenwerk betrachtet ein Netzwerk-Management System als eine aus vier Komponenten bestehende Einheit:

- **Netzwerk Management Station (NMS)** Hier sind eine oder mehrere Netzwerk-Management-Anwendungen (*Manager*)installiert.
- **Verwaltete Knoten (*Managed Nodes*)** Auf jeder verwalteten Einheit sitzt ein *Agent*, der die Anweisungen einer Management-Anwendung ausführt. Verwaltete Knoten können sowohl physikalische (z.B: Drucker) als auch logische Einheiten (z.B. eine Queue) sein.
- **Netzwerk-Management-Protokoll** Dieses Protokoll wird für den Austausch von Informationen zwischen Management-Anwendung und Agent verwendet.
- **Management-Information** repräsentiert die Informationen über einen verwalteten Knoten und wird in der *Management Information Base* (MIB) [RM91, CMRW96c] verwaltet.

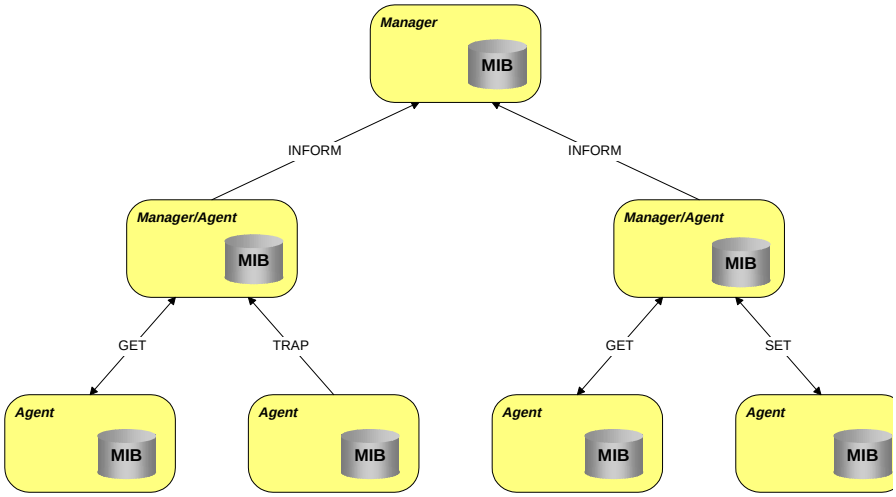

Fig. 5.29: Prinzip des *Simple Network Management Protocol*

Wie in Abbildung 5.29 dargestellt, definiert SNMP somit das Zusammenwirken zwischen Manager und Agent. Die für das Management der jeweiligen Komponenten benötigten Informationen, die die *Management Information Base* (MIB) konstituieren, können sowohl auf der Manager-Instanz als auch auf jeder kontrollierten Instanz liegen. Im ersten Fall muß die Manager-Instanz regelmäßig jede kontrollierte Instanz ansprechen (`GET`) und die Informationen auf deren Knoten anfordern. Außerdem kann ein Manager einen Agenten beauftragen, bestimmte Informationen in seiner MIB zu verändern (`SET`)[95]. Eine kontrollierte Instanz ist somit eher passiv - sie meldet sich nur in Ausnahmefällen (*TRAP*) bei ihrem Manager und informiert ihn über die eingetretene Zustandsänderung.

Für den Austausch der Informationen wird UDP[96] verwendet - ein relativ unsicheres Protokoll[97]. Außerdem kann es durch den eingesetzten Polling-Mechanismus zur Aktualisierung der Manager MIB zu einer erhöhten Netzlast kommen. Aus diesem Grund wurde SNMPv2 [CMRW96b, CMRW96a] als Ergänzung zu SNMP entwickelt. Dies erlaubt es Hierarchien von Managern mit einer eigenen Manager-zu-Manager (M2M) Kommunikation (`INFORM`). Ein Knoten kann nun Manager und verwalteter Knoten in einem sein. Die Sicherheitskonzepte in SNMPv2 wurden allerdings nie realisiert[98].

[95] Diese Funktion ist jedoch meist aus Sicherheitsgründen nicht implementiert. Die Gefahr des unberechtigten Veränderns von Informationen über das doch recht unsichere Protokoll wäre zu groß.
[96] siehe Abschnitt 5.2.2.
[97] z.B. kann SSL (siehe Abschnitt 5.2.2) nicht mit UDP verwendet werden.
[98] Zur Zeit ist unter dem Namen SNMPv3 ein neuer Versuch zur Beseitigung der Defizite von SNMP in Arbeit.

5.3 *Network Computing*-Referenzmodell

Als Ergebnis dieses Kapitels sind in der folgenden Tabelle noch einmal alle in meinem *Network Computing*-Referenzmodell relevanten Technologien zusammengefaßt. Anwendungen im Rahmen des *Network Computing* sollten auf diesen Technologien basieren, damit sie den Anforderungen der digitalen Wirtschaft gerecht werden können.

Präsentation	
Hypertext Markup Language	HTML
Multipurpose Internet Mail Extensions	MIME
Extensible Markup Language	XML
Portable Doucument Format	PDF
Graphics Interchange Format	GIF
JPEG	
Abstract Windowing Toolkit	Java/AWT
Persistenz	
Structured Query Language	SQL
Call Level Interface	SQL/CLI
Open Database Connectivity	ODBC
Java Database Connectivity	JDBC
SQLJ	
Transport	
TCP/IP	
Hypertext Transfer Protocol	HTTP
Internet-Inter Orb Protocol	CORBA/IIOP
Kollaboration	
Simple Mail Transfer Protokoll	SMTP
Post Office Protokoll	POP3
Net News Transfer Protokoll	NNTP
Internet Relay Chat	IRC
File Transfer Protokoll	FTP
Verzeichnisse und Lokalisierung	
Domain Name System	DNS
Lightwight Directory Access Protocol	LDAP
Uniform Resource Locator	URL

Sicherheit	
Zertifikate	X.509v3
Secure Socket Layer	SSLv3
Integration	
Common Gateway Interface	CGI
Message Queuing	MQI
Common Object Request Broker	CORBA/IDL
Transaktionen	
Distributes Transaction Processing Model	DTP TX/XA
Object Transaktion Service	CORBA/OTS
Programmierung	
Java	
JavaScript	
Komponenten	
JavaBeans	Java/JavaBeans
Enterprise JavaBeans	Java/EJB
Common Object Request Broker	CORBA
Management	
Simple Network Management Protocol	SNMPv2

Tab. 5.8: *Network Computing*-Refernzmodell

Die Beziehungen zwischen den einzelnen Konzepten und Technologien sind noch einmal in den Abbildungen 5.30 und 5.31 dargestellt. Man erkennt deutlich die einzelnen Schwerpunkte: Im Bereich **Communication** beherrschen TCP/IP basierte Protokolle das Bild; die **Computing** Dienste werden vor allem durch Java-Technologien erbracht; HTML mit den zughörigen MIME-Typen liefert die Grundlage für die Dienste des Bereichs **Content**.

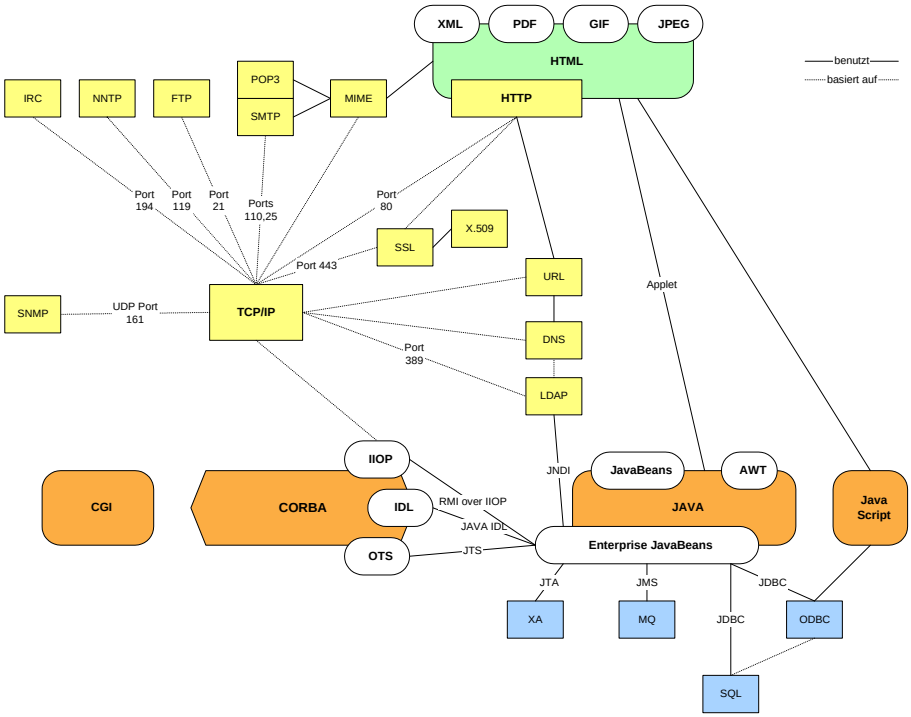

Fig. 5.30: Zusammenhänge im *Network Computing*-Referenzmodell

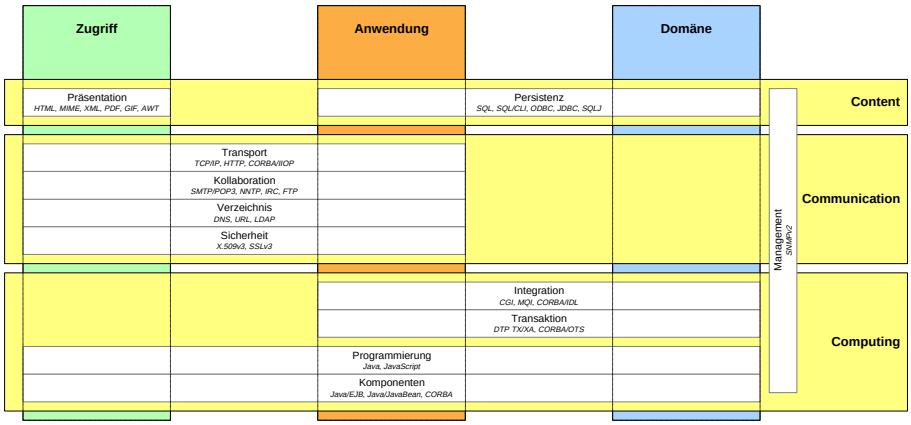

Fig. 5.31: Gesamtübersicht des *Network Computing*-Referenzmodells

5.4 Ein Szenario

Im nun folgenden Szenario dient eine Anwendung aus dem Bereich der *Local Marketing Services* als Beispiel für ein Anwendungssystem im Bereich der digitalen Wirtschaft.

5.4.1 Local Marketing Service

Unter *Local Marketing Service* versteht man eine Dienstleistung, mit der ein Hersteller seine Händler im Vertrieb unterstützt. In diesem komplexen, organisatorischen System agieren folgende Beteiligte:

- Der **Hersteller** bietet seinen Händlern eine Palette von produktbezogenen Werbemitteln wie Prospekte, Folder etc. und allgemeine Werbemitteln wie Displays, Poster etc. im *Corporate Design* an.
- Die **Händler** können über ein Werbemittelhandbuch einzelne Werbemittel für ihren lokalen Bedarf bestellen. Diese Werbemittel können nach Bedarf z.B. mit der Adresse des Händlers individualisiert werden.
- Die Lagerhaltung und Distribution sowie die Individualisierung einzelner Werbemittel erfolgt über einen speziellen **Dienstleister**, der auch die Abrechnung der bereitgestellten Werbemittel gegenüber den Händlern vorbereitet. Außerdem konzipiert er zusammen mit dem Hersteller das Werbemittelhandbuch sowie ein Abwicklungskonzept und betreibt ein *Call Center* für die Entgegennahme telefonischer Anfragen.
- Zusätzlich wird es **Endkunden** ermöglicht, Informationsmaterial wie Prospekte oder Kataloge zu den Produkten des Herstellers zu bestellen.

In Abbildung 5.32 sind die Beziehungen zwischen den einzelnen Beteiligten dargestellt.

Es gelten dabei folgende Anforderungen an das System: Der Hersteller muß die Möglichkeit haben, die aktuellen Werbemittelbestellungen seiner Händler zu überwachen und gegebenenfalls zu verändern. Die Händler müssen sich telefonisch nach dem aktuellen Stand ihrer Bestellungen erkundigen können. Die Informationen über die Adressen von Endkunden und evtl. abgefragte Marktforschungsdaten sollen regelmäßig an den Hersteller übermittelt werden.

5.4.2 Klassisches Kommunikationskonzept

Das klassische Kommunikationskonzept eines solchen Systems ist in Abbildung 5.33 dargestellt und läßt sich wie folgt beschreiben:

Der Dienstleister betreibt ein Warenwirtschaftssystem für die Auftragsabwicklung und Lagerverwaltung. Die Sachbearbeiter des Dienstleisters nehmen die Bestellungen der Händler telefonisch über ein *Call Center* oder schriftlich über Telefax oder Briefe entgegen und erfassen die entsprechenden Aufträge an ihren Arbeitsplätzen.

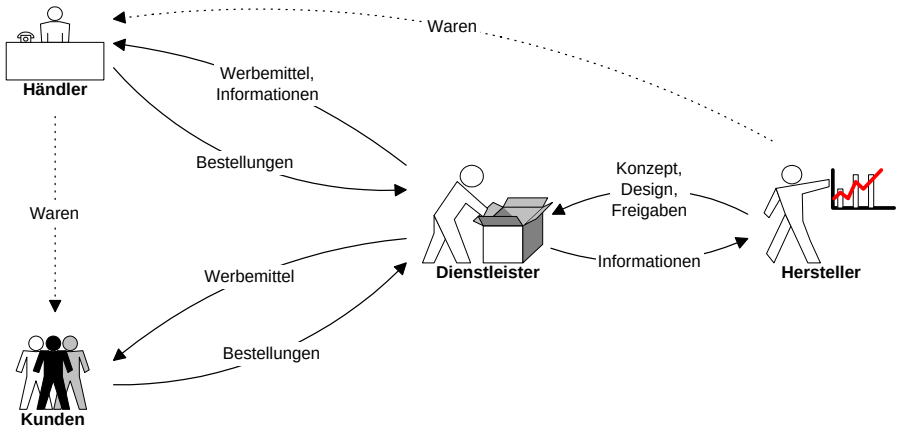

Fig. 5.32: Beziehungen im *Local Marketing Service*-System

Die Abrechnungsdaten werden dem Hersteller regelmäßig in einem proprietären Format auf einem Datenträger per Post zugeschickt. Dieser übernimmt sie anschließend in sein Fakturierungssystem. Lagerbestandsauskünfte werden regelmäßig in Form von Bestandslisten erstellt und dem Hersteller zugeschickt. Für Notfälle steht eine telefonische Auskunft zum aktuellen Status des Lagers und der Aufträge zur Verfügung. Eine Möglichkeit zur Beeinflussung der Bestellungen seiner Händler hat der Hersteller nur sehr eingeschränkt über den telefonischen Kontakt zu seinem Dienstleister. Die Adreßinformationen zu den Endkundenkontakten werden der Marketingabteilung des Herstellers regelmäßig auf einem Datenträger übermittelt und dort in die Direktmarketing Datenbank eingespielt.

Allerdings offenbart dieses Kommunikationskonzept einige Nachteile: So haben Hersteller kaum eine Möglichkeit, effektiv die Werbemittelbestellungen ihrer Händler zu koordinieren. Die benötigten Informationen sowie die daraus resultierenden Anweisungen an den Dienstleister sind über die klassischen Kommunikationswege deutlich zu langsam auszutauschen. Der Datenträgeraustausch zur Übermittlung der Abrechnungs- und Direktmarketingdaten an den Hersteller bindet dabei einen nicht unerheblichen Teil der Arbeitszeit in den betroffenen Abteilungen für das Erstellen, Versenden, Entgegennehmen und Einspielen der Daten. Viele Informationswünsche der Endkunden können nur pauschal über den Versand des entsprechenden Katalogs befriedigt werden. Details zu ihren Anfragen können nicht über die Datei-Schnittstelle übertragen werden. Die händlerbezogene Individualisierung einzelner Werbemittel gestaltet sich durch die Notwendigkeit des Austausches von Autorenkorrekturen[99] zur Freigabe durch den Händler recht umständlich.

[99] Vorabversionen der individualisierten Werbemittel

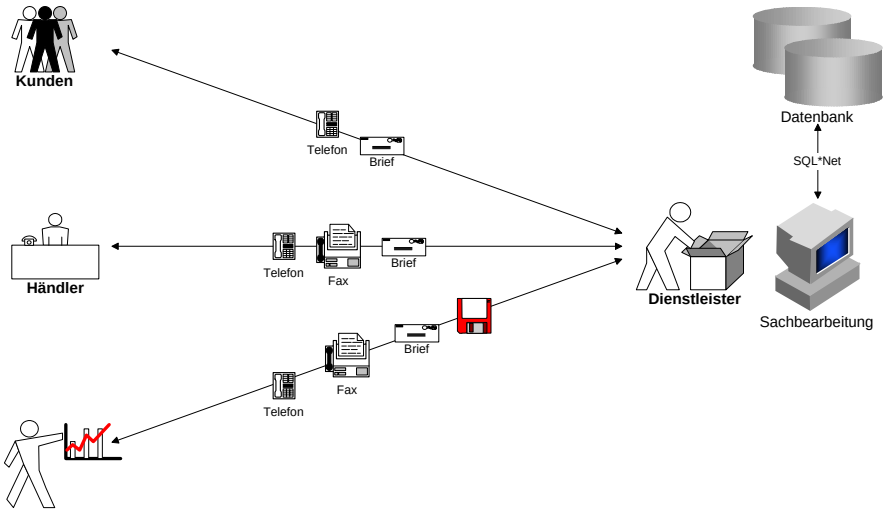

Fig. 5.33: Klassisches Kommunikationskonzept im *Local Marketing Service*

5.4.3 Einsatz von Konzepten der digitalen Wirtschaft

Im folgenden werde ich nun die Vision eines *Local Marketing Services* unter Berücksichtigung der Konzepte aus Kapitel 4 erstellen. Durch die Anwendung von Konzepten der digitalen Wirtschaft sollen dabei

- Kosteneinsparungen im Handling durch Straffung der Prozeßkette,
- ein innovatives Image des Herstellers,
- der Ausbau der Händlerunterstützung,
- die Beschleunigung des Informationsaustausches
- und ein verbesserter Kundenservice

erreicht werden.

Diese Vision umfaßt dabei Aktivitäten bis hin zur Ebene III der digitalen Wirtschaft[100].

Auf Kundenseite führt dies zu folgenden Veränderungen: Zunächst muß ein Bereich im *World Wide Web* aufgebaut werden, in dem sich potentielle Kunden jederzeit über die Produkte des Herstellers informieren können (*Ebene I*). Darauf aufbauend muß ihnen ermöglicht werden, weiterführende Informationen zu erhalten, oder den Kontakt zu einem Ansprechpartner des Herstellers aufzunehmen (*Ebene II*).

Auf Seiten der Händler bieten sich folgende Veränderungen an: Den Händlern wird eröffnet, über einen besonders geschützten Bereich des WWW-Angebotes

[100] siehe Abschnitt 4.1.2.

des Herstellers, der sich allerdings auf den Systemen des Dienstleisters befindet, ihre Werbemittelbestellungen online zu erfassen (*Ebene III*). In diesem Rahmen werden ihnen bereits bei der Erfassung ihrer Aufträge die aktuellen Lagerbestände der Werbemittel angezeigt und somit die Entstehung von Rückstellungen für nicht erfüllbare Bestellmengen minimiert. Außerdem können die Händler sich jederzeit über den aktuellen Status ihrer Bestellung informieren. Die Indiviualisierung der Werbemittel kann nun vom Händler direkt am Bildschirm vorgenommen werden - umständliche Abstimmungsverfahren können entfallen.

Der Hersteller erhält nun auch die Möglichkeit, sich jederzeit über den aktuellen Stand von Aufträgen und Lagerbeständen zu informieren. Er wird nunmehr in die Lage versetzt, einzelne Aufträge seiner Händler korrigieren zu können. Die Übermittlung der Fakturierungsdaten kann jetzt auf direktem elektronischen Weg erfolgen, ebenso die der Direktmarketinginformationen aus den Bestellungen und Anfragen der Endkunden.

Damit entfallen für den Dienstleister eine Vielzahl unproduktiver Tätigkeiten etwa im Bereich der Information, Abstimmung und Abwicklung. Zudem wird wird die Auftragserfassung durch den Wegfall eines Teils der schriftlichen und telefonischen Anfragen deutlich entlastet.

Unter Berücksichtigung des *Network Computing*-Profils aus Abschnitt 4.3 würde sich somit folgendes Lösungskonzept ergeben:

CONTENT	
Präsentation	Die Produktinformationen werden im PDF-Format zur Verfügung gestellt. Die Werbemittel-Anwendung wird auf der Basis von HTML und JavaScript erstellt. Die Abrechnungs- und Direktmarketingdaten werden über ein XML-Format ausgetauscht.
Persistenz	Die Direktmarketingdaten sowie die PDF-Dateien werden über ODBC in einer relationalen Datenbank abgelegt.
COMMUNICATION	
Transport	Die Kommunikation zwischen den einzelnen Beteiligten wird mittels HTTP über TCP/IP abgewickelt.
Kollaboration	Für die Kommunikation zwischen den Mitarbeitern des Dienstleisters, der Händler und des Herstellers sowie mit den Endkunden wird eMail über SMTP/POP3 eingesetzt. Die PDF-Dateien der Produktinformationen werden mittels FTP übertragen.
Verzeichnis	Der Server des WWW-Auftritts des Kunden wird im DNS eingetragen. Dies geschieht auch für den Mail-Server, unter dem die Ansprechpartner für die Kunden erreichbar sein werden.

Sicherheit	Die einzelnen Händler identifizieren sich über ihre Zertifikate bei der Werbemittel-Anwendung. Dies gilt auch für die Mitarbeiter des Herstellers. Die Übertragung von Informationen im Rahmen der Werbemittel-Anwendung wird über SSL verschlüsselt. Die Übertragung von Daten aus den Aktivitäten der Ebenen I und II erfolgt unverschlüsselt. Die drei Altsysteme werden durch Firewalls geschützt. Zwischen ihnen wird ein Virtuelles Privates Netzwerk (VPN) aufgebaut.
COMPUTING	
Integration	Die Integration des Warenwirtschaftssystems wird über CORBA abgewickelt, die Anbindung der Systeme des Herstellers erfolgt über jeweils eine *Message Queue*.
Transaktion	Die Transaktionen im Warenwirtschaftssystem werden über die XA-Schnittstelle abgesichert.
Programmierung	Die Programmierung der Feldprüfungen erfolgt in JavaScript. Die Anbindung des Warenwirtschaftssystems sowie die Realisierung der Händlerverwaltung erfolgt in Java.
Komponenten	Als Komponentenmodell werden die *Enterprise JavaBeans* eingesetzt.
META	
Management	Zur Überwachung und Kontrolle der Werbemittelanwendung und der dafür benötigten technischen Komponenten wird SNMP eingesetzt.

Tab. 5.9: *Network Computing*-Vision des *Local Marketing Service*

Das Ergebnis ist ein Extranet[101] aus Händler, Hersteller und Dienstleister sowie ein Internet mit Kunden, Hersteller und Dienstleister. Das entsprechende Kommunikationskonzept ist in Abbildung 5.34 dargestellt.

Mit diesem Beispiel endet das Kapitel und der Teil der Arbeit, der sich mit dem Wesen des *Network Computing* beschäftigt hat. Für die Beantwortung der ersten Frage - *Was verbirgt sich konkret hinter dem Konzept des Network Computing ?* - habe ich zunächst die digitale Wirtschaft anhand von Beispielen erläutert und Möglichkeiten zur Gliederung der Konzepte entwickelt, anschließend besondere aktuelle Aspekte betrieblicher Informationssysteme beschrieben. Im Haupteil dieses Kapitels habe ich mich dann der Entwicklung eines *Network Computing*-Referenzmodells gewidmet. Nach Definition der Kriterien und der Formulierung

[101] siehe Abschnitt 4.2.1.

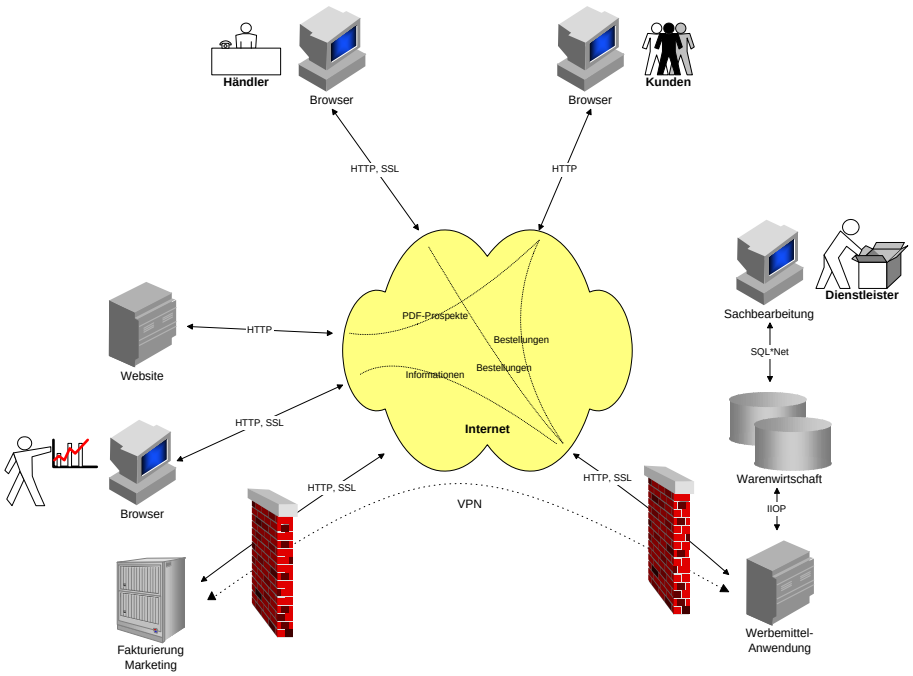

Fig. 5.34: Kommunikationskonzept unter Nutzung von *Network Computing*-Technologien

eines groben Verteilungs- und Dienstmodells habe ich den aktuellen Stand aller relevanten Technologien erläutert und diese miteinander in Zusammenhang gesetzt. Das Ergebnis ist eine Auswahl von Technologien und Konzepten, die die technologische Basis für die digitale Wirtschaft bilden - mein *Network Computing*-Referenzmodell. Abschließend habe ich an einem Beispiel aus dem Marketingbereich mögliche Veränderungen durch den Einsatz von Konzepten der digitalen Wirtschaft aufgezeigt sowie eine Lösungsvision auf der Basis des *Network Computing*-Referenzmodells entwickelt.

Dieses Referenzmodell dient dabei der Definition des Begriffs *Network Computing*. Zudem untermauert die Auswahl der Dienste und die Festlegung auf bestimmte Dienstausprägungen meine Definition des *Network Computing* als technologische Basis der digitalen Wirtschaft auf der Grundlage von Internet-Technologien und unter Integration der Inhalte existierender Anwendungen.

Im folgenden Kapitel werde ich nun untersuchen, ob mit dem aktuell etablierten Stand der Technik von Datenbankmanagementsystemen diese Vision auch realisiert werden kann und damit die zweiten Frage beantworten: *Welche Rolle spielen Datenbanksysteme im Rahmen des Network Computing ?*

Kapitel 6

NETWORK COMPUTING DATENBANKSYSTEM

Nachdem ich in Kapitel 5 den Begriff *Network Computing* über ein entsprechendes Referenzmodell definiert habe, werde ich in diesem Kapitel prüfen, in wieweit Datenbanksysteme in diesem Umfeld eine Rolle spielen.

Dazu werde ich - dem grundsätzlichem Kapitelaufbau dieser Arbeit folgend - zunächst auf einer abstrakten Ebene einen Vergleich des Datenbankprofils aus Kapitel 2 mit dem Infrastrukturprofil aus Kapitel 4 vornehmen. Das Infrastrukturprofil ist dabei entsprechend dem Ergebnis des Kapitels 5 mit einem *Network Computing*-Profil gleichzusetzen.

Anschließend wird das Ergebnis in bezug auf das *Network Computing*-Referenzmodell konkretisiert. Dazu stelle ich die *Oracle Network Computing Architecture* (Oracle NCA) als Beispiel für den aktuell etablierten Stand der Datenbanktechnologie vor und bewerte sie mit Hilfe des *Network Computing*-Referenzmodells. Das Ergebnis dieses Schritts ist die Beantwortung der zweiten Frage dieser Arbeit.

6.1 Profilvergleich

Die Frage nach der Rolle von Datenbanksystemen im Rahmen des *Network Computing* läßt sich prinzipiell durch einen Vergleich der benötigten mit den bereitgestellten Diensten beantworten. Die benötigten Dienste entsprechen dabei dem Infrastrukturprofil des Kapitels 4, die bereitgestellten Dienste entnehme ich dem Datenbankprofil aus Kapitel 2. Tabelle 6.1 stellt - in teilweise gekürzter Fassung - die beiden Profile einander gegenüber.

Dienst	Network Computing	Datenbank
Präsentation	Die Präsentationsdienste ermöglichen die Darstellung der Inhalte an der Bedienerschnittstelle und die Steuerung der Interaktionen.	nicht vorhanden[1]
Persistenz	Dieser Dienst bietet die dauerhafte Ablage und bedarfsgerechte Wiedergewinnung der Inhalte sowie der Informationen, die für den laufenden Betrieb der Anwendungen benötigt werden.	Dieser Dienst bietet den langfristigen Zugriff auf große Datenmengen unter Bereitstellung von Datenzugriffsoperationen über entsprechende Dienstschnittstellen wie z.B. Abfragesprachen.
Transport	Der Transportdienst dient der Übermittlung von Inhalten und Interaktionen zwischen den Kommunikationspartnern. Er muß mit dem Sicherheitsdienst kompatibel sein, um den entsprechenden Anforderungen genügen zu können.	Die Kommunikation zwischen Anwendungsprogramm und Datenbanksystem im Rahmen der Verteilungsaufgaben wird über Transportprotokolle und zugehörige interne Dienste des Datenbanksystems realisiert.
Kollaboration	Zu dem Bereich der Kollaborationsdienste zählen alle Dienste, die das gemeinsame Arbeiten von Personen unterstützen, wie elektronische Post, schwarze Bretter und der Austausch von Dateien.	nicht vorhanden
Verzeichnis	In verteilten Umgebungen bedarf es standardisierter Verzeichnisse, um sämtlichen Kommunikationspartnern in Systemen der digitalen Wirtschaft über einheitliche Schnittstellen das Lokalisieren von Personen, Diensten und Ressourcen zu ermöglichen.	Der Verzeichnisdienst verwaltet die Datenbankobjekte wie Benutzer, Schemadefinitionen u.s.w. und bietet somit einen Zugriff auf die Metadatenverwaltung des Datenbankmanagementsystems.

[1] Eine Unterstützung ist nur durch die Möglichkeit zur Speicherung der Datentypen der Bedienerschnittstelle gegeben.

Sicherheit	Dieser Dienst sichert zu, daß die Vertraulichkeit und Authentizität der verarbeiteten und übertragenen Information gewahrt bleibt und die Infrastruktur vor unautorisierten Zugriffen geschützt ist. Er ermöglicht vorallem die Identifizierung von Kommunikationspartnern und die verschlüsselte Übertragung von Inhalten unter Sicherstellung der Integrität der übermittelten Informationen.	Dieser Dienst realisiert die Identifizierung von Benutzern über Benutzername und Kennwort sowie die Regelung des Zugriffs auf Datenbankobjekte entsprechend meist deklarativer Beschreibung von Zugriffsregeln über die Dienstschnittstelle des Persistenzdienstes. Außerdem kann über diesen Dienst die Nutzung der Dienstschnittstelle des Persistenzdienstes protokolliert und überwacht werden.
Integration	Der Integrationsdienst stellt Möglichkeiten zur Anbindung existierender Informationssysteme an Systeme der digitalen Wirtschaft zur Verfügung. In diesen Dienst fallen Möglichkeiten zur synchronen und asynchronen Kopplung von Informationssystemen.	nicht vorhanden[2]
Transaktion	Der Transaktionsdienst bietet eine robuste, skalierbare und transaktionale Verarbeitungsumgebung für die Anwendungen der digitalen Wirtschaft. Dazu gehört unter anderem die Bereitstellung von verteilten Transaktionen und Möglichkeiten zur transparenten Lastverteilung.	Die Aufgaben der Fehlererholung, Integritätssicherung und Synchronisation im Rahmen der Zuverlässigkeit werden durch einen Transaktionsdienst realisiert.

[2] Die Integration von Anwendungen wird über ein gemeinsames Datenbankschema vorgenommen.

Programmierung	Der Programmierdienst stellt Programmier- und Skriptsprachen zur Implementierung von einzelnen problemorientierten Anwendungen zur Verfügung.	Dieser Dienst bietet die Möglichkeit, Anwendungsfunktionen in Form von Datenbankprozeduren oder Datenbanktriggern über eine Datenbankprogrammiersprache zu definieren.
Komponenten	Der Komponentendienst ermöglicht den Einsatz von Komponenten für die Erfüllung von problemorientierten, aber auch systemnahen Aufgaben.	nicht vorhanden
Management	Der Bereich der Managementdienste umfaßt die Verwaltung, Überwachung und Kontrolle aller Elemente der Infrastruktur.	Für die Verwaltung des Datenbanksystems müssen Managementdienste zur Verfügung stehen, die Informationen über den Zustand des Systems bereitstellen und die Beeinflussung einzelner Systemkomponenten erlauben.

Tab. 6.1: Profilvergleich

Diese Gegenüberstellung zeigt, daß die Dienste von Datenbanksystemen entsprechend dem Datenbankprofil einen großen Teil der Aufgaben des Infrastrukturprofils erfüllen können:

- Persistenz,
- Transport,
- Verzeichnis,
- Sicherheit,
- Transaktion,
- Programmierung sowie
- Management.

Für die folgenden Dienste kann eine Datenbank entsprechend dem Datenbank-Profil standardmäßig keine Dienste anbieten:

- Präsentation,
- Kollaboration,
- Integration sowie
- Komponenten.

Daß keine Unterstützung im Bereich der Präsentationsdienste vorliegt, ist wenig verwunderlich. Das Datenbanksystem sollte aber in der Lage sein, die präsentierten Informationen durch seinen Persistenzdienst sicher verwalten zu können. Kollaborationsdienste werden zwar meist datenbankgestützt realisiert, jedoch sind die Dienste selbst nie Bestandteil der Datenbanksysteme gewesen. Ein Integrationsdienst wurde bei der monolithischen Grundausrichtung des klassischen Datenbankbegriffs nach Kapitel 2 nicht benötigt. Allerdings haben sich, wie wir noch sehen werden, in diesem Bereich Erweiterungen der bestehenden Datenbanksysteme entwickelt. Ein Komponentenmodell im Sinne von Abschnitt 5.2.3 ist in dem hier untersuchten klassischen Datenbankprofil nicht anzutreffen, weil eine Erweiterung der Funktionalität der Systeme nur durch den Hersteller selbst vorgesehen war.

Zusammenfassend betrachtet läßt sich wie in Abbildung 6.1 feststellen, daß das *Network Computing*-Profil eine Obermenge des Datenbankprofils ist. Das klassische Datenbankprofil nach Kapitel 2 erbringt demnach ca. zwei Drittel der geforderten Dienste.

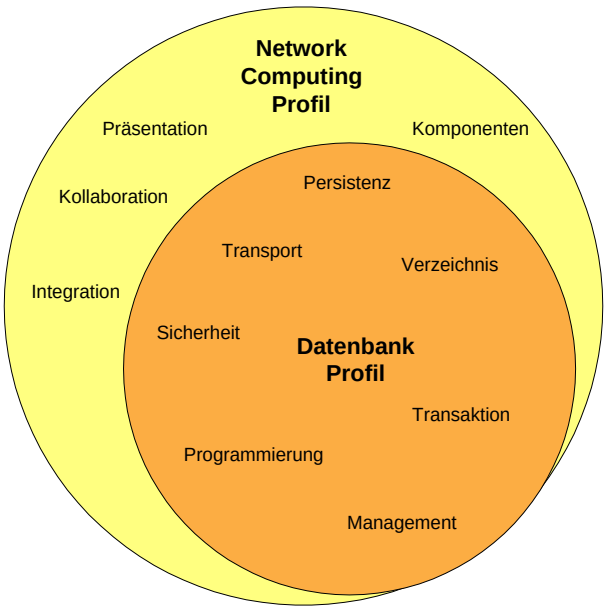

Fig. 6.1: Profilvergleich

Zur Beantwortung der Frage nach der Bedeutung von Datenbanksystemen im Rahmen des *Network Computing* reicht es jedoch nicht aus, nur zu fragen: Wer-

den die benötigten Dienste erbracht ? Es muß vor allem die Frage gestellt werden: In welcher Form werden diese Dienste erbracht? Um dies zu untersuchen werden ich im folgenden Abschnitt den Stand aktuell etablierter Datenbanktechnologie am Beispiel der Oracle NCA beschreiben und die bereitgestellten Dienste mit dem von mir entwickelten *Network Computing*-Referenzmodell prüfen. Dabei wird sich zeigen, daß der Umfang der durch aktuelle Systeme bereitgestellten Dienste die klassischen Datenbankdienste inzwischen deutlich übertrifft.

6.2 Die Oracle NCA

Die Oracle Corporation benutzt als Slogan die Aussage *Enabling the Information Age*. Damit soll zum Ausdruck gebracht werden, daß sich das Unternehmen als Lieferant aller benötigten Technologien[3] für die Realisierung der Vision vom Informationszeitalter sieht.

Das zeigt sich auch in der Produktstrategie, die unter dem Begriff *Network Computing Architecture* vermarktet wird. Oracle beschreibt seine Architektur mit den Worten:

> *Eine umfassende, offene, netzwerk-basierte Architektur, welche Erweiterbarkeit für verteilte Umgebungen bereitstellt.*

In diesem Abschnitt werde ich einen Überblick über die *Oracle Network Computing Architecture* (Oracle NCA) [Ora97a] geben. Die Oracle NCA kombiniert drei architekturelle Ansätze: mehrstufiges Client/Server, Internet und verteilte Objekt-Architekturen. Die in Abbildung 6.2 dargestellte, zugrundeliegende Infrastruktur läßt sich in sechs Bereiche eingeteilen:

- *Cartridges*,
- *Inter-Cartridge Exchange (ICX)*,
- Standarddienste,
- Clienten,
- Datenbankserver und
- Anwendungsserver.

Zusätzlich zu diesen Bereichen gibt es noch eine Entwicklungs- und Managementumgebung, auf die ich hier nicht weiter eingehen werde[4]. Im folgenden werde ich die einzelnen Bestandteile der Architektur kurz beschreiben.

6.2.1 Bestandteile des Systems

Cartridges

Cartridges sind Oracles Komponenten. Sie erweitern die Funktionalität von Clienten, Anwendungsservern und Datenbankservern. Anwendungen werden über das Zusammenspiel mehrerer dieser Cartridges realisiert. Je nach Ebene unterscheidet man:

- *Client-Cartridges*, die bestimmte Benutzerschnittstellenfunktionalitäten realisieren[5].

[3] Im Fokus wird jedoch berichtet, daß Larry Ellison den Hardware Bereich - Network Computer - aus der Produktpalette Oracles gestrichen hat.

[4] Einzelne Aspekte werden in der Diskussion der Dienste der NCA im Vergleich zum *Network Computing*-Referenzmodell angesprochen.

[5] vergleichbar mit den heutigen Browser-PlugIns oder JavaApplets.

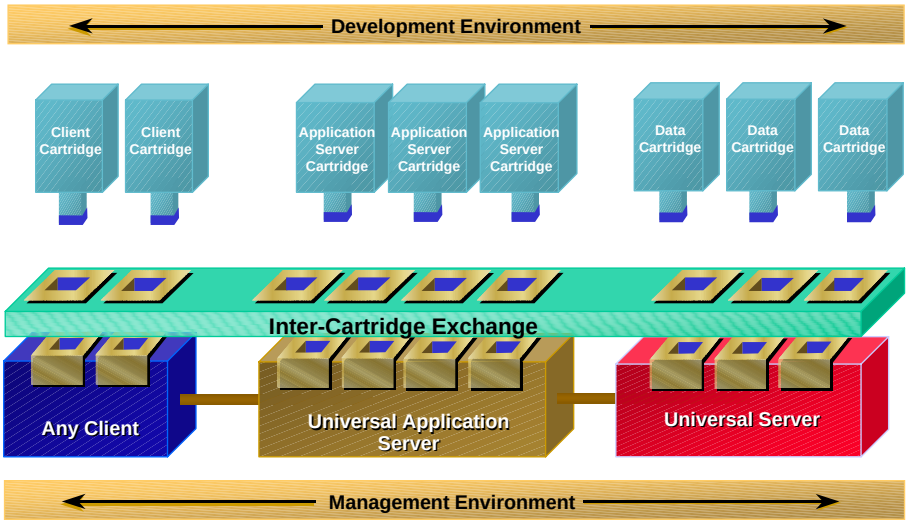

Fig. 6.2: Die Oracle *Network Computing Architecture*

- *Application-Cartridges* beinhalten die Anwendungslogik. Diese Komponenten werden auf dem Anwendungsserver eingesetzt und realisieren bestimmte fachliche Funktionen[6].
- *Data-Cartridges* erweitern die Funktionalität des Datenbankservers[7].

Ein Cartridge stellt seine Funktionalität über Schnittstellen zur Verfügung. Einige dieser Schnittstellen sind generisch[8], andere sind aufgabenspezifisch[9]. Ein Cartridge besteht aus mehreren internen Objekten, die die Schnittstellen implementieren sowie einigen Metadaten für die Verwaltung des Cartridge.

Für die Entwicklung von serverseitigen Anwendungen stehen eine Reihe von Standard-Cartridges zur Verfügung, die die Programmierung in Java, C, Perl oder PL/SQL ermöglichen.

Inter-Cartridge Exchange (ICX)

Die ICX ist das auf HTTP basierende Protokoll zur Kommunikation zwischen den Cartridges. Ein Cartridge, das eine ICX-Anfrage erhält, kann keinen Unter-

[6] Es gibt bereits eine Reihe von Drittanbietern, die Cartridges für elektronischen Zahlungsverkehr (`http://www.verifone.com`) oder Steuerberechnungen (TaxWare) anbieten. Diese Art von Komponenten können auch genutzt werden, um Altsysteme in die Architektur zu integrieren. Ein Beispiel wäre hier das *WebConnect-Cartridge* (OpenConnect), das eine 3270-Emulation über einen Web-Browser zur Verfügung stellt.

[7] Damit sind sie mit Informix-*DataBlades* vergleichbar. Allerdings patchen die *DataBlades* den Datenbank-Kernel, während die Data-Cartridges über das Komponentenmodell integriert werden. Auch hier gibt es bereits Lösungen von Fremdherstellern, wie z.B. zur Bildverarbeitung (Virage) oder Volltextsuche (ConText).

[8] z.B. für die Installation oder Instantiierung des Cartridge.

[9] z.B. bietet ein Web-Cartridge eine Schnittstelle für die Bearbeitung von HTTP-Requests.

schied zu einer Standard-HTTP-Anfrage erkennen. Dadurch sind alle Cartridges standardmäßig in der Lage, auf einen solchen Request zu reagieren.

Standarddienste

Für die Verwaltung und Nutzung der Cartridges werden eine Reihe von Diensten bereitgestellt. Diese reichen von den *Universal Cartridge Services* (Verwaltung des Cartridge-Lebenslaufs) über die *Scalable Cartridge Services* (z.B. Transaktionen, Datenbankzugriffe) bis hin zu *Specialised Cartridge Services* (z.B. Lastverteilung, Fehlererholung).

Clienten

Die NCA unterstützt ein breites Spektrum an Clienten. Die minimale Anforderung an die Clienten entspricht der Spezifikation des Network Computers [OG98].

Anwendungsserver

Der Anwendungsserver dient als Plattform für den Einsatz von Anwendungen in Form von Anwendungs-Cartridges. Er basiert auf einem ORB und bietet unter anderem Dienste in den Bereichen Transaktionsmanagement, Lastverteilung und Sicherheit.

Datenbankserver

Der Datenbankserver dient der robusten und skalierbaren Verwaltung von Massendaten. Zusätzlich zu klassischen relationalen Daten bietet er Möglichkeiten zur Erweiterung seines Typsystems über Data-Cartridges. Seine Dienste umfassen neben den Persistenzdiensten auch Transaktionsmanagement, Sicherheit und Message Queuing.

6.2.2 Beschreibung ausgewählter Mechanismen

Inter-Cartridge Exchange

Zur Beschreibung dieses Bestandteils der Architektur verwende ich folgendes Beispiel:

Eine PL/SQL-Prozedur ruft über die ICX ein Perl-Skript, das eine eMail verschickt[10].

Da die ICX zur Zeit nur HTTP als Kommunikationsmittel nutzen kann, erfolgt der Aufruf des Perl-Cartridge über das PL/SQL-Package `utl_http` folgendermaßen:

[10] Es wäre auch möglich, die eMail über die Oracle-eigene Groupwarelösung Oracle Office zu versenden, die entsprechende PL/SQL-Prozeduren anbietet.

```
vcResult := utl_http.request('http://www.dienstleister.de'||
'/lms/perl/sendConfirmation.pl?KD_ID=4711');
```

Dadurch wird ein, unter `/lms/perl` installiertes Perl-Cartridge aufgerufen und der Parameter `KD_ID` mit dem Wert `4711` übergeben. Das Ergebnis des Aufrufs wird als Zeichenkette in die Variable `vcResult` geschrieben.

Transaktionen

Der Anwendungsserver unterstützt deklarative und programmatische Transaktionsklammern. Deklarative Transaktionen werden zur Laufzeit durch Konfigurationsoptionen der Cartridges deklariert. Programmatische Transaktionen werden durch Aufrufe der jeweiligen APIs definiert. In Tabelle 6.2 sind die unterschiedlichen Transaktionstypen in Abhängigkeit von der eingesetzten Programmiersprache aufgeführt.

Programmier-sprache	**Transaktions-typ**	**Beschreibung**
C	deklarativ	über virtuelle Pfade, die jeweils ein BEGIN, COMMIT, ROLLBACK bezeichnen
	programmatisch	über die DTP TX-Schnittstelle
Java	deklarativ	über das Setzen bestimmter Attributwerte
	programmatisch	über den JTS
PL/SQL	deklarativ	über URLs, die automatisch jeweils ein BEGIN, COMMIT, ROLLBACK auslösen

Tab. 6.2: Transaktionstypen

Kontrollfluß einer JWeb-Anfrage

Das JWeb-Cartridge ermöglicht die Entwicklung von serverseitigen Java Anwendungen unter Verwendung von HTTP als Kommunikationsprotokoll. Es basiert auf dem Web Request Broker[11]. In Abbildung 6.3 ist ein Interaktionsdiagramm für den Aufruf einer Java-Klasse über das JWeb-Cartridge angegeben.

Der Listener empfängt (1) über HTTP eine Anfrage mit der URL `http://as.xenion.de/myJWeb/HelloWorld`. Der Dispatcher erkennt an der URL, daß es sich um eine Anfrage an ein Cartridge handelt und leitet (2) den Request an den Web Request Broker weiter. Auch dieser analysiert die URL. Da der virtuelle Pfad `myJWeb` in der Konfiguration des Anwendungsservers auf ein JWeb-Cartridge abgebildet wurde, leitet (3) der WRB die Anfrage an ein JWeb-Cartridge weiter. Dieses ermittelt aus dem Rest der URL den Namen der Java-Anwendung (`HelloWorld`), lädt (4) die entsprechende Klasse und

[11] im Gegensatz zum JCorba-Cartridge, welches auf dem Objekt Request Broker aufsetzt.

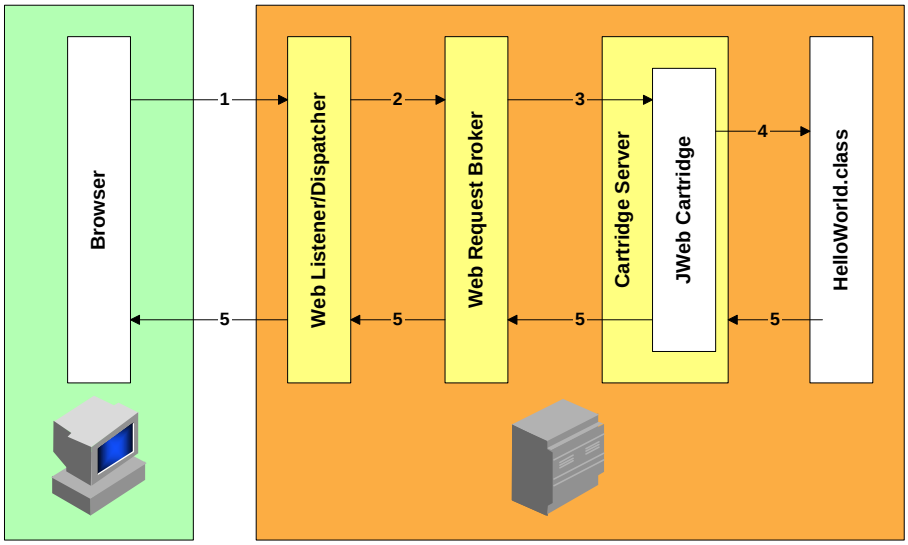

Fig. 6.3: Interaktionsdiagramm eines JWeb-Aufrufs

ruft den Einstiegspunkt `main()` auf. Die Java-Anwendung erzeugt eine Antwort[12]und sendet (5) diese über einen speziellen Ausgabestream `HtmlStream` an das JWeb-Cartridge zurück. Die Antwort wird anschließend über den WRB und den Listner an den Client übermittelt.

6.2.3 Die NCA als Datenbankmanagementsystem

Laut der Definition eines Datenbankmanagementsystem in Abschnitt 2.1.1 ermöglicht und organisiert dieses den Umgang mit der Datenbank.

Zunächst stellt sich hier also die Frage, ob der Anwendungsserver zum Datenbankmanagementsystem gezählt werden kann. Wenn man die in Tabelle 6.3 dargestellte Verteilung der Datenbankfunktionen aus Abschnitt 2.3.1 auf den Datenbankserver (DB) und den Anwendungsserver (AS) untersucht, läßt sich erkennen, daß der Anwendungsserver bis auf den Bereich der Persistenz einen Großteil der Datenbanksystem-Funktionalität ausführen kann.

Funktion	**DB**	**AS**
Langfristiger Zugriff auf große Datenmengen	X	
Metadatenverwaltung	X	
Operationen	X	
Datenunabhängigkeit	X	
Fehlererholung	X	X
Integritätssicherung	X	X
Synchronisation	X	X
Authentisierung	X	X

[12] bestehend aus HTTP-Response-Header und HTTP-Response-Body

Autorisierung	X	X
Verschlüsselung	X	X
Überwachung	X	X
Optimierung	X	X
Leistungssteuerung	X	X
Verteilung	X	X
Werkzeuge und Dienstschnittstellen	X	X

Tab. 6.3: Verteilung von Datenbank- und Anwendungsserver-Funktionalität

Die Aufteilung in Anwendungsserver und Datenbankserver ist aus meiner Sicht nur ein Marktingschachzug, um an der aktueller Anwendungsserver-Euphorie partizipieren zu können[13]. Es ist ohne Probleme möglich, den Anwendungsserver als Bestandteil des DBMS zu betrachten - die enge Verbindung und die teilweise gemeinsamen Dienste sind ein eindeutiges Indiez dafür. Der Kern des Oracle RDBMS selbst besteht schon länger aus mehreren Prozessen[14] . Es gibt also keinen Grund, die Prozesse des Anwendungsservers[15] seperat zu betrachten.

Der einzig neue Aspekt ist, daß diese Prozesse auch auf einer andern Maschine laufen können, dies ist bei den Kernprozessen des RDBMS nicht möglich[16]. Dieser Aspekt ermöglicht in des eine bessere Verteilung von Aufgaben und damit eine höhere Skalierbarkeit.

Somit ist die NCA mit ihren beiden Serverbestandteilen für mich ein Datenbankmanagementsystem. Die Struktur dieses Systems ist in Abbildung 6.4 dargestellt:

Der Zugriff auf das DBMS kann über IIOP, HTTP oder TNS erfolgen. Die HTTP- und IIOP-Zugriffe laufen über den Anwendungsserverteil, die TNS-Zugiffe über den Datenbankserverteil des DBMS.

Der Datenbankserverteil bietet über seine *Extensibility Services* zudem die Möglichkeit, unter Nutzung der Datenkernelservices seine Funktionalität über Datacartridges zu erweitern. Außerdem können über das *Procedural Gateway Message Queuing*-Systeme und über das *Transparent Gateway* Fremddatenquellen angeschlossen werden.

Der Anwendungsserverteil bietet die Möglichkeit, über die ORB-Schnittstelle oder die einfachere Web-Request-Broker-Schnittstelle Cartridges zu nutzen. Über beide Schnittstellen sind Dienste aus den Bereichen Transaktionsmanagement und Sicherheit nutzbar. Außerdem stehen eine Reihe von Standard-Cartridges bereit, mit denen eigene Anwendungen entwickelt werden können.

[13] Bei dem in Kürze verfügbaren Oracle8i Datenbanksystem wird diese Trennung ohnehin wieder zurückgenommen.

[14] *Recoverer, Process Monitor, System Monitor, Database Writer, Log Writer, Archiver.*

[15] *Configuration Provider, Log Server, Monitoring Daemon, Authentication Server, Authorization Host Server, Broker, Resource Manager Proxy, Cartridge Server, Virtual Path Manager, Cartridge Factory.*

[16] Es sei denn, man betrachtet die *Distributed Option* [Ora97i] des RDBMS aus dieser Perspektive.

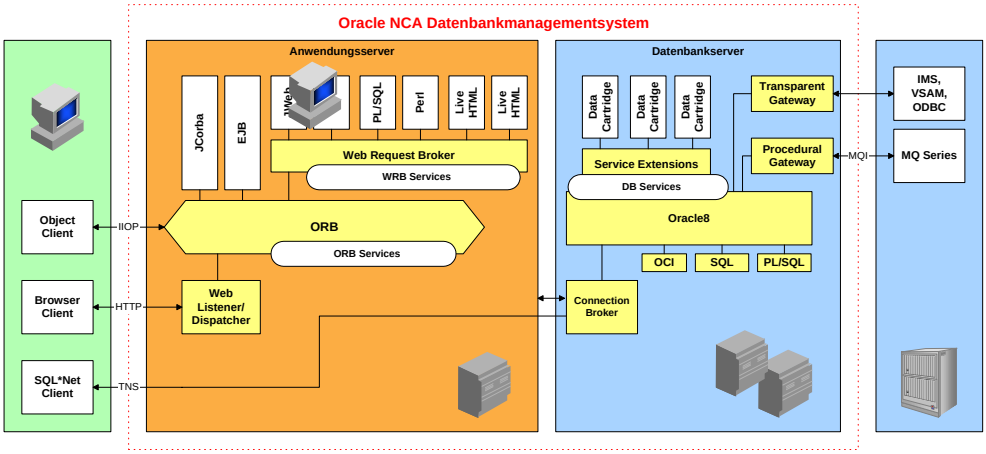

Fig. 6.4: Das Oracle NCA Datenbankmanagementsystem

Einzelne Details dieses DBMS werde ich in der nun folgenden Bewertung mit Blick auf das *Network Computing*-Referenzmodell diskutieren.

6.3 Bewertung in bezug auf das *Network Computing*-Referenzmodell

Um die Bedeutung der NCA im Rahmen des *Network Computing* beurteilen zu können, werde ich nun die Fähigkeiten der Oracle Systeme in bezug auf das *Network Computing*-Referenzmodell abprüfen. Dazu werde ich zu jedem Dienstbereich des *Network Computing*-Referenzmodells die entsprechenden Dienstangebote des Systems beschreiben. Die Informationen stammen aus der jeweiligen Dokumentation[17] sowie meinen Erfahrungen mit dem Einsatz der Systeme.

6.3.1 Content-Dienste

Präsentation

HTML/MIME Für die im Rahmen des Programmierdienstes bereitgestellten Programmiersprachen stehen Bibliotheken zur Unterstützung der Erstellung von HTML-Dokumenten zur Verfügung.

GIF/JPEG/PDF Für die Speicherung dieser Informationen stehen *Large Objects* (LOBs) zur Verfügung. Diese reichen von *Binary Large Objects* (BLOBs)[18] bis zu *Binary Files* (BFILE)[19].

[17] etwa [Ora98a, Ora98b, Ora98c, Ora97g].
[18] bis zu 4 GB groß.
[19] Dateien, die außerhalb der Datenbank gespeichert sind, bis zu 4 GB groß werden dürfen, aber read-only sind und nicht an Transaktionen teilnehmen.

XML Eine native Unterstützung von XML ist bei den hier untersuchten Servern noch nicht vorgesehen. Allerdings gibt es eine Suite von PL/SQL-Packages [Mue98], die grundlegende XML/XSL Funktionalitäten bereitstellen.

Persistenz

Der Zugriff auf die Datenbank kann über SQLJ[20], JDBC oder ODBC erfolgen.

SQL Oracles SQL ist kompatibel zum ISO SQL92 *entry level* Standard. Allerdings gibt es einige Oracle-spezifische Erweiterungen wie `INSTEAD OF`-Trigger[21] oder SQL3 *inline views*[22].

SQL/CLI Das *Oracle Call Interface* (OCI) [Ora97c] kann direkt in der Programmiersprache C angesprochen werden.

6.3.2 Communication-Dienste

Transport

HTTP/IIOP Als Transportprotokolle werden HTTP über den Listener und IIOP über den ORB unterstützt.

Inter-Cartridge Exchange (ICX) Die ICX ist der interne Kommunikationsmechanismus im Anwendungsserver. Sie basiert auf der Kommunikation mittels HTTP. Diese Art der Zusammenarbeit von Cartridges weist indes einige Probleme auf. Es kann nur ein Wert als Ergebnis zurückgeliefert werden. Soll es mehrere Rückgabeparameter geben, müssen diese innerhalb des einen Rückgabewerts kodiert werden[23]. Die ausgetauschten Parameter sind grundsätzlich vom Typ Zeichenkette - also quasi untypisiert. Außerdem ist die Größe der Zeichenkette auf 2000 Zeichen begrenzt[24]. Zudem gibt es derzeit auch noch keine Unterstützung von Java durch die ICX[25].

SQL*Net/Net8 Zusätzlich steht das proprietäre Protokoll *Transparent Network Substrate* (TNS) der Produkte SQL*Net und Net8 für die Kommunikation innerhalb der Oracle NCA zur Verfügung.

---

[20] Oracle hat den Standard mit eingereicht.

[21] für die indirekte Aktualisierung von Views: `CREATE TRIGGER emp_insert INSTEAD OF INSERT ON emp_view BEGIN...` .

[22] eine Abfrage innerhalb des `FROM`-Teils einer anderen Abfrage.

[23] Dazu bietet sich zum Beispiel das URL-Parameterformat an. Für die Kodierung und Dekodierung muß man jedoch eigene Mittel schaffen-

[24] Dies kann über die Prozedur `request_pieces` recht umständlich umgangen werden.

[25] Allerding kann man dieses Problem durch Java-eigene Mittel in Form der `java.net.URL` Klasse mit ihrer Methode `getContent()` relativ einfach umgehen. Damit läßt sich quasi ein ICX-Emulator für Java schreiben, der über HTTP andere Cartridges aufrufen kann.

Kollaboration

Weder der Anwendungsserver noch der Datenbankserver stellen Mechanismen für die Verwendung von SMTP/POP3 oder einem der anderen Kollaborationsprotokolle zur Verfügung. Entsprechende Funktionalitäten sind z.B. über Java- oder Perl-Anwendungen selbst zu realisieren.

Verzeichnis

LDAP Der Zugriff auf Verzeichnisse wird über LDAPv3 unterstützt. Dabei fungiert der Anwendungsserver als LDAP Client. Der Zugriff ist zur Zeit nur für den Netscape Directory Server zertifiziert, sollte aber für alle LDAP-fähigen Verzeichnis-Server funktionieren.

Oracle Names ist ein Produkt [Ora96a], das einen Verzeichnisdienst für Datenbanken und Datenbankverbindungen innerhalb der NCA bietet, der von dem proprietären Protokoll TNS genutzt.

Sicherheit

X.509 Die Identifizierung von Benutzern oder Anwendungen kann über unterschiedliche Wege erfolgen. Als Authentisierungsmechanismen werden der HTTP Benutzername und das zugehörige Kennwort, der Domainname, die IP-Adresse, X.509v3-Zertifikate in Verbindung mit LDAP-Verzeichnissen und Zugiffskontrollisten (ACLs) oder ein Datenbank-Benutzername mit zugehörigem Kennwort unterstützt.

SSL Die Verschlüsselung von HTTP- oder IIOP-Verbindungen kann über SSLv3 erfolgen. Die gegenseitige Authentisierung der Kommunikationspartner erfolgt dann über X.509v3-Zertifikate. Die Konfiguration der Nutzung von SSL über HTTP erfolgt über die Angabe eines entsprechenden TCP/IP-Ports, die Verschlüsselung von IIOP mittels SSL wird über die ORB Konfiguration definiert. Es stehen die Varianten unverschlüsselt, IIOP und IP-basierte ACLs sowie IIOP/SSL und Zertifikat-basierte ACLs zur Verfügung.

Advanced Networking Option Die Verschlüsselung des TNS im Rahmen von SQL*Net- oder Net8-Verbindung zum Datenbankserver kann über die *Advanced Networking Option* (ANO) [Ora97b] erfolgen.

6.3.3 Computing-Dienste

Integration

CGI Durch die Verwendung der Perl-, C- und JWeb-Cartridges steht CGI-Funktionalität für die jeweiligen Progammiersprachen zur Verfügung. Allerdings

muß im Gegensatz zur klassischen CGI-Nutzung nicht jedesmal ein neuer externer Prozeß gestartet werden. Die Anwendungen werden im Anwendungsserver unter Nutzung aller Lastverteilungs- und damit Performancevorteile ausgeführt.

Message Queuing Das MQI kann über den Datenbankserver in Form des *Procedural Gateway for MQSeries* genutzt werden. Es stehen eine Reihe von PL/SQL-Prozeduren zur Verfügung, mit denen die gewünschte Funktionalität erreicht werden kann.

CORBA Oracle verwendet seinen eigenen, IIOP-fähigen Object Request Broker namens OMX, der allerdings einige Einschränkungen aufweist. So steht z.B. nur für die Programmiersprache C das *Dynamic Invocation Interface* zu Verfügung. Ein direkter Zugriff auf die ORB-Funktionalität ist aufgrund fehlender Bindings nicht möglich, sondern nur über die bereitgestellte JCorba-Umgebung. Daher lassen sich Corba-Anwendungen anderer ORBs[26] nicht ohne Anpassungen verwenden. Die Benutzung der JCorba-Objekte ist nur über IIOP und gegebenenfalls unter Verwendung eines anderen ORBs möglich.

Transparent Gateways Über die *Transparent Gateways* lassen sich diverse andere Datenquellen an den Datenbankserver anschließen. Sie stehen dann wie normale Ressourcen des Datenbankservers zur Verfügung. Neben der Unterstützung der wichtigsten anderen relationalen Datenbanken besteht auch die Möglichkeit zur Verwendung beliebiger ODBC-Datenquellen.

Transaktion

Der Anwendungsserver unterstützt deklarative und programmatische Transaktionsklammern. Es wird das X/Open DTP-Modell mit TX/XA und der CORBA OTS unterstützt. Wie bereits erwähnt, ist HTTP ein zustandsloses Protokoll. Für HTTP-basierte Transaktionen wird der Sitzungskontext mittels einer Sitzungs-ID und einer Transaktions-ID über *Cookies* verwaltet. Der Datenbankserver besitzt eine DTP-XA-Schnittstelle und kann als Koordinator verteilter Transaktionen nach dem DTP-Modell eingesetzt werden.

Programmierung

Java wird serverseitig über das JWeb-Cartridge zur Verfügung gestellt. Dieses verfügt über eine JVM[27] sowie Klassenbibliotheken für die Generierung von HTML und die Verwendung der WRB-Dienste. Allerdings ist festzuhalten, daß sich die Schnittstellen von denen der Servlet Spezifikation unterscheiden. Oracle nutzt hier eigene APIs - eine Unterstützung der Standard-APIs ist jedoch in Vorbereitung. Für die Verwaltung von Transaktionen kann über den JTS der OTS des ORBs des Anwendungsservers genutzt werden. Außerdem steht JDBC für die Verwendung des Datenbankservers zur Verfügung. Die Verwendung bestehender PL/SQL-Anwendungen innerhalb einer Java-Anwendung wird durch

[26] wie etwa Visigenics VisiBroker.
[27] Java 1.1.4

die Bereitstellung eines *pl2java*-Werzeuges zur Erzeugung von Wrapper-Klassen für PL/SQL-Packages erleichtert.

Ein zweites Cartridge - das JCorba-Cartridge - bietet die Möglichkeit proprietäre Java-CORBA-Komponenten (JCO) zu erstellen. Der Zugriff auf diese Komponenten erfolgt immer über IIOP und nie über HTTP. Damit kann kein direkter Aufruf einer JCO über eine URL und damit auch nicht über die ICX erfolgen. Die Nutzung eines JCO ist nur über sein Remote Interface möglich und ist damit Java-Applets oder dem JWeb-Cartridge vorbehalten.

Eine Verwendung von Java im Datenbankserver ist derzeit noch nicht möglich [28].

JavaSkript Serverseitiges JavaSkript wird nicht unterstützt. Auf der Clientseite kann natürlich JavaScript eingesetzt werden.

C Die Programmiersprache C wird über das C-Cartridge unterstützt. Sie ist auch die einzige Programmiersprache, in der zur Zeit eigene Cartridges geschrieben werden können. C++ wird vom Anwendungsserver selbst nicht unterstützt, doch lassen über die normale CGI-Schnittststelle natürlich C++ Anwendungen integrieren.

PL/SQL ist die Datenbankprogrammiersprache von Oracle. Über ein PL/SQL-Cartridge kann diese Programmiersprache für die Entwicklung von Anwendungen genutzt werden. Es stehen Packages[29] zur Verfügung, die die Generierung von HTML, die Nutzung von Cookies, die Verwendung von CGI-Umgebungsvariablen[30] sowie die Kommunikation über die ICX unterstützen.

Perl Als Skriptsprache wird serverseitig zur Zeit nur Perl unterstützt. Dabei steht die Perl Version 5.003 mit der Oracle DBI/DBD-Schnittstelle [Bun98b, Bun98a] sowie eine OraPerl Emulation über das Perl Cartridge zur Verfügung [31].

Server Side Includes Das LiveHTML-Cartridge realisiert das Konzept der *Server Side Includes* (SSI). Über dieses Cartridge können auch Perl-Skripte in die HTML-Seiten eingebettet werden. Dies ist ein Ersatz für das Fehlen des serverseitigen JavaScripts.

Komponenten

JavaBeans können im Rahmen von JWeb und JCorba eingesetzt werden.

[28] jedoch mit der in Kürze erscheinenden Version 8i.
[29] Packages entsprechen Bibliotheken.
[30] z.B. über `owa_util.get_cgi_env('PATH_TRANSLATED')`.
[31] Außerdem werden wichtige weitere Perl Module wie CGI, libwww u.a. bereitgestellt.

EJB Der Anwendungsserver kann als EJB-Server fungieren und EJB Container zur Verfügung stellen. Die aktuelle Version des Anwendungsservers unterstützt keine Entity Beans, sondern nur transiente *Session Beans*. Außerdem wird nicht das *EJB Home Interface* genutzt, sondern eine eigene Verwaltung des Lebenszyklus realisiert[32].

CORBA Komponenten werden durch das JCorba-Cartridge realisiert. Allerdings hat dieses Komponentenmodell gewisse Nachteile. Die Komponenten sind nicht über das *Dynamic Invocation Interface* (DII), sondern nur über statische Stubs nutzbar. Existierende Corba-Komponenten müssen für die Verwendung mit dem JCorba-Cartridge angepaßt werden.

Cartridges sind die proprietären Komponenten der NCA. Das Cartridge-Konzept steht sowohl auf dem Anwendungsserver (*Application-Cartridges*) als auch für den Datenbankserver (*Data-Cartridges*) zur Verfügung. Cartridges können zur Zeit nur in der Programmiersprache C entwickelt werden.

6.3.4 Meta-Dienste

Management

SNMP Der Anwendungsserver unterstützt SNMP für die Identifizierung und Überwachung durch SNMP-basierte Managementanwendungen. Es werden Informationen zur Site, dem WRB, den installierten Anwendungen und Cartridges sowie Statistikinformationen in einer entsprechenden MIB bereitgestellt[33].

Auch für den Datenbankserver steht eine SNMP-Unterstützung zur Verfügung [Ora98d]. Hier werden die Informationen aus den Systemzustandstabellen[34] bereitgestellt.

Oracle Enterprise Manager Zusätzlich steht der Oracle Enterprise Manager [Ora97d] zur Verfügung, welcher für die umfassende Verwaltung und Steuerung aller Komponenten der NCA entwickelt worden ist. Er basiert allerdings auf einem proprietären Managementprotokoll mit einem eigenen Agentensystem.

6.4 Zusammenfassung

Nach der Diskussion der Leistungen der Oracle NCA in den einzelnen Bereiche des *Network Computing*-Referenzmodells fasse ich die Ergebnisse nun in der Tabelle 6.4 zusammen.

[32] z.B. mittels eines `setSessionContext()` anstelle des `ejbCreate()`.
[33] Leider steht dieser Dienst nur für die UNIX-Versionen des Anwendungsservers zur Verfügung - unter Windows NT muß darauf verzichtet werden.
[34] z.B. `V$SYSSTAT`.

Dienst	NC-Standard	Oracle-Konzept
Präsentation	HTML/MIME	Bibilotheken
	GIF/JPEG	LOBs
	PDF	LOBs
	XML	PLSXML
Persistenz	SQL	SQL
	SQL/CLI	OCI
	ODBC	ODBC
	JDBC	JDBC
	SQLJ	SQLJ
Kollaboration	SMTP/POP3	-
	NNTP	-
	IRC	-
	FTP	-
Transport	TCP/IP	TCP/IP
	HTTP	Listener
	IIOP	ORB
	-	TNS (SQL*Net, Net8)
Verzeichnis	LDAP	Authentication Server als LDAP Client
	DNS/URL	über TCP/IP
	-	Oracle Names
Sicherheit	X.509	Authentication Server
	SSL	Listener
Integration	CGI	WRB, Listener
	MQI	Procedural Gateway for MQSeries
	CORBA	ORB
	-	Transparent Gateways
Transaktion	TX/XA	WRB
	OTS/JTS	ORB
Programmierung	Java	JWeb, JCorba
	JavaSkript	-
	SSI	LiveHTML
	-	Perl
	-	C
Komponenten	JavaBeans	JWeb, JCorba
	EJB	EJB Server
	CORBA	JCorba
	-	Cartridges
Management	SNMP	Standard MIB
		Enterprise Manager

Tab. 6.4: Oracle Technologien im Rahmen des *Network Computing*

In den klassischen Datenbankbereichen Persistenz und Transaktionen bietet die NCA erwartungsgemäß alle erforderlichen Dienste. Erstaunlich sind allerdings

die ausgeprägten Fähigkeiten zur Integration anderer Systeme, die sogar die transparente Einbindung fremder Datenquellen erlauben.

Im Bereich der Komponentendienste offenbart die NCA hingegen deutliche Schwächen. Die aktuelle EJB-Implementation ist zur Zeit nicht in einem Zustand, der ihre Verwendung sinnvoll erscheinen läßt. Aktuell ist der ORB als IIOP-Kommunikationsmittel verwendbar - jedoch nicht als CORBA Integrationsplattform. In erster Linie liegt das an dem proprietären Komponentenmodell der Cartridges, das bei den unterschiedlichen Problemen immer wieder durchschimmert. Jedoch muß dabei berücksichtigt werden, daß dieses Komponentenmodell lange vor der Standardisierung der *Network Computing* Komponentenmodelle entstanden ist.

Positiv anzumerken ist, daß die NCA nach außen über HTTP und IIOP die benötigten Transportprotokolle bietet und außerdem in der Lage ist, die geforderten Sicherheitskonzepte inklusive der Verwaltung von Zertifikaten zu realisieren.

Im Bereich der Programmierdienste bietet die NCA abgesehen von JavaSkript alle geforderten Programmiersprachen; ersatzweise wird Perl zur Verfügung gestellt. Diese Skriptsprache hat historisch bedingt im Bereich der serverseitigen Programmierung von Internetanwendungen eine starke Verbreitung. Die verwirrende Zweiteilung in CORBA-basiertes Java und Web-basiertes Java ist beruht auf den unterschiedlichen Kommunikationsmöglichkeiten[35], die beide Ansätze bieten.

Für das Management bietet die NCA Informationen über ihre Komponenten in Form von speziellen MIBs. Diese stehen jedem SNMP-fähigen Managementwerkzeug zur Verfügung.

Zusammenfassend läßt sich festhalten, daß die NCA als Datenbankmanagementsystem einen Großteil der erforderlichen Dienste für das *Network Computing* erbringen und sowohl Aufgaben aus der Anwendungsebene als auch der Domänenebene erfüllen kann. Sie ist damit als zentrale Realisierungsplattform für Anwendungen der digitalen Wirtschaft nutzbar.

Auch ist deutlich erkennbar, daß der ehemals monolithische Ansatz der Datenbankmanagemensysteme mittlerweile von einer Orientierung hin zu offenen, verteilten Systemen abgelöst worden ist. Eine Richtung, in die sich Datenbanksysteme bewegen müssen, um künftig wieder eine zentrale Rolle einnehmen zu können.

[35] IIOP oder HTTP

Kapitel 7

ERGEBNISSE

Die Persistenz in offenen, verteilten Anwendungssystemen ist das Umfeld, in dem sich diese Arbeit bewegt. Konkret habe ich den Bereich des *Network Computing* als Idealform der offenen, verteilten Verarbeitung untersucht. Für den Bereich der Persistenz galt es dabei zu ermitteln, in welcher Form die klassischen Erbringer von Persistenzleistungen - die Datenbanksysteme - in das Verarbeitungsparadigma des *Network Computing* integriert werden können.

7.1 Zusammenfassung

Als ein Ergebnis dieser Arbeit habe ich das Konzept des *Network Computing* als technologische Basis von Anwendungen der digitalen Wirtschaft entwickelt. In einer solchen, von vornherein auf das Zusammenwirken von Hardware-, Software- und Kommunikationsbausteinen unterschiedlicher Hersteller ausgerichteten Umgebung müssen Interoperabilität und Portabilität besonders groß geschrieben werden. Aus diesem Grund wurden in dieser Arbeit wichtige Standards dargestellt, in den Zusammenhang des *Network Computing* eingeordnet und bewertet. Besonderes Anliegen dieser Arbeit ist es gewesen, das Übermaß an Teilinformationen zu kondensieren und in mein Gesamtkonzept des *Network Computing* zu überführen. Als Ergebnis findet sich das in Abbildung 7.1 dargestellte *Network Computing*-Referenzmodell, in dem die aktuell relevanten Standards zusammenfassend eingeordnet werden.

Um die Bedeutung von Datenbanksystemen im Rahmen des *Network Computing* zu messen, habe ich zunächst das klassische Datenbankprofil mit dem Infrastrukturprofil der digitalen Wirtschaft verglichen. Dieser Vergleich ergab zunächst eine Zwei-Drittel-Übereinstimmung von bereitgestellten und geforderten Diensten.

Im weiteren Verlauf der Untersuchung habe ich dann ein konkretes Datenbankmanagementsystem vorgestellt und analysiert. Bei der Einordung der Fähigkeiten dieses Systems in das *Network Computing*-Referenzmodell zeigt sich ein deutlich höherer Erfüllungsgrad, als es der abstrakte Profilvergleich vermuten ließ. Dies resultiert in erster Linie aus der Unterstützung der Erstellung von Dokumenten der Präsentationsdienste sowie aus den ausgeprägten Integrationsfähigkeiten des untersuchten Systems.

Allerdings haben sich in bezug auf die Dienstausprägungen des *Network Computing*-Referenzmodells noch deutliche Schwächen gezeigt, die in der ursprünglich

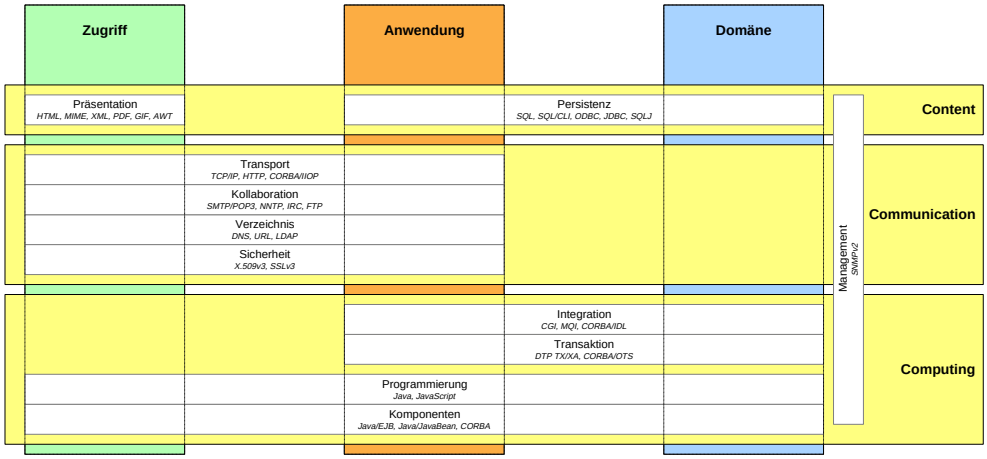

Fig. 7.1: *Network Computing*-Referenzmodell

eher Blackbox-artigen Systemphilosophie dieser Art von Systemen gründen. Die Öffnung in Richtung allgemeiner Standards findet zwar bereits statt, hat jedoch noch nicht den benötigten Grad erreicht.

Trotzdem läßt sich als Ergebnis festhalten, daß Datenbankmanagementsysteme - wie in Abbildung 7.2 ersichtlich - in der heute üblichen Ausprägung mit all ihren Bestandteilen sehr wohl als zentraler Diensterbringer des *Network Computing* fungieren können.

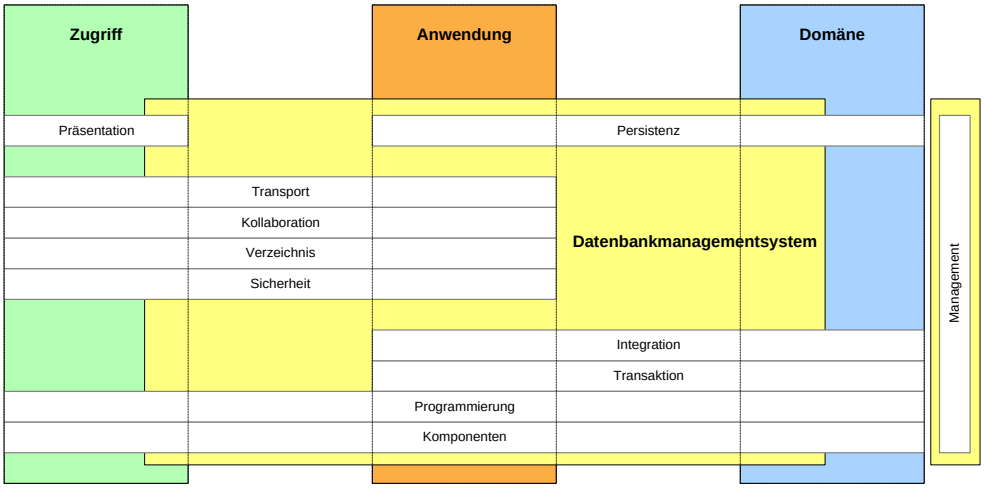

Fig. 7.2: Ein DBMS als zentraler Diensterbringer im *Network Computing*-Referenzmodell

7.2 Schlußfolgerungen

Im Rahmen der Einführung des Client/Server-Konzeptes und der damit verbundenen Dezentralisierung haben sich Datenbanksysteme zu *Commodities*[1] entwickelt. Eine Datenbank ist selbstverständlich - aber mehr auch nicht. Sie hat denselben Stellenwert wie ein Betriebssystem oder ein Netzwerk.

Meine Vision ist es, daß Datenbanksysteme wieder ihre ursprüngliche Bedeutung als zentrale Grundlage von Informationssystemen zurückgewinnen. Die einsetzende *Rezentralisierung* von Anwendungen und die damit einhergehende Konzentration von klassischen Diensten einer Datenbank bietet eindeutig diese Möglichkeit.

Eine Forderung an diese Systeme ist jedoch ihre Öffnung, etwa in Richtung der von mir ausgewählten *Network Computing*-Standards. Einige Hersteller - wie zum Beispiel Oracle - haben dieses inzwischen auch erkannt und entwickeln zur Zeit entsprechende Produkte[2].

Die neue alte Rolle von Datenbanksystemen ist in Abbildung 7.3 dargestellt. Im Rahmen des *Network Computing* erweitern Datenbankmanagementsysteme dabei ihre Fähigkeiten von der sicheren, zuverlässigen Verwaltung und Bereitstellung von Daten hin zu einer zentralen Instanz für die Nutzung von Anwendungen - sie werden zu Anwendungsmanagementsystemen.

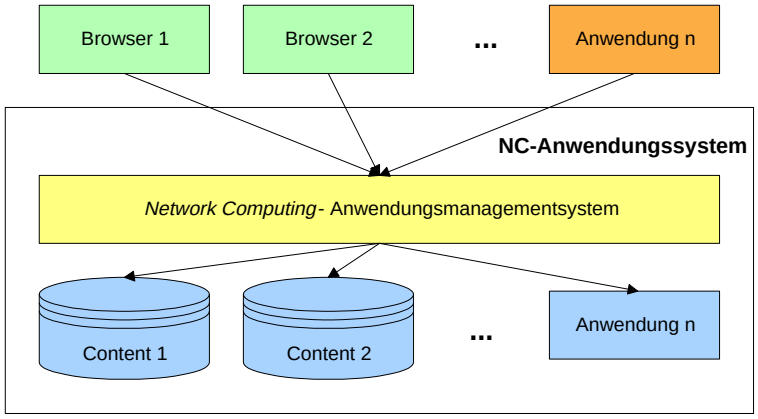

Fig. 7.3: Entwicklung von Datenbanksystemen zu *Network Computing*-Anwendungssystemen

Tendenzen dieser Entwicklung hat es immer wieder gegeben. Sie reichen von der Erweiterung der Datenbankprogrammiersprachen bis hin zu integrierten Datenbankentwicklungssystemen.

Doch durch die Entstehung der digitalen Wirtschaft mit dem ihr zugrundeliegenden Paradigma des *Network Computing* und seinem entsprechenden gemeinsamen Verarbeitungsmodell bietet sich jetzt mehr als *nur* eine weitere Chance

[1] Rohstoffe
[2] Oracle8i

für diesen Schritt - die Komplexität der entstehenden Anwendungen erfordert schlicht die Entwicklung von Anwendungsmanagementsystemen.

Kapitel 8

AUSBLICK

Diese Arbeit soll als Hilfestellung für die Beurteilung der aktuellen Veränderungen im Bereich der Konzepte für betriebliche Informationssysteme dienen, um so bestimmte Technologien und Konzepte in den Gesamtzusammenhang des *Network Computing* einordnen zu können. Darüber hinaus soll mit Hilfe des erarbeiteten Referenzmodells auch das Verständnis und der Gedankenaustausch von bzw. über das *Network Computing* gefödert werden.

Die in dieser Arbeit ausgearbeitete Vision der digitalen Wirtschaft als Manifestation des Übergangs zur Informationsgesellschaft basiert auf der Technik des Internet. Larry Tesler[1] beschreibt das Internet folgendermaßen: Das Internet behandelt Allgegenwärtigkeit und Interoperabilität. Es handelt von jedem, überall und der Möglichkeit über Standards zu kommunizieren [Tes97]. Diese Betonung der Standards als Basis für die Vision des universellen Zugriffs ist eine Grundlage meiner Arbeit. Es hat sich gezeigt, daß bereits ausreichende Standards für die unterschiedlichen Aspekte des *Network Computings* existieren. Jedoch hat die Analyse auch ergeben, daß die Implementierung der Standards in Form von verfügbaren, kommerziellen Produkten teilweise noch auf sich warten läßt[2].

Wie bei allen anderen Entwicklungen im EDV-Bereich begegnen wir auch hier dem Paradoxon, daß neue Technologien neue Geschäftsmöglichkeiten, Rationalisierungspotentiale und Effizienzsteigerungsmöglichkeiten eröffnen, gleichzeitg aber ein meist nur unvollständiges Verständnis der neuen Technologien, Komponenten, Plattformen und Entwicklungsumgebungen diese neuen Chancen behindert und einschränkt. Entsprechende Kompetenzen für netzwerk-zentrierte EDV müssen also erst noch aufgebaut werden. Dennoch bin ich der Meinung, daß eine ausreichende technische Unterstützung für den Anfang gegeben ist. Die Vision der verteilten Verarbeitung als Manifestation der dritten Generation von Client/Server-Architekturen mit ihrem Konzept der interagierenden Komponenten geht konform mit den Ansichten von Peter Wegner über den Wechsel von Algorithmen zu Interaktionen [Weg98]. Die Schaffung eines einheitlichen Verarbeitungsmodells auf der Basis einer umfassenden Infrastruktur für die Entwicklung, den Betrieb und die Verwaltung von Informationssystemen im Rahmen des *Network Computing* schreitet immer weiter voran - auch angesichts der vielen unterschiedlichen Ansätze von Standardisierungsgremien und Herstellern.

[1] Apple Computer's Chief Scientist

[2] Besonders die Abwesenheit von ausreichenden Implementierungen einer *Enterprise JavaBeans*-Umgebung habe ich schmerzlich vermißt.

Für mich hat der Übergang zur Informationsgesellschaft bereits begonnen. Einen Großteil der Informationen für diese Arbeit konnte ich über das Internet recherchieren. Selbst wenn die Information nicht online verfügbar war, konnte ich zumindest entsprechende Bezugsquellen erfahren und zum Teil sogar nutzen. Schon heute können wir Online-Bankgeschäfte tätigen und ohne großen Aufwand Kontakt zu Freunden und Familienmitgliedern in anderen Ländern der Welt halten. So konnte diese Arbeit von einem Freund in Barcelona, wie von einem Cousin in Melbourne begutachtet werden: Die räumliche Trennung wird durch das digitale Medium aufgehoben.

Dabei wird die Technik des Internet oft mit Gutenbergs Erfindung des Buchdrucks verglichen. Ich stimme aber Steve McGeady in [McG97] zu, der erst der Nutzung des Internet in Form des *World Wide Web* eine solch revolutionäre Bedeutung beimißt und dies mit der Reformation Martin Luthers vergleicht. Ähnlich wie bei der Buchdruckkunst hat es beim Internet eine Zeit gebraucht, bis eine Anwendungsform gefunden wurde, die wirklich gravierende gesellschaftliche Veränderungen bewirken kann. Gutenbergs Technik war zunächst nur für eine kleine Anzahl von Menschen relevant. Erst die Übersetzung der Bibel in eine, breiten Bevölkerungskreisen verständliche Sprache ermöglichte einer großen Anzahl von Menschen, eigene Interpretationen ihrer Weltordnung zu finden. Zwischen der Erfindung der Druckerpresse und der Reformation vergingen ca. 50 Jahre - zwischen der Erfindung des Internets und seiner Nutzung in Form des *World Wide Web* knappe 25 Jahre. Ein weitere Vergleich zieht parallelen zwischen der Nutzung des Fernsehens und der des Internet. Die aktuelle Anzahl von, an das Internet angeschlossenen Haushalten entspricht ungefähr der Anzahl von Schwarz-Weiß-Fernsehern im Jahre 1950. Es wird mit einer ähnlichen Entwicklung der Verbreitung von Internet-Technologien gerechnet, wie es das Fernsehen vorgemacht hat.

Eine Euphorie, die indes einen nachhaltigen Dämpfer erfährt. Diana Lady Dougan[3] formuliert dieses ebenso einfach wie plakativ: Mehr als die Hälfte der Weltbevölkerung muß noch ihren ersten Telefonanruf machen. Eine Tastatur zu berühren ist nicht einmal eine Vision [Dou97]. In der Tat betreffen die von mir beschriebenen wirtschaftlichen und gesellschaftlichen Veränderungen nur einen relativ kleinen Teil der Menschheit: die derzeitigen Industrienationen. Der Übergang in das Informationszeitalter setzt das Erfahren der Industriegesellschaft voraus. Einige sogenannte Entwicklungsländer versuchen diesen Schritt zu beschleunigen: So gibt es in Ägypten[4] ein Programm *Egypt & The 21st Century*[5], in dem explizit der Übergang in eine Informationsgesellschaft gefordert wird. Dieser Übergang stelle eine lebenswichtige politische Orientierung dar, die sich von den Schulen, über die wissenschaftlichen Einrichtungen bis hin zu den Produktionsstätten erstrecken müsse[Egy97].

Nun stellt sich die Frage nach den Voraussetzungen für den Übergang. Wie bereits aufgezeigt, ist ein Wechsel zu einer netzwerk-zentrierten Perspektive auf die Erstellung von Informationssystemen nötig. Die hierfür notwendige Kompetenz muß allerdings noch aufgebaut werden. Die Konvergenz von EDV, Kommunika-

[3] Global Information Infrastructure Commission

[4] Yahoo! listet 339 WWW-Sites in Äqypten. Im Vergleich dazu sind es in Österreich 880. Für Deutschland gibt es 4926 Server und in Indien sogar 5102.

[5] nachzulesen auf dem WWW-Server des Staatlichen Informationsdienstes `http://www.sis.gov.eg/egyptinf/html/e21cent/html/ch00.htm`.

tion und Medien findet zwar bereits in ersten Ansätzen statt, jedoch hat sich eine wirklich umfassende, integrierte Sichtweise der Technologien noch nicht durchgesetzt. Die umgesetzten Konzepte basieren meist im Kern nur auf einem der drei Bereiche und setzen die Techniken und Erfahrungen der anderen Bereiche nur unterstützend ein. Es wird sicher noch einige Zeit dauern, bis die integrierte Perspektive auch die Köpfe der Praktiker und Produzenten *durchdrungen* hat.

Das World Wide Web als eine Grundform des *Network Computing* ist ein gigantisches Informationssystem. In [SSU95] wird beschrieben, daß Webmaster erst langsam begreifen, daß sie eigentlich Datenbankadministratoren sind - allerdings meist ohne ein unterstützendes Datenbankmanagementsystem. Die Frage nach der Rolle der Datenbanksysteme im Rahmen dieser neuen Paradigmen steht im Zusammenhang mit der Frage, wo die Grenze zwischen Datenbanksystem und Middleware zu ziehen ist[6]. Betrachtet man das Profil aus Kapitel 2 und vergleicht dieses mit den bereitgestellten Diensten eines Anwendungsservers im Zusammenschluß mit einem Datenbankserver (entsprechend Kapitel 6), so kann man erkennen, daß noch immer alle Dienste zur Verfügung stehen. Allerdings können diese Dienste nun auf verschiedene Maschinen verteilt werden. Außerdem haben sich die Dienstschnittstellen von herstellerspezifischen zu offenen Standardschnittstellen entwickelt. Geht man also davon aus, daß mein Datenbank-Profil ein Datenbanksystem ausreichend charakterisiert, dann entspricht Oracle mit allen Elementen seiner NCA noch immer einem Datenbanksystem - allerdings mit einer deutlichen Ausrichtung hin zum Trend offener, verteilter Systeme. Dies ist ein Hinweis darauf, daß Datenbanksysteme ihrer traditionellen Rolle als Grundlage betrieblicher Informationssysteme auch im Rahmen des *Network Computing* gerecht werden können und damit auch der digitalen Wirtschaft.

In einer weiterführenden Arbeit könnte man das gefundene Referenzmodell nun weiter strukturieren und konkretisieren. Möglich wäre zum Beispiel, verschiedene Perspektiven[7] explizit mit in das Referenzmodell aufzunehmen. Außerdem wären formale Verfahren zur Durchführung der Bewertung eines konkreten Architekturmodells mit der Referenzarchitektur zu erarbeiten.

Im Bereich der Datenbanksysteme könnten die Anforderungen an Datenbanken im Rahmen des *Network Computings* weiter konkretisiert und in einer Art *Network Computing Database Manifesto* spezifiziert werden. Die aktuelle Diskussion über die Zukunft von Datenbanksystemen [SSU95, Cat91, SSU91, SZ97, SRL+90] ist meist viel weiter gefaßt und behandelt Themen aus den unterschiedlichsten Bereichen. Eine eingeschränktere Betrachtung mit der Perspektive auf den Bereich des *Network Computing* wäre meiner Ansicht nach durchaus legitim.

Welche sozialen und wirtschaftlichen Veränderungen aber wird dieser Übergang zur Informationsgesellschaft nach sich ziehen ? Der amerikanische Schriftsteller T.S. Elliot hat hierzu sehr treffende Worte gefunden - lange vor der Entstehung des Internet: *Where is the wisdom that we've lost in knowledge ? Where is the knowledge that we've lost in information ?*. Auch wenn ich mich in dieser Ar-

[6] Bei einem Netscape Firmenvortrag habe ich folgende Aussage gehört: *The Webserver ist just part of the database if you're Oracle, and just part of the operating system, if you're Microsoft.*

[7] Ähnlich dem RM-ODP [ISO95b] oder [AMJ+98].

beit auf eine sehr technophile Beschreibung der Vorgänge und Veränderungen beschränkt habe, läßt sich über diesen Aspekt hinaus eines auf jeden Fall feststellen: Die Auswirkungen dieser Paradigmenwechsel und Konvergenzen bzw. Konzentrationen werden mit Sicherheit nicht nur positive Effekte haben. Die Ergebnisse der *Harvard Conference on the Internet an Society* [Har97] geben einen ersten Eindruck von den umfassenden Veränderungen vor denen wir stehen. Oder, um es mit den Worten einer allgemeinen Weisheit unter den *Netzbewohnern* zu sagen: *In the NET you always travel but never arrive ...*

Teil III

ANHANG

Anhang A

ANDERE ARCHITEKTURMODELLE

The Open Group Architectural Framework (TOGAF)	The Open Group [OG97]
Distributed Computing Environment (DCE)	The Open Group [OSF91, OSF95]
Open System Interconnection (OSI)	ISO [Ker89]
Reference Model: Open Distributed Processing (RM-OPD)	ISO [ISO95b, PSW96]
Object Management Architecture (OMA)	OMG [OMG97c]
Open Blueprint	IBM [IBM97]

Tab. A.1: Andere Architekturmodelle

Anhang B

ORGANISATIONEN

ANSI	American National Standards Institute `http://www.ansi.org.org`
DTMF	Desktop Management Task Force `http://www.dtmf.org`
ECMA	European Computer Manufacturers Association `http://www.ecma.ch`
IEC	International Engineering Consortium `http://www.iec.org`
IETF	Internet Engineering Task Force `http://www.ietf.org`
ISO	International Organisation for Standardization `http://www.iso.ch`
ISOC	Internet Society `http://www.isoc.org`
ITU	International Telecommunication Union `http://www.itu.int`
OMG	Object Management Group `http://www.omg.org`
W3C	World Wide Web Consortium `http://www.w3.org`
X/Open	The Open Group `http://www.opengroup.org`

Tab. B.1: Organisationen

Anhang C

STANDARDS

Präsentation		
HTML 2.0	IETF	RFC 1866
HTML 3.2	W3C	REC-html32
HTML 4.0	W3C	REC-html40
SGML	ISO	8879
SDIF	ISO	9069
XML	W3C	REC-xml
MIME	IETF	RFC 2045, 2046, 2047, 2048, 2049
GIF	Compuserve	GIF89a
JPEG	ISO/IEC	10918
VRML	ISO/IEC	14772
AWT	SUN	
PDF	Adobe	

Persistenz		
SQL	ISO/IEC	9075
SQL/CLI	X/Open	C451
SQLJ	ANSI	X3.135.10-1998
ODBC	Microsoft	
JDBC	SUN	ISBN 0-201-30995-5

Transport		
IP	IETF	RFC 791
TCP	IETF	RFC 793
UDP	IETF	RFC 768
HTTP 1.0	IETF	RFC 1945
HTTP 1.1	IETF	RFC 2068
SHTTP	IETF	draft-ietf-wts-shttp-03
HTTPS	Netscape	
IIOP	OMG	CORBA 2.0
DCE RPC	X/Open	C309
RPC	IETF	RFC 1050

Kollaboration		
SMTP	IETF	RFC 821,822
POP3	IETF	RFC 1939
IMAP4	IETF	RCF 2060, 2061, 2062
IRC	IETF	RFC 1459
NNTP	IETF	RFC 977
FTP	IETF	RFC 959
X.400	ISO/IEC	10021
Verzeichnisse und Lokalisierung		
LDAP	IETF	RFC 1777, 1778
DNS	IETF	RFC 1034, 1035
URL	IETF	RFC 1738
URI	IETF	RFC 2396
X.500	ISO/IEC	9594
Sicherheit		
X.509	ISO/IEC	9594-8
X.509v3	IETF	draft-ietf-pkix-ipki-part1-11
SSLv3	IETF	draft-freier-ssl-version3-02
IPSec	IETF	RFC 1825-1829, draft-ietf-ipsec-pki-req-01
S/MIME	IETF	draft-ietf-smime-msg-06
Integration		
MQSeries (MQI)	IBM	ISBN 0-07-005730-3
CGI	IETF	draft-coar-cgi-v11-02
Transaktionen		
Distributed TP	X/Open	G504
XA	X/Open	C193
XA+	X/Open	S423
TX	X/Open	C504
OTS	OMG	CORBAservices
Programmierung		
Java	Sun	ISBN 0-201-63451-1
ECMAScript	ECMA	262
Perl		ISBN 1-56592-149-6
Komponenten		

CORBA	OMG	CORBA 2.0
JavaBeans	Sun	Specification 1.01
Enterprise JavaBeans	Sun	Specification 1.0
Management		
SNMP	IETF	RFC 1157
SNMPv2	IETF	RFC 1907
SNMPv3	IETF	draft-ietf-snmpv3-intro-00
MIB-II	IETF	RFC 1215

Tab. C.1: Standards

Anhang D

ABKÜRZUNGEN UND AKRONYME

ACID: *atomicity / consistency / isolation / durability*

ACL: *Access Control List*

ANSI: *American National Standardization Institute*

AWT: *Abstract Windowing Toolkit*

BLOB: *Binary Large Object*

BPR: *Business Process Reengineering*

CA: *Certificate Authority*

CDR: *Common Data Representation*

CGI: *Common Gateway Interface*

CLI: *Call Level Interface*

CORBA: *Common Object Request Broker Architecture*

CRL: *Certificate Revocation List*

CTI: *Computer Telephony Integration*

DB: Datenbank

DBA: Datenbank-Administrator

DBS: Datenbanksystem

DCE: *Distributed Computing Environment*

DDL: *Data Definition Language*

DII: *Dynamic Invocation Interface*

DLL: *Dynamic Link Library*

DML: *Data Manipulation Language*

DN: *Distinguished Name*

DNS: *Domain Name System*

DSI: *Dynamic Skeleton Interface*

DTD: *Document Type Definition*

DTS: *Distributed Time Service*

ECMA: *European Computer Manufacturers Association*

EDI: *Electronic Data Interchange*

EDV: Elektronische Datenverarbeitung

EJB: *Enterprise Java Bean*

FIPS: *Federal Information Processing Standard*

GIOP: *General Inter-ORB Protocol*

GIF: *Graphics Interchange Format*

HTML: *Hypertext Markup Language*

HTTP: *Hypertext Transfer Protocol*

ICX: *Inter-Cartridge Exchange*

IDE: *Integrated Development Environment*

IDL: *Interface Definition Language*

IETF: *Internet Engineering Task Force*

IIOP: *Internet Inter-ORB Protocol*

IMAP: *Internet Message Access Protocol*

IP: *Internet Protocol*

ISO: *International Standards Organization*

IT: Informationstechnologie

ITU: *International Telecommunication Union*

JDBC: *Java Datebase Connectivity*

JMS: *Java Message Service*

JMAPI: *Java Management API*

JNDI: *Java Naming and Directory Interface*

JTA: *Java Transaction API*

JTS: *Java Transaction Service*

JVM: *Java Virtual Machine*

LDAP: *Light-weight Directory Access Protocol*

MIB: *Management Information Base*

MIME: *Multipurpose Internet Mail Extensions*

MOM: *Message Oriented Middleware*

MQ: *Message Queuing*

MQI: *Message Queue Interface*

NC: *Network Computer*

NCSA: *U.S. National Centre for Supercomputing Applications*

ODBMS: *Object Databasemanagementsystem*

ODBC: *Open Database Connectivity*

ODL: *Object Definition Language*

ODMG: *Object Database Management Group*

OID: *Objectidentifier*

OLTP: *On-Line Transaction Processing*

OLAP: *On-Line Analytical Processing*

OMG: *Object Management Group*

OODBMS: *Object-Oriented Databasemanagementsystem*

OQL: *Object Query Language*

ORB: *Object Request Broker*

OSI: *Open System Interconnection*

OTS: *Object Transaktion Service*

PC: *Personal Computer*

PDF: *Portable Document Format*

PKI: *Public Key Infrastructure*

POP: *Post Office Protocol*

POS: *Persistent Object Service*

PSS: *Persistent State Service*

RDA: *Remote Data Access*

RDBMS: *Relational Databasemanagementsystem*

RM-ODP: *Reference Model - Open Distributed Processing*

RPC: *Remote Procedure Call*

SAA: *System Applications Architecture*

SAG: *SQL Access Group*

SDIF: *SGML Document Interchange Format*

SGML: *Standard Generalized Markup Language*

SSI: *Server-Side Includes*

SMTP: *Simple Mail Transfer Protocol*

SNMP: *Simple Network Management Protocol*

SQL: *Structured Query Language*

SSL: *Secure Socket Layer*

TCO: *Total Cost of Ownership*

TCP: *Transmission Control Protocol*

TCP/IP: *Transmission Control Protocol/Internet Protocol*

TNS: *Transparent Network Substrate*

TQM: *Total Quality Management*

UDP: *User Datagram Protocol*

URI: *Uniform Resource Identifiers*

URL: *Uniform Resource Locator*

UTC: *Universal Time Coordinated*

VRML: *Virtual Reality Modeling Language*

WRB: *Web Request Broker*

WWW: *World Wide Web*

XML: *Extensible Markup Language*

XSL: *Extensible Stylesheet Language*

LITERATURVERZEICHNIS

[ABD+89] Malcolm Atkinson, François Bancilhon, David DeWitt, Klaus Dittrich, David Maier, and Stanley Zdonik. The object-oriented database system manifesto. In *Proceedings of the 1st International Conference on Deductive Object-Oriented Database Conference (DOOD'89)*, pages 223–240, 1989.

[AMJ+98] Alessandra Agostini, Giorgio De Michelis, Matthias Jarke, Florian Matthes, John Mylopoulos, Klaus Pohl, Joachim W. Schmidt, Carson Woo, and Eric Yu. A three-faceted view of information systems: The challenge of change. *Communications of the ACM*, 41(12):64–70, December 1998.

[ANS86] ANSI Database Architecture Framwork Task Group (DAFTG - ANSI/X3/SPARC). Reference model for dbms standardization. *ACM SIGMOD Record*, 15(1):19–58, March 1986.

[App98] Robert E. Appelbaum. Persistence for business objects. *Object Magazine*, January 1998. GENESIS Development Corporation.

[Atk95] R. Atkinson. Security architecture for the internet protocol (IPSec). IETF request for comments, IETF Internet Engineering Task Force, August 1995. RFC 1825.

[Att96] Karim H. Attia. *Persistenz in objektorientierten Anwendungssystemen: Über die Abbildung von Objektstrukturen auf relationale Strukturen.* Studienarbeit, Universität Hamburg, 1996.

[BCM96] Tim Bienz, Richard Cohn, and James R. Meehan. Portable document format. Reference manual, Adobe Systems, November 1996.

[Bec96] Nuala Beck. *Shifting Gears: Thriving in the New Economy.* Harpercollins, Canada, September 1996.

[Ber92] Alex Berson. *Client/Server Architecture.* McGraw-Hill, New York, 1992.

[BHL95] B. Blakeley, H. Harris, and J.R.T. Lewis. *Messaging and Queuing Using the MQI: Concepts and Analysis, Design and Development.* McGraw-Hill, New York, March 1995.

[BLFF96] T. Berners-Lee, R. Fielding, and H. Frystyk. Hypertext transfer protocol – HTTP/1.0. IETF request for comments, IETF Internet Engineering Task Force, May 1996. RFC 1945.

[BLFIM98] T. Berners-Lee, R. Fielding, U.C. Irvine, and L. Masinter. Uniform resource identifiers (URI): Generic syntax. IETF request for comments, IETF Internet Engineering Task Force, August 1998. RFC 2396.

[BLMM94] T. Berners-Lee, L. Masinter, and M. McCahill. Uniform resource locators (URL). IETF request for comments, IETF Internet Engineering Task Force, December 1994. RFC 1738.

[Boo94] Grady Booch. *Object-Oriented Analysis and Design with Applications.* Benjamin/Cummings Publishing Company, Redwood City, California, 2. edition, 1994.

[Bun98a] Tim Bunce. *DBD::Oracle – an Oracle 7 and Oracle 8 interface for Perl 5.* England, 1998. ftp://ftp.spu.edu/pub/CPAN//modules/by-authors/Tim_Bunce/.

[Bun98b] Tim Bunce. *DBI - The Perl Database Interface.* England, 1998. ftp://ftp.spu.edu/pub/CPAN//modules/by-authors/Tim_Bunce/.

[Cat91] R.G.G. Cattel. What are next-generation database systems ? *Communications of the ACM*, 34(10):31 – 33, October 1991.

[Cat94] R.G.G. Cattel. *Object Data Management: Object-Oriented and Extended Relational Database Systems.* Addison-Wesley, Reading, MA, USA, 1994.

[Cat97] R.G.G. Cattel, editor. *The Object database standard: ODMG 2.0.* Data Management Systems. Morgan Kaufmann, San Francisco, 1997.

[CCA82] An architecture for database management standards. Technical report, Computer Corporation of America, 1982. NBS Spec. Pub. 500-86.

[CFSD90] J. Case, M. Fedor, M. Schoffstall, and J. Davin. A Simple Network Management Protocol (SNMP). IETF request for comments, IETF Internet Engineering Task Force, May 1990. RFC 1157.

[CMRW96a] J. Case, K. McCloghrie, M. Rose, and S. Waldbusser. Coexistence between Version 1 and Version 2 of the Internet-standard Network Management Framework. IETF request for comments, IETF Internet Engineering Task Force, January 1996. RFC 1908.

[CMRW96b] J. Case, K. McCloghrie, M. Rose, and S. Waldbusser. Introduction to Community-based SNMPv2. IETF request for comments, IETF Internet Engineering Task Force, January 1996. RFC 1901.

[CMRW96c] J. Case, K. McCloghrie, M. Rose, and S. Waldbusser. Management Information Base for Version 2 of the Simple Network Management Protocol (SNMPv2). IETF request for comments, IETF Internet Engineering Task Force, January 1996. RFC 1907.

[Cod70] E.F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387, June 1970.

[Cod79] E.F. Codd. Extending the database relational model to capture more meaning. *ACM Transactions on Database Systems*, 4(4):397–434, December 1979.

[Cod81] E.F. Codd. Relational database: A practical foundation for productivity. *Communications of the ACM*, 25(2):109–117, February 1981.

[Col98] Sean Collins. Welcome to the first edition of synergice. *KPMG synergICE*, 1(1), 1998.

[Com90] CompuServe. Graphics interchange format. Programming reference, CompuServe Incorporated, July 1990. GIF89a.

[CR98] Ken A L Coar and D.R.T. Robinson. The www common gateway interface - version 1.1. IETF draft, IETF Internet Engineering Task Force, December 1998. draft-coar-cgi-v11-02.

[Cur97] David Curtis. Java, RMI and CORBA. white paper, Object Management Group, Farmingham, MA, 1997.

[DA98] James Duncan Davidson and Suzanne Ahmed. Java servlet api specification. Version 2.1a, Sun Microsystems, Palo Alto, CA, November 1998.

[Dat95] C.J. Date. *An Introduction to Database Systems.* Systems Programming Series. Addison-Wesley, Reading, MA, USA, 6. edition, 1995.

[DD95] Hugh Darwen and C.J. Date. The third manifesto. *ACM SIGMOD Record*, 24(1):39 – 49, March 1995.

[DH98a] F. Dawson and T. Howes. vCard mime directory profile. IETF request for comments, IETF Internet Engineering Task Force, September 1998. RFC 2426.

[DH98b] Frank Dawson and Paul Hoffman. The vCard v3.0 XML DTD. IETF draft, IETF Internet Engineering Task Force, November 1998. draft-dawson-vcard-xml-dtd-02.

[Dou97] Diana Lady Dougan. Benchmarking the internet: Reaching beyond the bell curve. In *The Harvard Conference on the Internet and Society*, pages 38 – 46, Cambridge, March 1997. Harvard University, O'Reilly & Associates.

[ECM98] ECMAScript Language Specification, 2nd Edition. Standard, European Computer Manufacturers Association, August 1998. ECMA-262.

[Egy97] The Cabinet Arab Republic Of Egypt. Egypt & the 21st century, March 1997.

[EN89] R. Elmasri and S.B. Navathe. *Fundamentals of Database Systems.* Benjamin/Cummings, Redwood City, CA, 1989.

[FB96] N. Freed and N. Borenstein. Multipurpose internet mail extensions (mime) part one: Format of internet message bodies. IETF request for comments, IETF Internet Engineering Task Force, November 1996. RFC 2045.

[FGM+97] R. Fielding, J. Gettys, J. C. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext transfer protocol – HTTP/1.1. IETF request for comments, IETF Internet Engineering Task Force, January 1997. RFC 2068.

[FKK96] Alan O. Freier, Philip Karlton, and Paul C. Kocher. The SSL Protocol - Version 3.0. IETF draft, IETF Internet Engineering Task Force, November 1996. draft-freier-ssl-version3-02.

[Fla98] David Flanagan. *Java in a Nutshell.* O'Reilly, Köln, 2. edition, 1998.

[Gam99] Erich Gamma. Ulc - a portable thin client architecture. Kolloquiumsvortrag, 1999.

[GJS96] James Gosling, Bill Joy, and Guy Steele. *The Java Language Specification.* The Java Series. Addison-Wesley, Reading, MA, USA, August 1996.

[Goo97] Roxane Googin. An nc manifesto. *NC World*, April 1997.

[GR89] Adele Goldberg and Dave Robson. *SMALLTALK-80 - The Language.* Addison-Wesley, Reading, MA, USA, 1989.

[Gri98] Frank Griffel. *Componentware - Konzepte und Techniken eines Softwareparadigmas.* dpunkt Verlag, Heidelberg, 1998.

[HA97] John Hagel and Arthur G. Armstrong. *Net Gain. Profit im Netz - Märkte erobern mit virtuellen Communities.* Gabler, September 1997.

[Har97] *The Harvard Conference on the Internet and Society.* O'Reilly & Associates, Cambridge, March 1997.

[HFPS98] R. Housley, W. Ford, W. Polk, and D. Solo. Internet x.509 public key infrastructure - certificate and crl profile. IETF draft, IETF Internet Engineering Task Force, September 1998. draft-ietf-pkix-ipki-part1-11.

[Hoa72] C.A.R. Hoare. Notes on data structuring. In O.-J. Dahl, E.W. Dijkstra, and C.A.R. Hoare, editors, *Structured Programming.* Academic Press, New York, 1972.

[HR83] Andreas Härder and Andreas Reuter. Principles of transaction-oriented database recovery. *ACM Computing Surveys*, 15(4), December 1983.

[HW98] Christian Homburg and Harald Werner. *Kundenorientierung mit System: Mit Customer Orientation Management zu profitablem Wachstum*. Campus Verlag, Frankfurt, February 1998.

[IBM95] MQSeries - an introduction to messaging queuing. Technical report, IBM, June 1995. Document Number GC33-0805-01.

[IBM97] Introduction the the open blueprint : A guide to distributed computing. Technical report, IBM, October 1997. Document Number G326-0395-03.

[IET80] User datagram protocol (UDP). IETF request for comments, IETF Internet Engineering Task Force, August 1980. RFC 768.

[IET81a] Internet protocol (IP). IETF request for comments, IETF Internet Engineering Task Force, September 1981. RFC 791.

[IET81b] Transmission control protocol (TCP). IETF request for comments, IETF Internet Engineering Task Force, September 1981. RFC 793.

[ISO92] ISO/IEC. Information technology - Database languages - SQL. Standard, ISO/IEC, 1992. ISO/IEC 9075.

[ISO94] ISO/IEC. Information technology - digital compression and coding of continuous-tone still images: Requirements and guidelines. Standard, ISO/IEC, 1994. ISO/IEC 10918-1.

[ISO95a] ISO/IEC. Information technology - Database languages - SQL - Part3: Call-Level Interface (SQL/CLI). Standard, ISO/IEC, 1995. ISO/IEC 9075-3.

[ISO95b] ISO/IEC. Open Distributed Processing - Reference Model. Technical report, ISO/IEC, September 1995. ISO/IEC 10746.

[ISO98] ISO/IEC. Information technology - Database languages - SQL - Part 10: Object Language Bindings (SQL/OLB). Standard, ISO/IEC, 1998. ISO/IEC 9075-10.

[ITU97a] ITU. X.500 : Information technology - open systems interconnection - the directory: Overview of concepts, models and services. ITU-T recommendation, ITU International Telecommunication Union, August 1997. ISO/IEC/ITU 9594-1.

[ITU97b] ITU. X.509 : Information technology - open systems interconnection - the directory: authentication framework. ITU-T recommendation, ITU International Telecommunication Union, August 1997. ISO/IEC 9594-8.

[Jac92] Ivar Jacobson. *Object-Oriented Software Engineering - A Use Case Driven Approach*. Addison-Wesley, Reading, MA, USA, 1992.

[Kan98] Joel P. Kanter. *Understanding Thin-Client/Server Computing*. Strategic Technology Series. Microsoft Press, Redmond, 1998.

[Ker89] H. Kerner, editor. *Rechnernetze nach ISO-OSI, CCITT*. 1989.

[KGZ94] Klaus Kilberth, Guido Gryczan, and Heinz Züllighoven. *Objektorientierte Anwendungsentwicklung : Konzepte, Strategien, Erfahrungen.* Vieweg, Braunschweig/Wiesbaden, Germany, 2. edition, 1994.

[Kim96] Won Kim. Object-relational database technologie. White paper, UniSQL, 1996.

[KL86] Brian Kantor and Phil Lapsley. Network News Transfer Protocol (NNTP). IETF request for comments, IETF Internet Engineering Task Force, February 1986. RFC 977.

[KM97] D. Kristol and L. Montulli. HTTP State Management Mechanism. IETF request for comments, IETF Internet Engineering Task Force, February 1997. RFC 2109.

[KPM97a] Electronic commerce. Research report, KPMG Management Consulting, 1997.

[KPM97b] Intranets - a guide for business users. Research report, KPMG Management Consulting, 1997.

[LS87] Peter C. Lockemann and Joachim W. Schmidt, editors. *Datenbank-Handbuch.* Informatik-Handbücher. Springer-Verlag, Berlin, Germany / Heidelberg, Germany / London, UK / etc., 1987.

[Mar93] James Martin. *Principles of Object Oriented Design.* Prentice Hall, Englewood Cliffs, NJ, 1993.

[Mar96] James Martin. *Cybercorp: The New Business Revolution.* AMACOM, New York, October 1996.

[Mat93] Florian Matthes. *Persistente Objektsysteme: Integrierte Datenbankentwicklung und Programmerstellung.* Springer-Verlag, 1993.

[McG97] Steven McGeady. The digital reformation: Total freedom, risk, and responsibility. In *The Harvard Conference on the Internet and Society*, pages 59 – 65, Cambridge, March 1997. Harvard University, O'Reilly & Associates.

[Mey90] Bertrand Meyer. *Objektorientierte Softwareentwicklung.* Hanser, Wien, 1990.

[Mey94] Bertrand Meyer. *Reusable software : The Base object-oriented component libraries.* The object-oriented Series. Prentice Hall, Hemel Hempstead, UK, 1994.

[Mic97] Microsoft Corporation. *ODBC 3.0 Programmer's Reference*, 1997.

[Moc87] P. Mockapetris. Domain names - concepts and facilities. IETF request for comments, IETF Internet Engineering Task Force, November 1987. RFC 1034.

[MR96] J. Myers and M. Rose. Post office protocol - version 3 (POP3). IETF request for comments, IETF Internet Engineering Task Force, May 1996. RFC 1939.

[Mue98] Steve Muench. PLSXML Utilities and Demos. Technical report, Oracle, Redwood City, CA, December 1998. http://www.oracle.com/xml/plsxml.

[Neg96] Nicholas Negroponte. *Being Digital.* Vintage Books, January 1996.

[Net97] Writing server-side javascript applications. Version 2.1a, Netscape Communications Corporation, Mountain View, CA, August 1997.

[OG97] The open group architectural framework - version 3. Technical report, The Open Group, Cambridge, December 1997.

[OG98] Network computer. Technical standard, The Open Group, Berkshire, February 1998. Document Number C720.

[OHE96] Robert Orfali, Dan Harkey, and Jeri Edwards. *The Essential Distributed Objects Survival Guide.* Jon Wiley & Sons, New York, 1996.

[OMG92] OMG. Object management architecture guide. Technical report, Object Management Group, 1992.

[OMG95] OMG. CORBAfacilities:Common Facilities Architecture. Technical report, Object Management Group, November 1995.

[OMG97a] OMG. Common Business Objects. White paper, Object Management Group, Farmingham, MA, December 1997.

[OMG97b] OMG. CORBAservices: Common Object Services Specification. Technical report, Object Management Group, Farmingham, MA, July 1997.

[OMG97c] OMG. A discussion of the object management architecture. Technical report, Object Management Group, January 1997.

[OMG97d] OMG. Persistent state service, version 2.0. Request for proposal, Object Management Group, Farmingham, MA, June 1997.

[OMG98a] OMG. CORBA Finance. Technical report, Object Management Group, December 1998.

[OMG98b] OMG. Persistent state service 2.0 - joint revised submission. Request for proposal, Object Management Group, Farmingham, MA, May 1998. orbos/98-05-10.

[OMG98c] OMG. The Common Object Request Broker: Architecture and Specification. Technical report, Object Management Group, Farmingham, MA, February 1998.

[OR93] J. Oikarinen and D. Reed. Internet Relay Chat Protocol (IRC). IETF request for comments, IETF Internet Engineering Task Force, May 1993. RFC 1459.

[Ora96a] Oracle, Redwood City, CA. *Oracle Names Administrator's Guide, Release 2.0*, 1996.

[Ora96b] Oracle, Redwood City, CA. *PL/SQL - User's Guide and Reference, Release 2.3*, 1996.

[Ora96c] Oracle, Redwood City, CA. *Understanding SQL*Net, Release 2.3.3*, 1996.

[Ora97a] Oracle. Network Computing : Increasing the Return on Inverstment. White paper, Oracle Corporation, June 1997.

[Ora97b] Oracle, Redwood City, CA. *Oracle Advanced Networking Option Administrator's Guide, Release 8.0*, December 1997.

[Ora97c] Oracle, Redwood City, CA. *Oracle Call Interface Programmer's Guide, Volumes 1 & 2, Release 8.0*, December 1997.

[Ora97d] Oracle, Redwood City, CA. *Oracle Enterprise Manager Concepts Guide - Release 1.5.0*, December 1997.

[Ora97e] Oracle. Oracle express server: Delivering olap to the enterprise. White paper, Oracle Corporation, Redwood City, CA, January 1997.

[Ora97f] Oracle, Redwood City, CA. *Oracle Net8 Administrator's Guide, Release 8.0*, December 1997.

[Ora97g] Oracle, Redwood City, CA. *Oracle Web Application Server 3.0 - Developing your own Web Application Server Cartridge*, 1997.

[Ora97h] Oracle, Redwood City, CA. *Oracle8 Concepts, Release 8.0*, December 1997.

[Ora97i] Oracle, Redwood City, CA. *Oracle8 Distributed Database Systems, Release 8.0*, December 1997.

[Ora97j] Oracle, Redwood City, CA. *PL/SQL - User's Guide and Reference, Release 8.0*, December 1997.

[Ora98a] Oracle, Redwood City, CA. *Oracle Application Server - Administration Guide - Release 4.0*, 1998.

[Ora98b] Oracle, Redwood City, CA. *Oracle Application Server - Developers Guide - Release 4.0*, 1998.

[Ora98c] Oracle, Redwood City, CA. *Oracle Application Server - Security Guide - Release 4.0*, 1998.

[Ora98d] Oracle, Redwood City, CA. *Oracle SNMP Support Reference Guide, Release 8.0.5*, June 1998.

[OSF91] OSF / Brad Curtis Johnson. A distributed computing environment framework: An osf perspective. White paper, Open Software Foundation, June 1991.

[OSF95] OSF. Introduction to osf dce, release 1.2.2. Technical report, Open Software Foundation, 1995.

[Pos82] Jonathan B. Postel. Simple mail transfer protocol (SMTP). IETF request for comments, IETF Internet Engineering Task Force, August 1982. RFC 821.

[PR85] J. Postel and J. Reynolds. FILE TRANSFER PROTOCOL (FTP). IETF request for comments, IETF Internet Engineering Task Force, August 1985. RFC 959.

[PSW96] Claudia Popien, Gerd Schürmann, and Karl-Heinz Weiß. *Verteilte Verarbeitung in offenen Systemen*. Leitfäden der Informatik. Teubner, Stuttgart, Germany, 1996.

[RBP+91] James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, and William Lorensen. *Object-Oriented Modeling and Design*. Prentice Hall, Prentice Hall, 1991.

[RM90] M. Rose and K. McCloghrie. Structure and Identification of Management Information for TCP/IP-based Internets. IETF request for comments, IETF Internet Engineering Task Force, May 1990. RFC 1155.

[RM91] M. Rose and K. McCloghrie. Management Information Base for Network Management of TCP/IP-based internets: MIB-II. IETF request for comments, IETF Internet Engineering Task Force, March 1991. RFC 1215.

[Rou98] Yvon Le Roux. Electronic commerce - will europe be left behind? In *Trends in Electronic Commerce - TrEC'98*, Hamburg, June 1998. Cisco Systems Europe.

[Sch90] A.-W. Scheer. *Wirtschaftsinformatik - Informationssysteme im Industriebetrieb*. Springer Verlag, Berlin, 3. edition, 1990.

[Sen73] M.E. Senko. Data structures and access in data base systems. *IBM System Journal*, 12:30–93, 1973.

[Sey98] Patricia B. Seybold. *Customers.com : how to create a profitable business strategy for the Internet and beyond*. Times Books, New York, 1998.

[SRL+90] Michael Stonebraker, Lawrence A. Rowe, Bruce Lindsay, James Gray, Michael Carey, Michael Brodie, Philip Bernstein, and David Beech. Third-generation database system manifesto. *ACM SIGMOD Record*, 19(3):31–44, September 1990.

[SS98] Bernhard Schmalzl and Jakob Schröder. *Managementkonzepte im Wettstreit: Total Quality Management versus Business Process Reengineering*. Beck Juristischer Verlag, München, January 1998.

[SSU91] Avi Silberschatz, Michael Stonebreaker, and Jeff Ullmann. Database systems: Achievemnts and opportunities. *Communications of the ACM*, 34(10):110–121, October 1991.

[SSU95] Avi Silberschatz, Mike Stonebreaker, and Jeff Ullman. Database research: Achievements and opportunities into the 21st century. Report of an nsf workshop, Stanford, May 1995.

[Ste96] Mark Stefik. *Internet Dreams: archetypes, myths, and metaphors.* MIT Press, Cambridge, 1996.

[STS97] Gunter Saake, Can Türker, and Ingo Schmitt. *Objektdatenbanken.* Informatik Lehrbuch-Reihe. International Thomson Publishing, Bonn, Albany, Attenkirchen, 1. edition, 1997.

[Sun97a] JavaBeans. Specification 1.01, Sun Microsystems, Mountain View, CA, July 1997.

[Sun97b] JDBC : A Java SQL API. Specification 1.2, Sun Microsystems, Mountain View, CA, January 1997.

[Sun98a] Enterprise JavaBeans. Specification 1.0, Sun Microsystems, Palo Alto, CA, March 1998.

[Sun98b] Enterprise JavaBeans to CORBA Mapping. Specification 1.0, Sun Microsystems, Palo Alto, CA, March 1998.

[Sun98c] Java remote method invocation - distributed computing for java. Technical report, Sun Microsystems, Palo Alto, CA, September 1998. http://java.sun.com/marketing/collateral/javarmi.html.

[SZ97] Avi Silberschatz and Stan Zdonik. Database systens - breaking out of the box. *ACM SIGMOD Record*, 26(3):36 – 50, September 1997.

[Tap97] Don Tapscott. *The Digital Economy: Promise and Peril in the Age of Networked Intelligence.* McGraw-Hill, August 1997.

[Tes97] Larry Tesler. Apple's internet strategy. In *The Harvard Conference on the Internet and Society*, pages 15 – 24, Cambridge, March 1997. Harvard University, O'Reilly & Associates.

[Tha98] Rodney Thayer. Pki requirements for ip security (IPSec). IETF draft, IETF Internet Engineering Task Force, September 1998. draft-ietf-ipsec-pki-req-01.

[TLT98] Don Tapscott, Alex Lowy, and David Ticoll, editors. *Blueprint to the Digital Economy.* McGraw-Hill, 1998.

[Vis95] ParcPlace-Digitalk, Sunnyvale, CA. *VisualWorks User's Guide, Rev. 2.0*, October 1995.

[W3C98a] Extensible markup language (xml) 1.0. Recommendation, World Wide Web Consortium, February 1998. REC-xml-19980210.

[W3C98b] HTML 4.0 Specification. Recommendation, World Wide Web Consortium, April 1998. REC-html40-19980424.

[WCS96] Larry Wall, Tom Christiansen, and Randal L. Schwartz. *Programming Perl.* O'Reilly & Associates, Sebastopol, CA, 2. edition, September 1996.

[Weg98] Peter Wegner. Why interaction is more powerful than algorithms. *Communications of the ACM*, 40(5):80 – 91, May 1998.

[Wet94] Ingrid Wetzel. *Programmieren mit Style : über die systematische Entwicklung von Programmierumgebungen.* Dissertation, Universität Hamburg, Frankfurt, 1994.

[WN95] Kim Walden and Jean-Marc Nerson. *Seamless object-oriented software architecture : analysis and design of reliable systems.* The object-oriented Series. Prentice Hall, Hemel Hempstead, UK, 1995.

[X/O92] X/Open. Distributed TP: The XA Specification. Technical standard, The Open Group, Berkshire, February 1992. Document Number C193, ISBN 1-872630-24-3.

[X/O94] X/Open. Distributed TP: The XA+ Specification, Version 2. Technical standard, The Open Group, Berkshire, July 1994. Document Number S423, ISBN 1-85912-046-6.

[X/O95a] X/Open. Data management: Sql call level interface (CLI). Technical standard, The Open Group, Berkshire, April 1995. Document Number C451, ISBN 1-85912-081-4.

[X/O95b] X/Open. Distributed TP: The TX (Transaction Demarcation) Specification. Technical standard, The Open Group, Berkshire, April 1995. Document Number C504, ISBN 1-85912-094-6.

[X/O96] X/Open. Distributed TP: Reference Model, Version 3. Guide, The Open Group, Berkshire, February 1996. Document Number G504, ISBN 1-85912-170-5.

[YHK95] W. Yeong, T. Howes, and S. Kille. Lightweight directory access protocol (LDAP). IETF request for comments, IETF Internet Engineering Task Force, March 1995. RFC 1777.

[Zül98] Heinz Züllighoven. *Das objektorientierte Konstruktionshandbuch nach dem Werkzeug & Material-Ansatz.* dpunkt,lehrbuch. dpunkt.verlag, Heidelberg, 1998.

[Zon97] Java and the enterprise. White paper, Zona Research, 1997.

INDEX