Dr. R. Steinmetz
- 7. März 1995

Eingeladener Beitrag zu einem Sonderheft IT+TI.
Erscheint Herbst 1995

[SBKo95] *Ralf Steinmetz, Gerold Blakowski, Daniel Köhler;* **Architektur und Implementierung eines verteilten Multimedia-Kiosksystems;** erscheint in it+ti Herbst 1995

Architektur und Implementierung eines verteilten Multimedia-Kiosk-Informationssystems

Ralf Steinmetz
Daniel Köhler

IBM European Networking Center
Vangerowstraße 18 • 69115 Heidelberg • Germany

Phone: +49-6221-59-3000 • Fax: +49-6221-59-3300

steinmetz@vnet.ibm.com

Zusammenfassung: Ein Kiosk-System ist ein i. allg. öffentlich zugängliches, rechnerbasiertes System, bei dem der meist annonyme Anwender meistens für eine kurze Zeit Informationen abruft und Transaktionen veranlaßt. Die Daten können dabei als diskrete Medien (wie Text und Grafik) und als kontinuierliche Medien (wie Audio und Video) vorliegen. In diesem Beitrag werden exemplarisch für ein verteiltes Multimedia-Kiosk-Informationssystem die wesentlichen konzeptionellen Aspekte, der "Abstrakte Kiosk" als Datenrepräsentation, die Architektur, die Komponenten und die Erfahrungen bei der Implementierung und dem Einsatz eines solchen Systems beschrieben. Dabei zeichnet sich das hier betrachtete System durch eine optimale Anpassung der benötigten Funktionalität an die verschiedenen Rollen der Personen aus, die mit diesem System arbeiten; dies sind der Ersteller, der Betreiber und der eigentliche Anwender. Diese Anpassung bedingt eine neuartige Systemstruktur von Kontrolleinheiten, Manipulationseinheiten und einer gemeinsamen verteilten Datenhaltung.

Schlagworte: Multimedia, Hypermedia, Kiosk, interaktives Multimedia Dokument

1 Einleitung

In Anlehnung an [Stei93] verstehen wir im folgenden Multimedia als die integrierte rechnergestützte Verarbeitung von unabhängigen Daten zeitunabhängiger (diskreter) und zeitabhängiger (kontinuierlicher) Medien. Die Forschung im Bereich Multimedia hat sich bisher im wesentlichen auf technologische Aspekte und einige wenige ausgezeichnete Anwendungsfelder konzentriert. Auf der Technologieseite sind dies u.a. Kompressionsverfahren für Audio und Video, Datennetze im Hochgeschwindigkeitsbereich, Kommunikationssysteme, Betriebssystemaspekte, Betriebsmittelverwaltung, Dateisysteme, Datenbanken, Benutzerschnittstellen, Synchronisation und Hypertext/Hypermedia. Die vorzugsweise bisher betrachteten Anwendungsgebiete sind Videokonferenzsysteme, elektronische Post, allgemein kooperatives Arbeiten[1] und interaktives Fernsehen. Dabei liegt immer noch der Schwerpunkt in der Technologie und nicht im anwendungsnahen Umfeld.

Dies sieht aus kommerzieller Sicht etwas anders aus. Der eigentlichen Euphorie und den aus kommerzieller Sicht extrem hohen Erwartungen in diesem Bereich steht eine heute noch eher nüchterne Bilanz des zur Zeit erzielten Umsatzes gegenüber. Dabei ist zu beachten, daß bei diesen Angaben oft der Umsatz in der Medienbranche (als Informationsanbieter) und der der Telekom (als Informationsverteiler) hier zugeschlagen werden. Den durch "Multimedia" verbesserten Informationsauskunfts- und Transaktionssystemen kommt jedoch heute schon eine besondere Bedeutung zu; hier stehen heute nicht nur Prototypen und Produkte bereit, sondern diese sind und werden schon täglich genutzt. Einige bekannte Beispiele sind die verschiedensten Touristenauskunftssysteme, die Bankselbstbedienung und die Museumsinformationskioske. Im wesentlichen handelt es sich hier um interaktive Programme, die auf eine intuitive Weise dem Menschen innerhalb von kürzester Zeit die gewünschten Informationen bereitstellen oder die von ihm gewünschten Transaktionen veranlaßt. Heute stehen diese Systeme meistens an öffentlich zugänglichen Orten und sind unter dem Begriff "Kiosk" bekannt geworden.

Oft wird fälschlicherweise das Kiosk-System mit Hypertext bzw. Hypermedia gleichgesetzt. Während der Ersteller und der Leser von Hypermedia-Dokumenten [Niel90] meistens dieselbe Person sind, ist der Benutzer eines Kiosks i.allg. nicht der Ersteller. Zusätzlich existieren bei Kiosk-Systemen die entsprechenden Betreiber. Hypertechnik bezeichnet eigentlich nur eine Strukturierung von Informationen durch Knoten und Kanten. Bei Kiosk-Systemen im allgemeinen und insbesondere bei unserem System bedeuten Kanten jedoch nicht nur ein Zustandsübergang, sondern beinhalten auch eine Besitzhierarchie. Die Knoten sind dafür nicht nur die darstellbaren Einheiten, sondern sind Elemente innerhalb einer Hierarchie, bei der einige dieser Elemente eine grafische Ausprägung besitzen.

In diesem Beitrag motivieren und beschreiben wir die Architektur eines neuartigen Kiosk-Systems. Den Kern bildet ein extrem effizientes (und einfaches) Laufzeitsystem, welches die Abläufe im System nach den Regeln der hier vorgestellten Datenstruktur koordiniert. Diese Struktur und dazugehörige Inhaltsdaten liegen in Form einer objektorientierten Darstellung vor, die in einer persistenten verteilten Datenhaltung gespeichert wird. Die Darstellung der Inhalte übernehmen die einzelnen Objekte, vom Laufzeitsystem dazu veranlaßt, selbst. Dedizierte Editoren, Konfiguratoren und Navigationshilfen unterstützen sowohl den Ersteller als auch den Betreiber und eigentlichen Anwender des Systems.

Die Realisierung dieses Systems erfolgte nach einer Serie intensiver Voruntersuchungen und Studien alternativer bestehender Werkzeuge zur Generierung multimedialer Präsentations- und Auskunftssysteme wie z.B. das Asymetrix ToolBook[2] oder der Ultimedia Builder/2[3]. Es exi-

1. Computer Supported Cooperative Work, CSCW

stieren überwiegend lokale Systeme, welche zur Erstellung von Hypermediadokumenten bzw. Präsentationen geeignet sind, die aber, bezogen auf ein Kiosk-System nicht die unterschiedlichen Anwender (Benutzerrollen), den Ersteller, den Betreiber und den eigentlicher Benutzer unterscheiden. Die Systeme legen mehr Wert auf ein möglichst mächtiges Werkzeug, mit dem alle Arten einer interaktiven Präsentation gestaltet werden können. Deshalb wird auch bei bestehenden Ansätzen fast immer von einer skript-artigen Sprache ausgegangen, die zur Laufzeit interpretiert wird. Das Erstellen des Skripts erfolgt entweder mit kontextsensitiven Editoren [EHVä93] oder über grafische Programme. Damit wird der eigentliche Ersteller sehr gut unterstützt. Der Betreiber eines Kiosks ist jedoch immer als identisch zum Ersteller angenommen, und der Benutzer selbst kann keine Änderungen vornehmen. Somit kann man entweder als Experte alles ändern, oder man hat nur den Laufzeitmodul und kann nichts ändern. Auch läßt die Schnelligkeit bezüglich des Weiterblätterns leider bei allen Werkzeugen noch zu wünschen übrig.

Im Rahmen des EU-Programms RACE wurde im Projekt BANK [LuOe93] ein Prototyp mit einer ersten verteilten Datenhaltung und Sychronisation zwischen den kommunizierenden Kiosk-Systemen erstellt [Hall94]. Vorarbeiten erfolgten im Rahmen von Diplomarbeiten [HoHe94] und [Krie94]. In diesen Arbeiten wurden (1) alternative Konzepte der Verteilung basierend auf spezifischen Erweiterungen der BERKOM-Protokolle und (2) Varianten der Datenhaltung und internen Strukturierung der Daten analysiert. Unser System ist als eine Integration und Erweiterung der hier erarbeiteten Konzepte und Module in Hinblick auf die Rollenunterstützung und Schnelligkeit zu verstehen.

Das von der Fa. ADI in Karlsruhe entwickelte Kiosk-System unterstützt als erstes diese konkrete Rollenverteilung und ist hinsichtlich der Schnelligkeit beim Blättern optimiert worden. Das hier beschriebene System verwendet einige dieser 'Tricks' in einer allerdings komplett veränderten Systemstruktur, die sowohl wesentlich besser auf eine Verteilung als auch auf eine modulare Erweiterung und Entwicklung zugeschnitten ist.

Die hier vorgestellte Strukturrepräsentation eines interaktiven Multimedia-Dokuments basiert nicht auf einer skript-artigen Zwischensprache wie die meisten Systeme, sondern auf einem erweiterten, objektorientierten Ansatz [KANe93]. Das System ist auf Schnelligkeit ausgelegt und auf die Bedürfnisse der Zielgruppe mit ihrem spezifischen Anforderungsprofil abgestimmt, erhebt aber keinen Anspruch auf Allgemeinheit im Sinne eines offenen Systems. Zu diesem Zweck laufen Arbeiten, die sich mit dem kommenden ISO Standard MHEG (Multimedia and Hypermedia Expert Group) im Allgemeinen und im Kontext von einem offenen Zugriff auf Telekom-Dienste, das BERKOM GLASS-Projekt [GLASS94] befassen.

In diesem Beitrag beschreiben wir zuerst die grundlegenden Konzepte und das aus dem Einsatzfeld extrahierte Anforderungsprofil für ein Kiosk System. Anschließend wird die Abstraktion eines Kiosks dargestellt, d.h., es wird die interne Struktur der Daten zur Spezifikation interaktiver Multimedia-Präsentationen erläutert. Hieraus werden die Architektur und die Komponenten des realisierten Systems abgeleitet. Abschließend werden der Stand der Implementierung und der Einsatz, sowie die nächsten Schritte ausgeführt.

---

2. Produkt der Asymetrix Corporation
3. Produkt der IBM Corporation

2 Ein Kiosk und sein Anforderungsprofil

In Anlehnung an [HoHe94] verstehen wir unter einem **Kiosk-Systemein** i. allg. öffentlich zugängliches rechnerbasiertes System, bei dem der vorzugsweise anonyme Anwender meistens für eine kurze Zeit Informationen abruft und Transaktionen veranlaßt.

Als einführende **Beispiele** bekannter Kiosk-Systeme, die in der Fachliteratur öfters besprochen werden, seien hier das Programm zur Mercedes C-Klasse und der Karstadt Music Master erwähnt.

- Mit dem Programm über alle Varianten, Ausstattungsmerkmale und Preise der Mercedes C-Klasse kann man auch Probefahrten "buchen". Insgesamt wurde dieses Programm ca. 40.000 mal auf Diskette verteilt und hierdurch wurden dann ca. 26.000 Probefahrten vereinbart [Holt94]. Allerdings haben die Anwender auch in Durchschnitt ca. 59 Minuten mit diesem Programm verbracht.
- Der Karstadt Music Master erlaubt als interaktives Informations-, Werbungs- und Bestellsystem das Anhören von CDs und das Bestellen, wenn der gewünschte Titel nicht in dem jeweiligen Geschäft auf Lager ist. Hier hat sich gezeigt, daß die 1994 in 25 Filialen je zwischen einem und vier installierten Geräte im Durchschnitt von 200 Anwendern pro Tag verwendet werden.

Beide Beispiele zeigen, daß Kiosk-Systeme unterschiedliche Ausprägungen besitzen können und von Kunden akzeptiert werden, somit in diversen Umfeldern heute produktiv und zum Vorteil von Kunde und Betreiber eingesetzt werden können. Als weiteres Marktsegment sei hier der Bankenbereich und die Touristikbranche genannt, in welchen Selbstbedienungssysteme effizient eingesetzt werden können.

Als Erfolgsfaktor und Grund der Installation von solchen Kiosk-Systemen gilt nicht die Rationalisierung im Sinne von Personaleinsparung, sondern eigentlich die Kommunikative Rationalisierung [Holt94]. Wir stehen danach heute vor einer Informationsflut, die optimiert werden sollte. Dies kann durch mehr Interaktion mit Multimedia-Systemen im Gegensatz zu dem reinen Konsumieren über Zeitung, Zeitschrift, Radio und Fernsehen geschehen. Solche interaktiven Systeme wurden im Kontext von Hypertext und Hypermedia zuerst bekannt. Routinearbeiten können auf Kiosk-Systeme abgewälzt werden, wodurch Mitarbeiter entlastet werden. Damit bleibt zum einen mehr Zeit für z.B. individuelle Kundenbetreuung, zum anderen können sich Kunden bereits vorher grob über Produkte/Dienstleistungen informieren und so z.B. mit gezielteren Fragen in ein beratendes Gespräch gehen oder dieses ganz vermeiden. Durch Hinzunahme von Werbung und Verfügbarkeit auch außerhalb normaler Geschäftszeiten können wesentlich mehr Kunden angesprochen werden und z.B. Dienstleistungen in Anspruch nehmen, was insbesondere im Bankenbereich von Vorteil ist.

Hypertext bzw. **Hypermedia** stellen eine nichtlineare Verkettung von Informationseinheiten [Stei93] dar. Diese Verkettung selbst wird als Kante oder "Link" bezeichnet, die Informationseinheit als Knoten. Diese Knoten enthalten im Falle Hypertext die konkreten Textpassagen und allgemeiner bei Hypermedia, Daten in den unterschiedlichsten Medien kodiert. Im folgenden betrachten wir der Einfachheit halber "Hypermedia" als gleichbedeutend mit "Hypertext". Die Eigenschaften und Unterschiede von Kiosk-Systemen gegenüber Hypertext können wie folgt zusammengefaßt werden:

- Bei Hypertext steht immer noch das Medium Text im Vordergrund der Betrachtungen. Bei Kiosk-Systemen werden die Informationen auf eine wesentlich intuitivere Art vermittelt; hier stehen Grafik, Video und Audio im Mittelpunkt. Im ggs. zu Hypermedia, womit z.B. detaillierte und ausführliche Nachschlagewerke erstellt werden könnten, soll ein Kiosk eher

"leicht konsumierbar" und der schnellen Orientierung denn der detaillierten Information dienen.

- Die Interaktionsdauer eines Anwenders mit Hypertext-Systemen ist meist deutlich höher als mit Kiosk-Systemen. Dies resultiert aus dem Einsatzfeld für das jeweilige System. Hypertextdokumente stehen meist zusätzlich zu bestimmten Anwendungen (Hilfesysteme) bzw. als eine Form von Nachschlagewerken zur Verfügung. Kiosk-Systeme hingegen stehen an öffentlich zugänglichen Stellen, werden somit meist stehend bedient und sollen dem Benutzer in möglichst kurzer Zeit mit nötigen Informationen versorgen.
- Deshalb kann man auch bei einem Anwender von Hypertext gewisse Handhabungsgrundprinzipien grafischer Oberflächen voraussetzen; dies trifft bei Kiosk-Systemen nicht zu.
- Ein Leser eines Hypertextdokuments ist oft auch ein Autor von solchen Dokumenten (Ausnahme Hilfedokumente) oder kann dies ohne größere Anstrengungen werden. Dies trifft bei Kiosk-Systemen nicht zu, da hier nur der Betreiber bzw. Ersteller als Autor tätig sein kann.
- Bei Hypertext erfolgt ein Zugriff auf verteilte Daten fast immer nach dem "Request on Demand"-Prinzip. D.h., die verteilten Daten werden bei Bedarf übertragen. WWW (world wide web) mit seiner internen HTML-Kodierung (Hypertext Markup Language) der Struktur ist ein solches Beispiel. Bei Kiosk-Systemen werden fast alle notwendigen Strukturdaten auf dem Präsentationssystem gehalten. Nur Inhalte einzelner Seiten werden bei Bedarf auf Anforderung hin geholt. Als Beispiel seien hier die Auszüge des aktuellen Kontostands und der letzten Buchungen in einem Bankselbstbedienungssystem genannt.
- Wesentlich bei Kiosk-Systemen ist auch die Zugriffsgeschwindigkeit auf die Daten sowie die Antwortzeit auf Benutzereingaben. Bei Hypertextdokumenten steht der Inhalt im Vordergrund, Wartezeiten sind hier tolerierbar. Ein Kiosk-System hingegen muß lange Wartezeiten vermeiden, da diese zu Unsicherheiten und Fehlern bei der Bedienung führen können und damit die Akzeptanz des Systems wesentlich beeinflussen. Dies ist besonders wichtig, falls mit dem Kiosk-System Transaktionen durchgeführt werden sollen.
- Insgesamt steht bei Hypertext eher die Informationstechnik im Hintergrund, bei Kiosk-Systemen eher die Werbebranche für Printmedien sowie Audio- und Videoproduktionstudios. Ein Kiosk-System ist im Ggs. zu Hypertextdokumenten nicht auf Benutzereingaben festgelegt sondern kann auch von sich aus aktiv werden oder von außen gesteuert sein und z.B. Werbevideos oder Attraktionsseiten zeigen, die zur Benutzung des Systems auffordern oder Produkte vorstellen. Ein Hypertextdokument wird immer vom Anwender, meist zu einem bestimmten Zweck/Problem konsultiert.

Vor dem eigentlichen Design muß man die Gütekriterien des Anwenders, der mit einem solchen System interagiert, genau betrachten. Hier unterscheiden wir zwischen dem Ersteller der Inhalte, dem Betreiber eines solches Systems und dem eigentlichen Anwender. Der Anwender sollte immer im Vordergrund stehen. Aus einer von uns ausgeführten Studie und aus zahlreichen Kundenprojekten haben wir folgende Kernaussagen gewonnen: Als erster Punkt ist hier die grafische und intuitive Benutzeroberfläche von entscheidender Bedeutung, damit ein Anwender ein solches System annimmt. Leider entsprechen sehr viele heutige Systeme noch nicht dieser Anforderung. Als zweites wesentliches Merkmal hat sich das schnelle Blättern und die schnelle Reaktion des Systems auf Benutzereingaben erwiesen. Hier ist uns bisher kein Kiosk-System bekannt, das bei entsprechend realistischen Inhalten diesen Anforderungen genügt. Außerdem wäre es wünschenswert, wenn in zukünftigen Systemen das Programm zu Hause, z.B. Home Banking, dieselbe Logik und ähnliches Aussehen wie der Kiosk vor Ort hat.

Daneben möchte der Anwender immer aktuelle Informationen erhalten, was wiederum dem Betreiber ein verkürztes Time-to-Market Dienstangebot ermöglicht.

Neben dem Anwender sollte man den Ersteller mit seinen Anforderungen beachten. Betrachtet man den technischen **Erstellungsprozeß**, so steht am Anfang (erster Schritt) die Medienaufnahme und deren Digitalisierung. Konkret sind das Programme zum Scannen, Konvertieren von Formaten, Digitalisieren von Ton und Bewegtbild mit meist anschließender Kompression [Stei94]. Der zweite Schritt ist die unabhängige Medienbearbeitung, d.h. über spezielle Bearbeitungsprogramme für die unterschiedlichen Medien werden die erfaßten Daten noch losgelöst voneinander nachbearbeitet. Im schließlich dritten Schritt findet die eigentliche Medienintegration statt. Hier werden interaktive Systeme zusammengestellt, d.h. die vorbereiteten Inhaltsdaten in logische Einheiten zusammengefaßt und die unterschiedlichsten Arten kausaler und temporaler Beziehungen zwischen diesen festgelegt, jedoch nicht mehr die Inhalte (Medien) selbst verändert.

Den Anforderungen im ersten und zweiten Schritt werden mit zahlreichen, sehr guten Werkzeugen Rechnung getragen, hier muß der Ersteller zwar meistens mit einer Menge unterschiedlicher Systeme arbeiten, kann sich aber die für ihn am besten geeigneten Programme oder Systeme beliebig zusammenstellen. Zur Bewältigung der Anforderungen des dritten Schritts werden heute sogenannte Autorenwerkzeuge eingesetzt, die meist in einem Rapid-Prototypen-Modus arbeiten. Heutige Kiosk-Systeme beinhalten oft die Autorenwerkzeuge und genügen damit den Anforderungen der **Ersteller** sehr gut. Sie haben grafische Benutzerschnittstellen, arbeiten intern mit sehr flexiblen Skript-Sprachen, integrieren Bilder und andere Medieneinheiten in den verschiedensten Formaten und bieten allgemein eine gute Funktionalität, wie z.B. eine große Anzahl von Übergangseffekten beim Szenenwechsel oder Bildaufbau. Dies wirkt sich allerdings nachteilig auf die Geschwindigkeit aus, mit der man durch ein solches Dokument blättern kann. Dabei sind auch fast alle Kiosk-Systeme als primär lokale Systeme - oft mit CD-ROM - konzipiert. Bei der Erstellung solcher Systeme werden bestehende Inhalte oft wiederverwendet, obwohl dies bisher in der Verantwortung des Erstellers selbst liegt und kaum Werkzeuguntertützung besteht. Schlechter wird bisher allerdings die Wiederverwendbarkeit von schon früher realisierten Kiosk-Strukturen unterstützt.

Es bestehen jedoch Anforderungen, denen alle uns bekannten heutigen Systeme nicht gerecht werden. Das wesentlichste dabei ist, daß in den meisten Systemen nicht nur der Anwender und der Ersteller wesentliche Rollen spielen, sondern auch der **Betreiber.** Er möchte vor Ort selbst geringfügige Änderungen am Inhalt eines Kiosks ausführen, ohne dabei Experte (als Ersteller) sein zu müssen. Er möchte beispielsweise den bestehenden Text aktualisieren, Bilder ändern oder einzelne Pfade innerhalb des Systems aus- oder einblenden. Oft sollen auch weitergehende Aktualisierungen durch Anlegen, Entfernen und Verschieben einzelner Seiten möglich sein. Dieser aktualisierte Inhalt muß dann automatisch an alle laufenden Systeme versendet werden. Bei allen Änderungen soll die bestehende Corporate Identity allerdings immer bestehen bleiben. Am besten wäre es, wenn der Betreiber alle seine Veränderungen nur im Rahmen der vorab verabredeten Regeln des Corporate Identity ausführt. Auch hierbei ist zu beachten, daß der Betreiber eigentlich nicht unbedingt Informatik-Experte sein möchte, und er möchte nicht zusätzlich mit allen Details und Tricks beim Umgang mit einem solchen Werkzeug belastet werden. Ein Betreiber muß auch seine bestehenden Betriebsmittel und die bestehende Infrastruktur weiterverwenden, weshalb das Kiosk-System eine offene Schnittstelle zur Anbindung anderer Komponenten bieten muß. Der Betreiber wurde bei globalen Systemarchitekturen bisher immer außen vor gelassen, wir sehen jedoch, daß das Kiosk-System selbst den Betreiber unterstützen muß.

Das in diesem Kapitel beschriebene Anforderungsprofil konnte mit den bestehenden und von uns untersuchten Werkzeugen nicht befriedigend gelöst werden. Dafür wurden z.B. im Rahmen einer Kooperation mit dem Asymetrix ToolBook[4] interaktive Dokumente erstellt und unter MS-Windows entsprechende Programme geschrieben, allerdings blieb dann leider immer die Effizienz und die Möglichkeit einer modularen Weiterentwicklung auf der Strecke. Neben einer auf Schnelligkeit der Darstellung bei gleichzeitig hoher Flexibilität optimierten Architektur ist das Ziel des hier vorgestellten Systems, die unterschiedlichen Benutzerrollen Ersteller - Betreiber - Anwender zu ermöglichen. Im folgenden Abschnitt werden die Grundkonzepte der entstandenen, neuartigen Systemstruktur vorgestellt.

3 Vorüberlegungen

In diesem Abschnitt wird mit der Darstellung der grundlegenden Konzepte auch die im weiteren verwendete Terminologie festgelegt. Das Ziel ist dabei, die Entscheidungsgrundlagen für die Strukturierung der konkret entstandenen Komponenten zu verdeutlichen.

Zu Anfang des Projektes sollte eine Abstraktion eines Kiosks im Hinblick auf eine flexible Struktur und effiziente Realisierung gefunden werden. Zur verteilten Datenhaltung sollte die für das BANK Projekt [LuOe93] entwickelte Datenbankschnittstelle verwendet werden, welche durchweg für einen effizienten Zugriff konzipiert wurde. Die Datenhaltung arbeitet dabei objektorientiert und gestattet die Definition von Objektklassen, Attributen und Beziehungen (Relationen). Damit bestand die Abstraktion im Definieren von Objektklassen und geeigneten Beziehungen mit den zugehörigen Attributen, wodurch eine abstrakte Beschreibung einer beliebigen Kiosk Struktur möglich sein sollte.

In Anlehnung an Datenbankkonzepte kann diese grundlegende Struktur als "Schema" bezeichnet werden. Einen Kiosk zu initialisieren bedeutet das "Füllen" des Schemas mit konkreten Objekten. Das bedeutet, um ein reales Kiosk-System zu erhalten, werden in der Datenhaltung entsprechende Instanzen von Klassen abgelegt. Eine Instanz, also das Objekt einer Klasse, erhält alle in der Klasse beschriebenen Attribute und kann über die zwischen den Klassen beschriebenen Beziehungen mit anderen Objekten verknüpft werden. Ein Objekt ist somit eine "reale" Abbildung der Klasse, das physischen Speicherplatz benötigt und mit Daten gefüllt werden kann. Welche Daten ein Objekt selbst aufnehmen kann, legen die für die Objektklasse definierten Attribute fest.

Das zu definierende Schema, von uns im folgenden **Abstrakter Kiosk** genannt, liegt jedem unserer Kioske zugrunde. Jedes Laufzeitsystem unserer Kioske verwendet denselben Abstrakten Kiosk. Vergleicht man dies mit einem Texteditor, so bearbeitet dieser Dokumente. Alle Dokumente werden dabei intern mit derselben Datenstruktur "dargestellt", diese ist mit dem Abstrakten Kiosk vergleichbar.

Dies bringt einen gravierenden Vorteil gegenüber skript-basierten Systemen: Um einen neuen Kiosk zu entwickeln, ist es nicht nötig das Laufzeitsystem zu verändern (gemeint ist der Teil eines Skripts, welcher die Funktionalität beschreibt), sondern lediglich die Datenhaltung mit anderen Objekten zu füllen. Im Gegensatz zu einer Skriptprogrammierung, in der jedesmal die Objekte, deren Eigenschaften und deren Verwendung (Funktionalität) neu programmiert werden müssen, müssen hier lediglich die definierten Objekte entsprechend instanziert und verknüpft werden. Dies hat zur Folge, daß die Funktionalität eines Objektes bei unserem Ansatz nicht erst zur Laufzeit, d.h. wenn das Skript interpretiert wird, bekannt wird, sondern schon in

4. Asymetrix®, ToolBook® sind eingetragene Warenzeichen der Asymetrix Corporation

das Laufzeitsystem selbst implementiert werden kann. Dies wirkt sich besonders vorteilhaft auf die Geschwindigkeit des Systems aus.

Den scheinbaren Nachteil den man sich damit erkauft, ist der höhere Aufwand der nötig wird um eine neue Objektklasse zu implementieren, da hier nicht in einer "einfachen" Skriptsprache, sondern in unserem Fall ein C oder C++ Modul implementiert werden muß. "Scheinbar" deswegen, weil sich zeigte, daß durch sorgfältige Klassenbildung und Abstraktion viele Objekte teilweise vollkommen gleiche Funktionalität erhalten, die somit nur ein einziges mal implementiert werden mußte. Ein Beispiel dafür ist das Handhaben der Beschriftung von Objekten: Auswahl des Schrifttyps, setzen von Schriftstil und Schriftfarbe und schließlich das Ausgeben des Textes auf dem Bildschirm kann für alle Objekte über dieselben Funktionen geschehen. Wesentlich ist auch, daß die Funktionalität der Objekte lediglich für die jeweilige Objektklasse einmal implementiert werden muß, unabhängig von der tatsächlich im Kiosk vorkommenden Anzahl instanzierter Objekte. Der Vorteil der wesentlich effizienteren und auch speichersparenderen Implementierung wiegt den etwas höheren Aufwand der Programmierung bei weitem auf. Außerdem haben die wenigsten Kioske derart unterschiedliche Funktionen, als daß ständig neue Objektklassen implementiert werden müßten.

Man erkennt, daß es vor der eigentlichen Implementierung nötig ist, festzustellen welche Objekte von einem Kiosk System benötigt werden und eine Struktur zu entwerfen, durch deren Verwendung wirklich alle Anwendungsfälle abgedeckt werden. Daß dies nicht auf Anhieb gelingen kann spielt keine größere Rolle, wichtig ist nur, daß die grundlegende Struktur vollständig definiert wird. Hinzufügen von Objekten bzw. Erweiterungen in einzelnen Objekten sind relativ einfach auch nachträglich durchführbar. Nach vielen Diskussionen entstand der in Abbildung 1 gezeigte Abstrakte Kiosk als grundlegende Struktur für unsere Kiosk-Systeme. Dieser wurde im Laufe der Projektlaufzeit immer wieder überarbeitet und bezüglich der unterschiedlichsten Aspekte optimiert und soll im folgenden näher erläutert werden.

4 Abstrakter Kiosk

Die Elemente des Abstrakten Kiosks können, wie in Abbildung 1 dargestellt, entweder abstrakte Klassen oder konkrete Klassen/Objekte sein.

Beispiele **abstrakter Klassen** sind die Elemente Item und Target (siehe auch Abbildung 1). Diese abstrakten Klassen fassen gemeinsame Eigenschaften anderer Klassen in Form einer Hierarchie zusammen, um sowohl bei der Realisierung als auch später bei der Laufzeit des Systems die Vorteile der Wiederverwendbarkeit und Erweiterbarkeit optimal auszunutzen. Bei der Instanzierung eines konkreten Kiosks mit seinen eigentlichen Inhalten existieren aber nie Objekte dieser abstrakten Klassen.

Beispiele **konkreter Klassen** und Elemente sind Druckknöpfe (Button) sowie Eingabefelder (Entryfield). Diese sind gleichzeitig aktive bzw. eingabefähige Elemente. Der Benutzer kann in diesem Fall Aktionen, die den Ablauf einer Präsentation beeinflussen, auslösen. Daneben gibt es noch sog. passive Elemente die von der Klasse Item abgeleitet sind; das Anzeigen einer Grafik - alleine - ist ein solches Beispiel. Im typischen Fall befinden sich auf einer Seite neben der Grafik weitere Druckknöpfe und andere Elemente, über die der Anwender mit dem System interagieren kann. Zur Laufzeit existieren somit nur Objekte von konkreten Klassen.

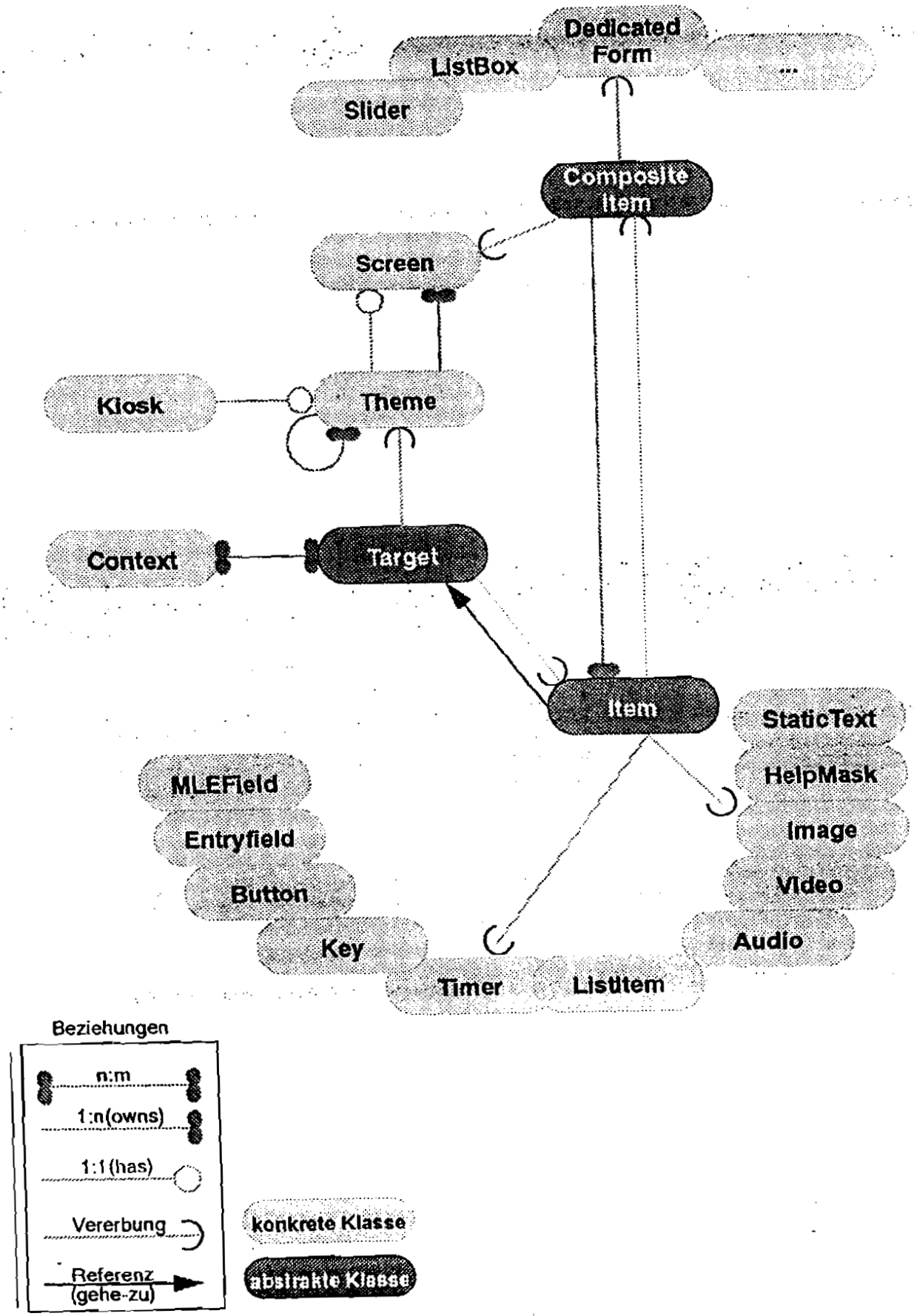

Abbildung 1: Abstrakter Kiosk mit allen Elementen und Beziehungen

Die in unserem Abstrakten Kiosk bestehenden Objektklassen und Zusammenhänge werden in obiger Abbildung verdeutlicht. Zum besseren Verständnis sollen die wichtigsten Klassen mit ihren Beziehungen näher erläutert werden:

- Beginnend bei der Klasse **Kiosk** gelangt man über eine 1:1-Beziehung (Teil sein von: *has*) zu genau einem Thema (Theme), dem sogenannten Anfangsthema. In einem Kiosk Objekt werden dabei nur einige wenige, globale Daten, wie z.B. der Timeout für einen Trailer, gehalten. Im späteren Kiosk existiert nur ein einziges Objekt Kiosk und dient auch als definierter Anfangspunkt für das Laufzeitsystem.
- Ein **Thema** repräsentiert ein bestimmtes Informationsgebiet, welches der Anwender auswählen kann. Ein Thema kann wiederum Themen, vergleichbar mit Kapiteln in einem Buch besitzen. Ein Thema selbst stellt noch keine Repräsentation von Information dar, sondern besitzt zu diesem Zweck einen oder mehrere Seiten, die sog. *Screens*.
- Eine **Seite** (Screen) stellt ein erstes darstellendes Element dar. Es beschreibt den Hintergrund, auf dem Informationen dargestellt werden. Ein Screen ist vergleichbar mit dem Medium Papier eines Buches. Es besitzt z.B. eine bestimmte Größe, Farbe und Struktur, jedoch noch keinen eigentlichen Informationsgehalt. Ein Screen kann mit mehreren Screens gleichgestellt in einem Thema verankert werden, wobei es immer genau eine sog. Anfangsseite (1:1 Beziehung) gibt. Wählt der Anwender ein bestimmtes Thema, weiß so das Laufzeitsystem, welche Seite es zunächst anzeigen muß. Natürlich kann auch direkt von einer Seite auf eine beliebige andere Seite eines anderen Themas gewechselt werden. Die Zusammenfassung von Seiten zu einem Thema wurde vor allem als Hilfe für den Ersteller und Betreiber zur besseren Übersicht eingeführt. Außerdem ist es so möglich beim Löschen eines Themas gleich alle betroffenen Seiten mit zu löschen, was sonst sehr mühsam werden kann. Jede Seite bildet den Präsentationsbereich für die darstellenden Elemente, die *Items*.
- **Darstellende Elemente** (Items) schließlich stellen die Informationsinhalte dar. Jede daraus abgeleitete Itemklasse (Vererbung, s.u.) beschreibt eine bestimmte Art von Information, z.B. Text, Grafik sowie Audio und Video, die beliebig miteinander kombiniert werden können. Sie sind vergleichbar mit den Buchstaben und Grafiken auf einer Buchseite. Außerdem gestatten die Items auch die Kommunikation mit dem Benutzer, da Items nicht nur Information wiedergeben, sondern auch Benutzereingaben empfangen können (z.B. Druckknöpfe). Items können über die 'gehe-zu' Beziehung mit anderen Items oder auch mit Themen oder Seiten verbunden werden und somit eine Themenauswahl, einen Seitenwechsel oder einen Zustandswechsel eines anderen Items veranlassen. So ist die Benutzerschnittstelle nicht festgelegt, sondern kann optimal auf den Benutzerkreis oder die Funktionalität des Kiosk-Systems abgestimmt werden.
- Eine Besonderheit stellt das **Composite Item** dar. Es dient der Zusammenfassung von Items zu einem neuen Element. Damit kann sich ein Composite Item der Grundfunktionalität der Elemente bedienen und diesen eine neue, übergeordnete Funktionalität hinzufügen, wobei das Composite Item die Kontrolle über das gesamte zusammengesetzte Element innehat. Die zusammengefaßten Elemente können neben den Grundelementen wiederum Composite Items sein. So ist z.B. eine Auswahlliste (List Box) eine Kombination von z.B. Text-Items und zwei Button-Items, welche zur Navigation innerhalb der Liste dienen.

Zusammen mit den dargestellten Beziehungen (Relationen) der Klassen läßt sich ein beliebiges Kiosk System auf recht einfache Weise durch den hier gezeigten Abstrakten Kiosk beschreiben. Wichtig in diesem Zusammenhang ist vor allem die Beziehung der Vererbung. Dabei werden nicht nur reine Attribute vererbt, sondern insbesondere die für eine Klasse definierten Beziehungen zu anderen Klassen. Beispielsweise vererbt die abstrakte Klasse *Target*

den Endpunkt einer gehe-zu Beziehung: Somit kann ein Item eine gehe-zu Verbindung zu einem Thema, einer Seite (Vererbung Item->CompositeItem->Screen) einem Composite Item oder einem anderen Item besitzen. Diese vier Möglichkeiten und deren Verwendung durch jede wiederum aus Item abgeleitete konkrete Klasse (Button, Key, etc.) werden durch eine einzige Beziehung ausgedrückt, was letztlich die Implementierung erheblich vereinfacht. Auch die 1:n Beziehung (Element A besitzt viele Elemente B; *ownership*) zwischen CompositeItem und Item wird an die Klasse Screen vererbt, wodurch ein Screen ebenfalls mehrere Items besitzen kann.

Damit ergibt sich die in folgender Abbildung 2 gezeigte Besitzhierarchie zwischen Objekten der verschiedenen Klassen.

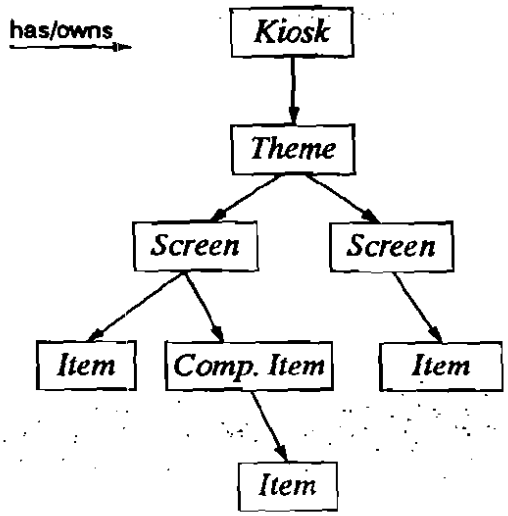

Abbildung 2: Klassenhierarchie

Ein Kiosk kann ein oder mehrere Themen besitzen (wobei nur ein einziges dem Objekt Kiosk direkt zugeordnet wird). Diese wiederum besitzen verschiedene Seiten, welche der Anwender z.B. durch Druck auf entsprechende Druckknöpfe interaktiv auswählen kann. Jeder Screen besitzt wieder seine darstellenden Einheiten, dies sind somit alle aus Item abgeleiteten Klassen, z.B. Druckknöpfe, Textelemente, Grafiken etc.

Durch die Verwendung einzelner Objekte ist auch jederzeit eine Erweiterung bzw. Hinzunahme von neuen Objektklassen sogar zur Laufzeit möglich, ohne daß an der Implementierung der anderen Objekte oder des Laufzeitsystems etwas verändert werden müsste.

5 Systemarchitektur

Basierend auf dem Schema für einen Kiosk, mußte ein entsprechendes System entworfen werden, welches die aus dem Anforderungsprofil abgeleiteten Ziele bezüglich der **Funktionalität** berücksichtigt:

- Optimale Beschreibung des Abstrakten Kiosk, die alle uns bisher bekannten Anwendungsfälle abdeckt und eine 'abstrakte' Implementierung möglich macht.
- Schnelle Schnittstelle zur Datenhaltung und der Benutzeroberfläche, um kurze Antwortzeiten zu erhalten.
- Entkopplung der einzelnen Komponenten (Objektbildung), einerseits um eine unabhängige Entwicklung zu ermöglichen, andererseits um das System je nach der benötigten Funktionalität konfigurieren zu können. Einzelne Komponenten sollen auch später entwickelt und zum laufenden System hinzugefügt werden können, was eine offene Architektur verlangt.
- Relative Unabhängigkeit von dem verwendeten Betriebssystem und der jeweiligen Hardware. Allerdings läßt sich dies bei den komplexen Interaktionen mit der grafischen Benutzeroberfläche nur bedingt realisieren, weil auch die Geschwindigkeit im Vordergrund steht.

Die kritischsten Aspekte bezüglich der **Leistungsfähigkeit** des Systems in Form von Geschwindigkeit und kurzen Antwortzeiten im gesamten Design sind (1) die Minimierung der Interaktionen zwischen der Datenhaltung und den darauf zugreifenden Modulen, (2) das

schnelle Darstellen aller grafischen Elemente im Rahmen der Interaktion zwischen Laufzeitsystem und der grafischen Benutzeroberfläche. Der Abstrakte Kiosk und die Implementierung der verschiedenen Module in Form möglichst unabhängiger Objekte trägt beiden Rechnung.

Beim Systemdesign müssen die Aufteilung der Aufgaben auf ggf. verschiedene Prozesse und die Aufrufmechanismen festgelegt werden. Soll das System modular erweiterbar sein, müssen entsprechende Standardmechanismen wie z.B. die Verwendung von DLLs (dynamic link library) berücksichtigt und in die Architektur aufgenommen werden

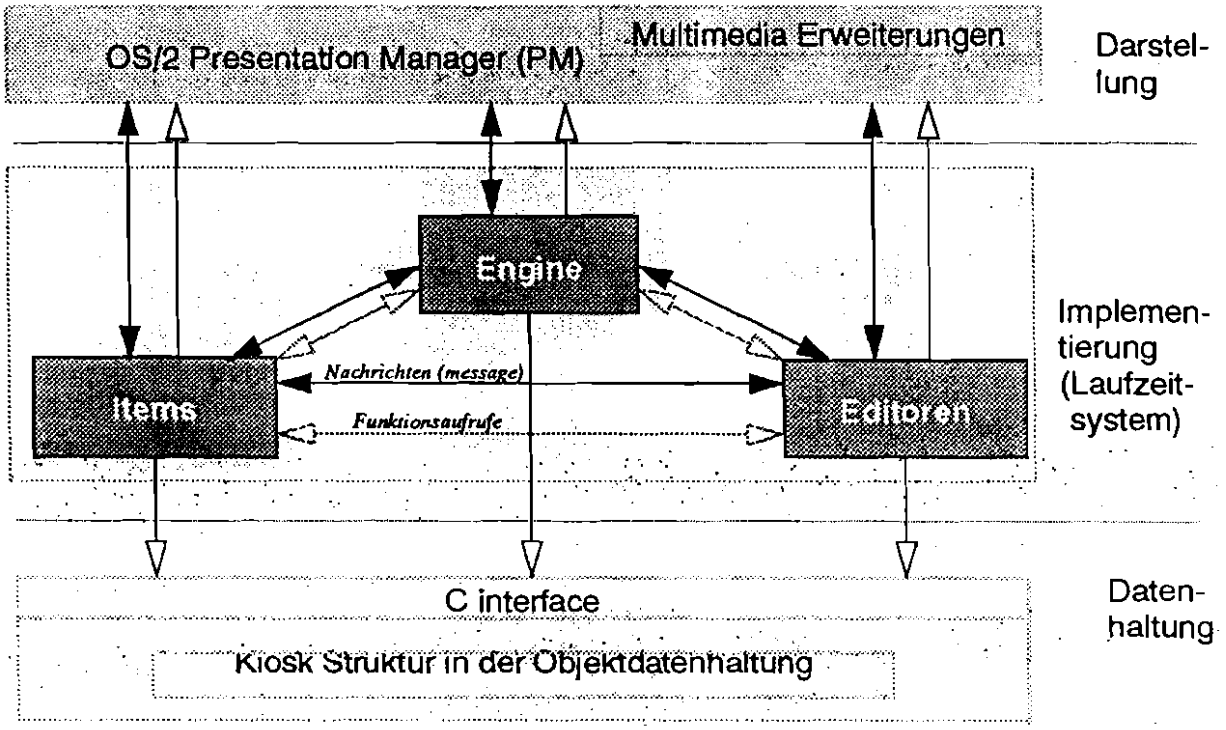

Abbildung 3: Architekturübersicht

Die prinzipielle Architektur des hieraus entstandenen Systems zeigt Abbildung 3. Diese läßt sich grob in drei funktionale Schichten aufteilen. Die Interaktion zwischen den Schichten bzw. Modulen ist durch Pfeile angedeutet. Die oberste Schicht stellt die **grafische Oberfläche** dar. Hier ist die Benutzerschnittstelle realisiert, die je nach Rolle des Benutzers unterschiedlich ausgeprägt sein kann.

- Der Endanwender erhält eine leicht und intuitiv bedienbare Oberfläche zur Anforderung von Informationen, welche in multimedialer Form angeboten werden können.
- Der Betreiber des Systems sollte für seine Inhaltsänderungen ebenfalls eine möglichst einfache, auf das wesentliche konzentrierte Benutzerschnittstelle erhalten. Ideal erscheint es hier, dem Betreiber nicht eine neue Schnittstelle, sondern eine funktional erweiterte Endanwenderschnittstelle zur Verfügung zu stellen. Damit kann der Betreiber über die gleichen (intuitiven) Mechanismen wie der Endanwender im System navigieren, jedoch zusätzlich z.B. über eine erweiterte Tastatur besonderen Einfluß auf das System ausüben, wie z.B. das Ändern von Bildern oder Textfeldern.
- Beim Ersteller hingegen ist die Mächtigkeit der Hilfsmittel sicherlich wichtiger als eine intuitive Oberfläche, jedoch sollte er im Sinne effektiven Arbeitens WYSIWYG-Funktionalität (What You See Is What You Get) erhalten.

Als grafische Oberfläche dient in unserem System der Presentation Manager (PM) von OS/2, erweitert durch den Multimedia-Presentation-Manager (MMPM/2). Dieser ermöglicht die einfache und vor allem standardisierte Programmierung von im System installierten Geräten, wie z.B. Sound-Karten oder Video-Player Hardware, wodurch das Kiosk-System offen und geräteunabhängig wird.

Die zweite Schicht im System bildet die **Implementierung** der Objekte/Module des Systems. Hier ist das Kontrollmodul Engine, die darstellenden Elemente des Abstrakten Kiosks sowie die unterschiedlichen Editormodule für die verschiedenen Benutzerrollen zusammengefaßt. Die Editoren werden je nach Rolle des Benutzers optional geladen bzw. installiert und aktiviert.
Alle Module stehen untereinander in Verbindung und können auch direkt auf die darüberliegende Oberfläche und darunterliegende Datenhaltung zugreifen. Durch Hinzufügen bzw. Entfernen von Modulen, auch zur Laufzeit, kann das System individuell konfiguriert, implementiert und erweitert werden. Der direkte Zugriff der Module auf Oberfläche und Datenhaltung ermöglicht eine flexible und leistungsfähige, d.h. geschwindigkeitsoptimierte Realisierung, da jedes Modul unabhängig voneinander optimal auf die gegebenen Schnittstellen und zu bedienenden Datenformate (Medien) angepaßt werden kann.

Die unterste Schicht beschreibt die **Datenhaltung** [EHVä93]. Diese betrifft nicht nur die Speicherung der Inhalte, sondern vor allem die der Beschreibung bzw. des Abstrakten Kiosk. Dies ist ein wesentlicher Aspekt in unserem Konzept. Durch eine externe Verwaltung des Kioskschemas wird es möglich, durch bloßen Austausch der Datenhaltung das System nicht nur inhaltlich sondern auch strukturell völlig neu zu gestalten, ohne an der Implementierung des übrigen Systems etwas zu verändern.

Um einen besseren Eindruck von der Funktionalität des entstandenen Kiosk Systems zu erhalten werden im folgenden einige der genannten Module und der Aspekt der Erweiterbarkeit näher erläutert.

Engine: Systemkontrolle und Steuerung

Die Kontrolle über das System besitzt das Kontrollmodul, die sogenannte **Engine**. Die Engine stellt das Kernstück des Laufzeitsystems dar, welches die Inhalte des Abstrakten Kiosk aus der Datenhaltung ausliest und in entsprechende Steuerung der Items umsetzt. Die Engine koordiniert sämtliche Vorgänge insoweit, daß sie die Items entsprechend benachrichtigt und für eine geordnete Darstellung derselben sorgt. Dabei greift die Engine zum einen direkt auf die Datenhaltung zu, zum anderen auch direkt auf die Funktionen der grafischen Oberfläche, z.B. zum Anzeigen der Itemfenster. Verlangt ein Item eine Aktion des Systems, z.B. den Wechsel auf eine andere Seite des Kiosks, so benachrichtigt das Item die Engine, welche daraufhin alle nötigen Schritte übernimmt. Dabei ermittelt die Engine aus der Datenhaltung, welche Items von einer Aktion betroffen sind. Wie schließlich ein Item dargestellt wird oder welche Funktion es ausführt, muß das betroffene Item selbst wissen. Die Engine bleibt völlig unabhängig von der Darstellung der Inhalte. Dadurch beschränkt sich das "Wissen" der Engine auf den Abstrakten Kiosk und die Schnittstelle zu den verschiedenen Items.

Item: Darstellende Einheiten

Bis auf wenige Ausnahmen erhält ein Objekt der aus der Klasse *Item* abgeleiteten Klassen ein Fenster auf der grafischen Oberfläche zugeordnet. Reine Datenobjekte, z.B. Listeneinträge

(ListItem), die nur Daten einer Zeile einer Liste enthalten, sind eine solche Ausnahme. Andere Items hingegen, z.B. Audioobjekte erhalten zwar Fenster, nutzen diese aber nicht zur Ausgabe, sondern gewährleisten damit lediglich eine einheitliche Implementierung und damit transparente Handhabung aller Objekte. Fenster solcher Objekte werden niemals dargestellt und haben demnach auch keine Größen und Positionen auf der grafischen Oberfläche.

Ein Fenster besitzt eine sogenannte Fensterprozedur, welche alle Nachrichten an das Fenster verarbeitet. Dies können Nachrichten von der Oberfläche selbst, von der Engine oder von beliebigen anderen Prozessen oder Modulen im System sein. Das Fenster legt den Ein- und Ausgabebereich eines Item-Objekts fest, die zugehörige Fensterprozedur realisiert die Funktionalität. Innerhalb seines Fensters kann ein Item u.a. grafische Ausgaben erzeugen, aber auch Eingaben erhalten. Dabei realisiert die Fensterprozedur nicht genau ein spezielles *Objekt* der Klasse Item, sondern stellt die Implementierung der *Klasse* Item bzw. der daraus abgeleiteten Klassen dar. Beispielsweise wird die aus Item abgeleitete Klasse Image in einer Fensterprozedur realisiert, die die Ausgabe einer Grafik als Aufgabe hat. Jedes Image-Objekt erhält zwar ein eigenes Fenster, jedoch ist die Fensterprozedur für alle Fenster dieser Itemklasse nur ein einziges Mal implementiert.

Ediereinheit

Zur Veränderung der Inhalte in der Datenhaltung ist neben dem reinen Laufzeitsystem ein Ediermodul oder Editor notwendig. Der Editor greift direkt auf die Datenhaltung zu und kann sich, soweit sinnvoll, des restlichen Laufzeitsystems zur Darstellung der Inhalte bedienen. Je nach Rolle des Benutzers des Systems (Betreiber oder Entwickler) stellt das Editormodul unterschiedliche Funktionalität bzw. Einflußnahme auf die Datenhaltung zur Verfügung. Durch direkte Bedienung der Oberfläche kann der Editor noch zusätzliche Funktionen oder Hilfen wie z.B. eine Werkzeugleiste dem Entwickler zur Verfügung stellen. Der Betreiber des Systems will meist nur Inhalte des Kiosk Systems ändern, muß jedoch nicht das Design oder den Aufbau des Kiosks ändern. Er wird somit nur Inhalte bestimmter Seiten verändern, z.B. Themenbereiche ausblenden oder Informationstexte aktualisieren. Der Benutzer schließlich erhält zwar ggf. Eingabemöglichkeiten (z.B. Kontonummer), jedoch keinerlei Editierfunktionen für das System selbst.

Der Editor ist im Vergleich zum Rest des Systems zeitlich unkritisch, hier steht wieder Funktionalität im Vordergrund. Durch die getrennte Realisierung des Editormoduls kann für jede Nutzerrolle der entsprechende Modul hinzugefügt oder individuell angepaßt werden. Der einfache Editor des Betreibers, der nur zum Ändern von Inhalten dient, ist größtenteils direkt in das Laufzeitsystem (Engine, Items) integriert, wird aber erst durch eine besondere Tastenkombination auf einer erweiterten Tastatur aktiviert. Somit kann sich der Betreiber wie der Anwender im System zu der gewünschten Seite bewegen und direkt durch Aktivieren des Editors diese editieren. Damit entfällt eine eigene Oberfläche, und die Einlernphase im Umgang mit dem System wird auf das Minimum reduziert.

Datenhaltung

Die Datenhaltung dient der Speicherung des im vorhergehenden Kapitel beschriebenen Abstrakten Kiosks sowie des daraus abgeleiteten konkreten Kiosks. Ein konkreter Kiosk besteht aus Instanzen der im Abstrakten Kiosk festgelegten Klassen, den sog. Objekten und ihren Beziehungen. Wichtig ist hierbei, daß die Vererbungsbeziehungen, nachdem das Schema

festgelegt wurde nicht mehr verändert werden können, d.h. eine Instanzierung von abstrakten Klassen ist zwar möglich aber nicht sinnvoll da sie keine Implementierung besitzen.

Neben der reinen Objekt-Datenhaltung muß eine Schnittstelle dafür sorgen, daß zum einen die Inhalte der Objekte, zum anderen deren Beziehungen zueinander gesetzt und abgefragt werden können. Die Schnittstelle stellt eine transparente Sicht auf die Objekte zur Verfügung, d.h. ermöglicht Zugriff auf die Objekte unabhängig davon, wie diese physisch gespeichert sind (verteilt, lokal im Speicher, lokale Platte etc.). Dabei stellt die Schnittstelle nur die abstrakten Operationen, wie Lesen/Setzen von Attributen, Anlegen/Löschen von Objekten sowie Setzen/Löschen/Abfragen von Beziehungen zur Verfügung. Vorteilhaft ist, daß sowohl die Kiosk-Objekte als auch der Abstrakte Kiosk über dieselbe Schnittstelle realisiert werden können. Dazu betrachtet man schlicht die Klassen der Objekte selbst wiederum als Objekte von sog. Metaklassen. Damit reduziert sich die "Eigenintelligenz" der Datenhaltung lediglich auf diese Metaklassen, d.h. die Attribute und möglichen Beziehungen der Metaklassen müssen in der Datenhaltung bekannt sein.

Die dargestellte Datenhaltung speichert in den Attributen der Objekte teilweise direkt Inhalte der Kiosk Anwendung (z.B. Beschriftungstexte) und teilweise Referenzen auf Inhalte, z.B. auf Video-/Audio-Dateien. Auch gemischte Verwendung der Attribute für Inhalte und Referenzen ist möglich. Damit wird der Speicherbedarf der Objektdatenhaltung stark reduziert und ermöglicht die Verwendung eines lokalen Cache im Speicher des Präsentationssystems für die Speicherung aller Kioskobjekte und deren Beziehungen. Die zeitraubenden Operationen erstrecken sich damit nur mehr auf die Beschaffung referenzierter Inhalte, d.h. der Grafiken oder Videodaten und damit auch eher auf die Hardware und Betriebssystem, denn auf die Applikationssoftware.

Erweiterbarkeit

Wichtig an dieser Stelle ist, daß ein konkretes Kiosk-System meist aktuelle Informationen darstellen soll; dafür ist es neben einer schnellen Änderung der Inhalte auch notwendig, evtl. neue Techniken oder Informationsquellen in das System einzubinden. Desweiteren benötigen Kioske unterschiedlicher Betreiber meist unterschiedliche Peripheriekomponenten und Kommunikationsmöglichkeiten (z.B. Scheckdrucker oder Schnittstellen zu einem Buchungssystem), die sich ergänzen oder ausschließen können. Um hier flexibel und vor allem Erweiterbar für zukünftige Anforderungen zu bleiben, wurde das objektorientierte Denken konsequent angewendet.

Ein Item ist ein eigenständiges Objekt, welches mit anderen Items oder Objekten kommunizieren kann. Durch Weglassen oder Hinzufügen von Items kann sowohl Funktionalität in der Benutzeroberfläche, aber auch innerhalb des Systems individuell verändert werden. In unserem Fall geschieht dies durch Realisieren von Items in sogenannten *dynamic link libraries* (DLL), einer Funktionalität des OS/2 Betriebssystems. Eine DLL kann während der Laufzeit einer Applikation zu dieser hinzugeladen werden, womit die darin enthaltenen neuen Funktionen zur Verfügung stehen. Das DLL-Item hat die Möglichkeit, sich zu initialisieren und z.B. eine Nachrichtenschnittstelle zu erzeugen, wodurch es wie der Rest des Systems Zugriff auf die Datenhaltung und Oberfläche erhält und sich transparent in das System einbinden kann. Zum Hinzuladen einer DLL muß ihr Name bekannt sein, der als Attribut zu einem Item gespeichert wird, womit ein leistungsfähiges und flexibles Konzept zur Verfügung steht. Neue Funktionen werden in DLLs realisiert und können nachträglich zum Kiosk-System hinzugefügt werden, wobei es unerheblich ist, ob ein DLL-Item eine grafische oder eine interne Aufgabe,

wie z.B. Kommunikation über BTX innehat. Das Item muß nur die kioskinterne Kommunikation interpretieren und anwenden können, um transparent zu bleiben.

Eine andere Art der Erweiterung besteht im Zusammenfassen von bestehenden Items zu einem bereits beschriebenen Composite Item. Ein zusammengesetztes Item beinhaltet wie jedes Item eigene Funktionalität mit dem Unterschied, daß es sich der Funktionalität der von ihm zusammengefügten Items ebenfalls bedienen kann. Da auch solch ein zusammengefügtes Item wieder durch eine DLL realisiert werden kann, ist auch so ein Maximum an Flexibilität erreicht. Durch die einfachen Schnittstellen und weitgehende Selbständigkeit der Items bleibt eine hohe Geschwindigkeit des Gesamtsystems erhalten.

6 Komponenten

Nach Abbildung 4 läßt sich das vollständige Szenario eines Kiosksystems in vier Bereiche zerlegen: Präsentation, Integration, Manipulation und Datenhaltung.

Die **Präsentationskomponenten** stellen das Laufzeitsystem dar. Sie beinhalten die Engine und die Implementierung der verschiedenen Objekte zur Darstellung der Kioskinhalte.

Der **Integrationsbereich** beschreibt Komponenten, die zur Integration besonderer Dienste in das System dienen. Je nach Konfiguration und Standort kann ein Anbieter zusätzliche Objekte in das System integrieren. Diese werden in der Datenhaltung beschrieben und zur Laufzeit des Systems in den Präsentationsbereich übernommen. Die Objekte beinhalten dabei zusätzlich eine Implementierung, die auch vom Anbieter selbst realisiert werden kann.

Der **Manipulationsbereich** stellt eine sehr breit gefächerte Komponente dar. Hier sind Editoren und alle Hilfsmittel zusammengefaßt, die zur Aktualisierung und Verwaltung des Kiosks dienen. Dies sind die verschiedenen, rollenspezifischen Editoren, Datenhaltungs-Hilfsmittel sowie Programme zur Manipulation und Erstellung von Inhalten wie z.B. Grafik, Audio/Video, welche getrennt von der Kioskanwendung genutzt werden. Durch Verwendung von Standardformaten bei den Inhalten steht es dem Anwender frei, welche Hilfsmittel hier zum Einsatz kommen, wodurch eventuell im Betrieb vorhandene Resourcen weiter genutzt werden können. Zur primären Erstellung eines neuen Kiosks steht momentan (1/95) noch kein geeignetes Hilfsmittel zur Verfügung. Daher wird hier z.Zt. über den Umweg einer textuellen Beschreibung der im Kiosk vorkommenden Objekte (ASCII-Datei) und einen Konverter die Datenhaltung befüllt. Hier ist ein grafischer Editor angestrebt, der dem Ersteller zur Verfügung gestellt werden kann und eine leichte Generierung neuer Kioske ermöglicht. Die bereits existierenden Laufzeiteditoren, d.h. insbesondere der Editor des Betreibers arbeitet hingegen direkt auf der Datenhaltung, wodurch auf einfachste Weise schnell Änderungen am Kiosk vorgenommen werden können, wie z.B. Ändern von Texten oder Ein-/Ausblenden und Erweitern von Themen (siehe auch Abbildung 5).

Gemeinsam allen vorgenannten Komponenten ist die beschriebene **Datenhaltung**. Hier ist die Speicherung sowohl des Abstrakten Kiosk als auch der Inhalte zusammengefaßt. Die Präsentations- sowie die Integrationskomponente greifen ausschließlich lesend auf die Datenhaltung zu, wohingegen die Manipulationskomponenten lesend und schreibend hier eingreifen.

Weitere, in Abbildung 4 nicht aufgeführte Komponenten stellen die Installation und die Fernwartung des Systems dar.

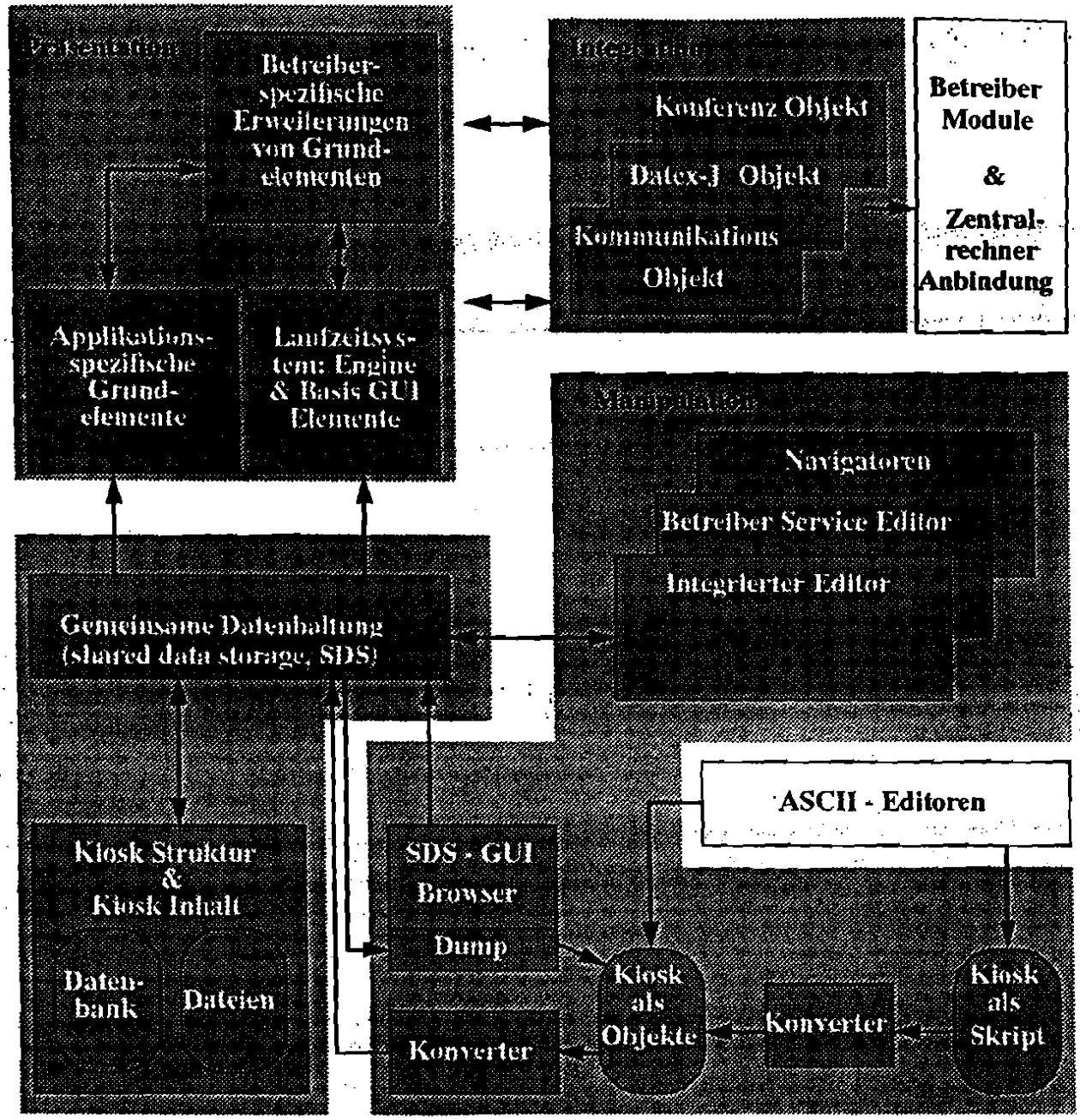

Abbildung 4: Komponenten

Die folgende Abbildung 5 stellt exemplarisch die Auswahl von einer Seite nach vorgegebenen Style Guides dar. Dies ist ein typischer Bildschirm bei der Interaktion des Betreibers mit dem System.

Man erkennt im Hintergrund das normale Laufzeitsystem, d.h. Druckknöpfe zum Verzweigen in bestimmte Themenbereiche sowie Navigationsknöpfe. Der zweite Themenauswahlknopf wurde hier deaktiviert, sodaß der Endanwender später keine Möglichkeit haben wird dieses Thema anzuwählen. Daneben sind noch eine "Werkzeugleiste" mit verschiedenen Hilfen, sowie die "Seitenauswahl"-Liste zu sehen. Hier stehen vordefinierte Seitenlayouts zur Verfügung welche durch einfachen Druck auf die verkleinerten Bilder nach der aktuellen Seite eingefügt werden können.

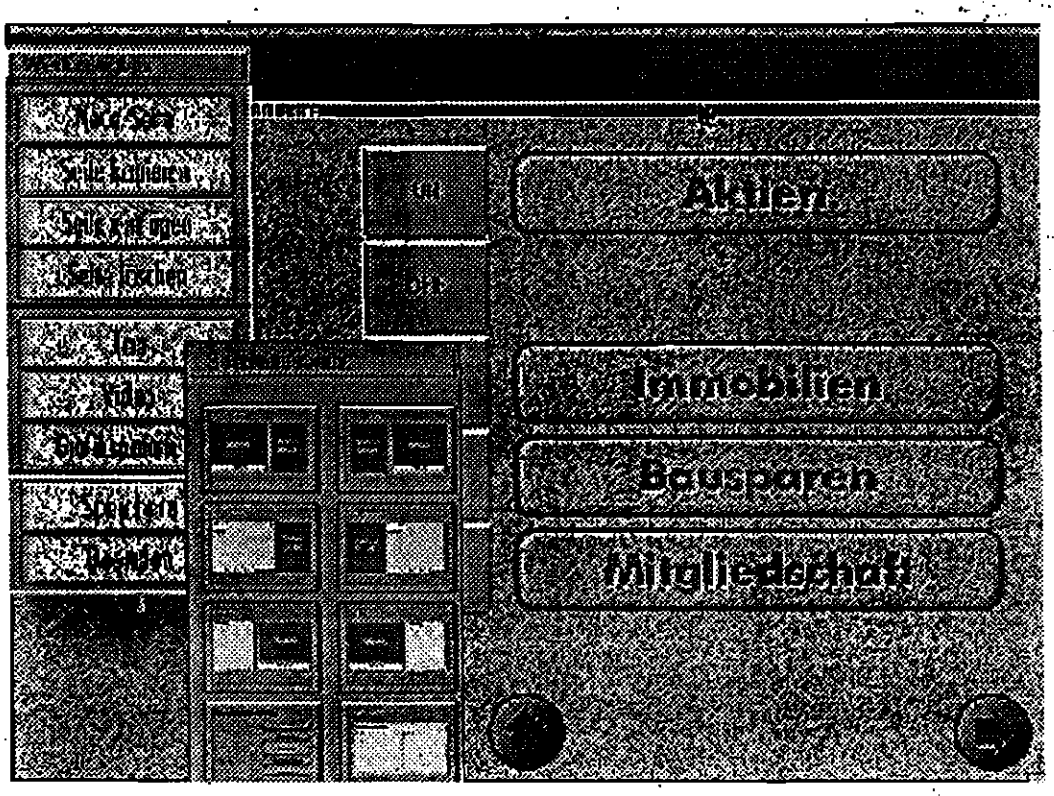

Abbildung 5: Darstellung bei Manipulation eines konkreten Kiosks durch den Betreiber

7 Erfahrungen bei der Implementierung und dem Einsatz

Die ersten Prototypen einzelner Komponenten und Konzepte dieses Systems entstanden Ende 1993. 1994 wurde das System neu strukturiert und in seiner heutigen Form der Version 1.1 realisiert (Stand Januar 1995). Insgesamt stecken bisher (1/1995) insgesamt ca. 5 Mann-Jahre aus der Entwicklung der Fa. IBM in Heidelberg in diesem Projekt. Für die nächsten zwei Jahre sind schon einige konkrete Arbeitspunkte festgelegt. Danach ergibt sich die berechtigte Frage nach ersten **Erfahrungen in der Implementierung** und dem Einsatz.

Bisher fehlt - trotz vieler Ansätze im Hypertext-Bereich - noch ein allgemeingültiges Referenzmodell nach dem man auch ein hier beschriebenes Kiosk-System entwerfen und realisieren könnte, wie dies beispielsweise aus dem Kommunikationsbereich mit dem ISO-OSI-Bereich bekannt ist. Hier greifen weder ODP (Open Distibuted Processing), noch sind die in MHEG (Multimedia Hypermedia Expert Group) erstellten Architekturen geeignet um unsere Funktionalitäten genau einordnen zu können. Jedoch hat sowohl die Entkopplung der einzelnen Module gemäß der in unserer Architektur aufgezeigten Funktionsblöcke als auch die Festlegung auf den Abstrakten Kiosk den Zweck eines Referenzmodells für uns sehr gut erfüllt.

Anfangs dachten wir, daß der Abstrakte Kiosk entsprechend einer Modulschnittstelle zu definieren sei und dann nie mehr verändert wird. Dem war nicht so, dieser Abstrakte Kiosk wurde in mindestens 50 Varianten immer wieder geändert und verbessert. Wir sind sicher, daß er auch heute noch nicht optimal ist, trotzdem haben wir seit geraumer Zeit immer nur noch Ergänzungen eingefügt und keine grundsätzlichen Strukturänderungen vorgenommen.

Als primäre Plattform für dieses System wurde OS/2 mit seinen Multitasking-Fähigkeiten gewählt. Hierdurch ließ sich die im System immanente Parallelität der eigentlichen Engine, der Editoren und weiteren unabhängigen Komponenten sehr gut realisieren. Allerdings haben wir immer wieder mit den unterschiedlichsten kleineren Details und Randbedingungen bei der Presentation-Manager Programmierung viel Zeit verbracht. Als Beispiel sei hier das Font-Handling bei einer möglichst unabhängigen Programmierung des später im System integrierten Grafikadapters und der verwendeten Auflösung zu nennen. Insgesamt hat uns die Aufteilung in relativ unabhängig realisierbare Module, besonders bezüglich der darstellenden Elemente eine verteilte Entwicklung ermöglicht, die zu einer sehr strukturierten Denkweise und wohlbedachten Strukturierung der Klassenhierarchien führte. Es erfolgte eine außerordentlich gute Zuteilung von Funktionalität, Komponente und Zuständigkeit an die involvierten Entwickler.

Die Aufteilung in Kernsystem und dynamische Erweiterbarkeit durch DLLs, erlauben eine modulare funktionale Erweiterung des Systems. Dies kann analog zum Schreiben von Gerätetreibern und deren Integration in das Betriebssystem gesehen werden. Wir haben dieses Vorgehen inzwischen in zwei mit unserem System erstellten Kiosken durchgeführt und haben damit eine aus softwaretechnischer Sicht optimale Entkopplung zwischen Kern und Zusätzen ohne jegliche Leistungseinbuße festgestellt. (Zum Vergleich wurden einzelne Module, die eine Erweiterung des Abstrakten Kiosk um weitere Elemente bedeuten, sowohl im Kern als auch als DLL realisiert).

Wir wollten die Datenhaltung als eine Form des gemeinsamen Speichers (shared memory) mit geeignetem Sperren und zur persistenten Ablage der Struktur- und Inhaltsdaten verwenden. Deshalb wurden in dem Bereich der Datenhaltung ausführliche Vergleiche der bestmöglichen Alternativen ausgeführt. So ergaben sich u.a. folgende Alternativen der Speicherung verteilter persistenter Daten: Verwendung einer Datenbank der zweiten Generation wie DB2/2 (diese relationale Datenbank hat Vorteile bei der Flexibilität, Fehlertoleranz, Stabilität; sie bedeutet jedoch Lizenzgebühren, Installations- und Administrationsaufwand), Einsatz einer Datenbank wie ObjectStore (objekt-orientierte Datenbanken sind dem Problem angepaßter, sie bedeuten ebenso Lizenzgebühren, haben einen hohen Installations- und Wartungsaufwand, sind leider oft noch instabil), Verwendung von Dateien und Netzwerk-Filesystemen und Realisierung einer eigenen auf diesen Fall hin optimierten Datenbank. Es wurde -primär aus Gründen eines zeitoptimalen Zugriffs basierend auf den experimentell verifizierten Vorüberlegungen- ein eigener persistenter Cache mit Anschluß sowohl an eine relationale Datenbank als auch an das Dateisystem realisiert. Durch die Realisierung einer generischen Datenhaltungsschnittstelle konnten wir ohne wesentliche Änderung in unseren Komponenten die oben erwähnten Veränderungen des Abstrakten Kiosk während der Entwicklung ohne größere Probleme vollziehen.

Auch bezogen auf den mehr **praktischen Einsatz** haben wir inzwischen eine Menge von interessanten Erfahrungen gesammelt.

Eine aus technischer Sicht nicht sofort verständliche, generelle Forderung war immer wieder Schnelligkeit vor Funktionalität. Anwender und Betreiber haben immer wieder lieber auf zeitintensive Funktionen verzichtet, wenn damit die Schnelligkeit beeinträchtigt wurde. Somit haben wir auch viele Designentscheidungen bei der Konzeption und Implementierung nach dieser Richtlinie getroffen.

Wie eine Zeitschrift primär an dem Inhalt und deren Aufmachung gemessen wird, so stellen wir bei Kiosksystemen auch vermehrt fest, -sowie die Hardware, Software und das Kiosksystem schnell genug sind- das eigentliche primäre Qualitätsmerkmal der Inhalt und deren Darbietung ist. Bei der Erstellung wird deshalb die Inhaltserstellung oft unterschätzt. Man sollte hier eher den Blick aus der Medien- und Werbebranche haben. So lagen beispielsweise die

durchschnittlichen Erstellungskosten im Jahr 1994 für den Inhalt einer 60 minütigen Präsentation zu Ausbildungszwecken nach einer Studie der FhG-IAO bei ca. 62.000 DM [Müll94].

Bisher wurde an uns von unterschiedlichster Seite aus immer wieder das bei anderen Systemen meist bestehende Problem der laufenden Änderung und Aktualisierung von Inhalten und Struktur genannt. Man hat hier einmalig Mittel eingesetzt und möchte nun nicht immer wieder alle paar Monate Dienstleistungsaufträge in Höhe der schon einmalig eingesetzten Mitteln vergeben müssen. Hier möchte der Betreiber selbst gerne gewisse Änderungen auf einfache Art durchführen können. Dieser Forderung haben wir von Anfang an Rechnung getragen. Dabei aber immer die Möglichkeiten so eingeschränkt, wie es die Style-Guides beschreiben. Dazu erhält der Betreiber z.B. eine vorgefertigte Anzahl generischer Seiten (Schablonen), welche er in den Kiosk integrieren und die darin vorgegebenen Felder füllen kann. Dabei kann das Layout dieser Seiten selbst nicht verändert werden, wodurch die Corporate Identity im System sichergestellt wird. Dieses Merkmal hat sich immer als äußerst positiv erwiesen.

Wir dachten anfangs, daß man mit der Realisierung eines Kiosk-Werkzeugs alle Anwenderwünsche befriedigen könnte, ohne nachträgliches Programmieren nötig zu machen. Dies war nicht so: fast jedes System erfordert einzelne Masken mit besonderer Logik oder den Anschluß an weitere meist bereits vorhandene Module. Deshalb wurde hier viel Wert auf ein offenes System bezüglich der Erweiterbarkeit gelegt.

Insgesamt haben die Anforderungen der eigentlichen Anwender immer Vorrang vor denen der Betreiber und diese wiederum Vorrang vor den Wünschen der Ersteller. Dies haben wir im Gegensatz zu allen uns bekannten anderen Systemen so konsequent umgesetzt.

8 Ausblick

Der Anfang der Forschung und Entwicklung auf diesem Bereich der Multimedia-Anwendungen und -Systeme in Heidelberg geht auf erste Konzeptionen und Realisierungen im Rahmen des EU RACE Projekts BANK zurück. Allerdings haben die dort realisierten Konzepte heute nur noch sehr wenig mit dem aktuellen und hier beschriebenen System gemeinsam. Diese initialen Entwicklungen haben damit eigentlich "nur" zu einer Sensibilisierung für diese Thematik und zum Aufbau unseres Wissen und Erfahrung wesentlich beigetragen. Heute befindet sich unser System im täglichen Einsatz, und es stehen neben der funktionalen Erweiterung auch Wartungs-, Installations-, und Schulungsmaßnahmen auf dem Programm.

Als Beiträge zur aktuellen Forschung sehen wir hier in erster Linie folgende Lösungen in unserem System:

- Eine neuartige Architektur, die eine unabhängige Entwicklung der einzelnen Komponenten durch eine Konzeption und Realisierung als Aufteilung in den Systemkern (vergleichbar mit einem Betriebssystemkern) und spezifische Erweiterungen (vergleichbar mit Gerätetreibern in Betriebssystemen) ermöglicht. Diese könnte als Referenzarchitektur solcher Systeme in Zukunft dienen.
- Der Konzeption und Realisierung des Abstrakten Kiosk, der eine höchst leistungsfähige Symbiose zwischen Objekt-Orientierung, endlichen Automaten (d.h. hier in Form von Zustands-Übergangs-Mechanismen "if ... then ... go to") und Modulstrukturen (d.h. beinhaltet als "is subpart") darstellt.

Das heute im Einsatz befindliche System beinhaltet die meisten der in diesem Beitrag beschriebenen Komponenten. Allerdings wurden Videokonferenz, Datex-J, die Abbildung des Abstrakten Kiosk auf eine relationale Datenbank und alle über die Funktionalität vom Online-

und Service-Editor hinausgehende Manipulationskomponenten bisher nur als Prototyp implementiert oder als Entwurf ausgearbeitet, die jetzt konkret umgesetzt werden.

Bei der gesamten Realisierung solcher Systeme sollte man immer sehr genau auch die Vorteile anderer bestehender Ansätze betrachten. Hier existieren heute schon viele Kiosk-Systeme mit einer großen Menge von sehr hilfreichen Funktionen. Als ein Beispiel sei hier der IBM Ultimedia Builder genannt; dort werden zum Beispiel außerordentlich gute "Dissolve-Methoden" angeboten, die von Designern oft auch intensiv verwendet werden. Deshalb arbeiten wir zur Zeit an einer Schnittstelle zwischen diesen Systemen, um die Vorteile anderer Systeme hier einbinden zu können. Im Hinblick auf Portabilität ist eine noch weiterreichende Entkopplung zwischen Engine und einem "User-Interface-Handler" wünschenswert, ebenso sind Schritte in Richtung offener interaktiver Dokumente mit zum Beispiel MHEG wichtig. Hier wird im Rahmen des Berkom GLASS-Projekts (Globally Accessible Services) mit unser Beteiligung eine MHEG-Engine für Multimedia-Dienstübergänge und interaktives Fernsehen erstellt.

Danksagung

Unser Dank gilt in erster Linie der Heidelberger MTG (Multimedia Technology Group), insbesondere haben hierzu Christian Halstrick, Michael Klischat, Harald Kühn, Ralf Müller und Peter Sander beigetragen. Außerdem haben Keith Hall, Wieland Holfelder, Ralf Guido Herrtwich, Markus Wieland sowie Christoph Wetekam und August Wegmann zum Erfolg dieses Projektes beigetragen.

Literaturverzeichnis

[GLASS94] *Berkom Glass System Specification*, DeTeBerkom, 1994.

[Hall94] K. Hall; *Databases for Kiosk Systems*, IBM European Networking Center, Technical Report 43.9421, Heidelberg, 1994.

[Holt94] B.Holtkamp (Hrsg.);Prof. Dr. H. Backhaus; *Anwendungserfahrungen: Akzeptanz, Kosten*; Symposium Integrierte Point-of-Information und Point-of-Sale Systeme, ISST Frauenhofer-Einrichtungen fuer Software- und Systemtechnik, Dortmund, 21.September 1994.

[EHVä93] J.L. Encarnação, W. Hübner, K.Väänänen; *Autorenwerkzeuge für multimediale Informationssysteme*, Informationstechnik und technische Informatik, no. 2, pp. 31-38, April 1993

[HoHe94] W. Hölfelder, D. Hehmann; *A Networked Multimedia Retrieval Management System for Distributed Kiosk Applications*, Proceedings of the 1994 IEEE International Conference on Multimedia Computing and Systems, Mai 1994

[ISO93] ISO/IEC; *Information Technology - Coded Representation of Multimedia and Hypermedia Information Objects (MHEG)*, Committee Draft ISO IEC 13522-1, Part I (Base Notation ASN.1), 1993.

[KANe93] W. Klas, K. Aberer, E. Neuhold; *Object-Oriented Modelling for Hypermedia Systems Using the VODAK Model Language*, Object-Oriented Database Management Systems, NATO ASI Series, Springer Verlag Berlin Heidelberg, November 1993.

[Krie94] M. Krieger; *Ein objektorientiertes System zur interaktiven Generierung von verteilten multimedialen Kiosksystemen*, Diplomarbeit im Fach Informatik, Inst. für Telematik Universität Karlsruhe, März 1994.

[LuOe93] N. Luttenberger, R. Oechsle; *RACE BANK - a Multimedia Broadband Cooperation Project in the Banking Business Sector*, Euro-ARCH 93, Berlin: Springer Verlag, 1993.

[Müll94] B. Müller; *Der Trend heißt Vielfalt*; Screen Multimedia, September 1994, ss. 122-123.

[Niel90] J. Nielsen; *HyperText & HyperMedia*, Academic Press Inc., 1990.

[Price93] R. Price; *MHEG: An Introduction to the future International Standard for Hypermedia Object Interchange*, ACM Multimedia 93, Conference Proceedings, Anaheim CA, acm press, Aug. 1993, pp. 121-128.

[Rako93] T.C. Rakow et al.; *Einsatz von objektorientierten Datenbanksystemen für Multimedia-Anwendungen*, Informationstechnik und technische Informatik, no. 2, April 1993, pp. 4-17.

[Stei93] R. Steinmetz; *Multimedia-Technologie: Einführung und Grundlagen*, Springer Verlag Berlin Heidelberg, Sept. 1993.

[Stei94] Ralf Steinmetz; *Data Compression in Multimedia Computing: Principles and Techniques*; acm/Springer 'Multimedia Systems', Vol.1, No.4, pp.166-172, February 1994, und Ralf Steinmetz; *Data Compression in Multimedia Computing: Standards and Systems*; acm/Springer 'Multimedia Systems', Vol.1, No.5, March 1994.