Inhaltsverzeichnis

Kapitel 1

Einleitung

In diesem Kapitel werden zunächst in Abschnitt 1.1 die Motivation und der Kontext sowie in Abschnitt 1.2 die Aufgabenstellung dieser Diplomarbeit erläutert. Der Abschnitt 1.3 beinhaltet einen Überblick über den Aufbau der Arbeit.

1.1 Motivation und Kontext

Informationssysteme in Krankenhäusern stellen einen wesentlichen Faktor für die Qualität der Patientenversorgung und für die Qualität der Verwaltung und Leitung der Krankenhäuser dar. Eine zentrale Stellung beim Einsatz von medizinischen Informationssystemen nimmt die Erfassung, Verwaltung und Auswertung der Daten bei der Behandlung der Patienten ein. Traditionell werden Patientenakten in Papierform für die Aufzeichnungen der Patientendaten eingesetzt. Die Inhalte der Patientenakten sind teilweise krankenhaus- oder abteilungsspezifisch und umfassen unterschiedliche Dokumentarten wie bspw. informelle Texte (Arztbriefe, Entlassungsberichte), standardisierte Formulare, Diagramme und Bilddaten.

Verbunden mit der Einführung von rechnergestützen Krankenhausinformationssystemen in den siebziger Jahren konnten Teile der Patientenakte DV-technisch unterstützt werden. Elektronische Patientenakten bieten gegenüber traditionellen Patientenakten eine Vielzahl von Vorteilen. Hierzu zählen u.a. gleichzeitiger Zugriff von örtlich verteilten Arbeitsplätzen, höherer Aktualität der Daten, verbesserte Möglichkeiten zur Suche und Darstellung, vereinfachte Archivierung, Verringerung von Datenverlusten und Vereinfachung der Auswertung.

In den letzten Jahren ermöglichte der Einsatz multimedialer Techniken weitere Möglichkeiten zur effizienten Aufzeichnung, zur Repräsentation und zur Verwaltung der Patientendaten. Eine multimediale Patientenakte erlaubt die rechnergestützte Bearbeitung unterschiedlicher Datentypen wie Texte, Grafiken, Filme oder Tonaufnahmen.

1.2 Aufgabenstellung

Die Aufgabe dieser Diplomarbeit besteht in der Entwicklung einer hypermedialen Patientenakte, um bestehende Ansätze zu verbessern und zu erweitern. Im folgenden werden die Teilaufgaben bei der Analyse, dem Entwurf und der prototypische Realisierung einer hypermedialen Patientenakte näher festgelegt.

Im Rahmen der Anforderungsanalyse ist zunächst eine Untersuchung der traditionellen und der rechnergestützten Patientenakte notwendig. Hierbei kann auf die vorliegenden Ergebnisse einer Projektgruppe zurückgegriffen werden. Als Ergebnis der Anforderungsanalyse soll ein Katalog von fachlichen und DV-technischen Anforderungen vorliegen.

Die Patientenakte soll als hypermediales System konstruiert werden. Somit weist die Patientenakte als wesentliche Eigenschaft eine nicht-lineare Informationsverkettung auf. Die Struktur der Patientenakte entspricht einem Graphen bestehend aus Knoten und Kanten. Die Knoten sind die eigentlichen multimedialen Informationseinheiten, die Kanten stellen den Bezug zwischen verschiedenen Informationseinheiten her. Sie werden üblicherweise als Verweis oder Link bezeichnet und ermöglichen die Navigation in dem Hypermedia-System.

Im Verlauf der Arbeit sollen einzelne Aspekte der hypermedialen Patientenakte näher untersucht werden. Dies umfaßt Fragen zur Integration von Werkzeugen (insbesondere der Integration bestehender, weitverbreiteter Werkzeuge), Aspekte des Datenschutzes und der Datensicherheit, die verteilte Speicherung von Daten und Fragen hinsichtlich der Aktualität der Daten. Die verschiedenen Aspekte sollen spätestens nach der Anforderungsanalyse festgelegt und konkretisiert werden.

Zum Entwurf und zur prototypischen Realisierung der hypermedialen Patientenakte soll der Ansatz *Object Oriented Hypermedia Design Methodology* (OOHDM) von Schwabe, Rossi und Barbosa eingesetzt und evaluiert werden ([SRB95a]). OOHDM ist eine Vorgehensweise zur Konstruktion von Hypermedia-Applikationen, die auf einem vierstufigen Prozeß basiert.

Im ersten Schritt wird ein konzeptionelles Problembereichsmodell erstellt. Hierbei werden im wesentlichen Modellierungs- und Abstraktionstechniken eingesetzt, wie sie bspw. von OMT propagiert werden. Der zweite Schritt dient der Spezifikation von navigationellen Sichten auf das Objektmodell. Hier werden unter der Berücksichtigung von Benutzerprofilen verschiedene Systemsichten modelliert. Dies geschieht mit Hilfe der Primitive Knoten, Verknüpfungen und anderen, indem das Konzeptmodell auf die navigationellen Sichten abgebildet wird. Der dritte Schritt besteht in der Spezifikation von abstrakten Schnittstellen, über die das System von den Benutzern wahrgenommen und gesteuert wird. Der letzte Schritt umfaßt die Implementation der zuvor spezifizierten Modelle auf der Basis einer verfügbaren Hypermedia-Plattform wie Toolbook, Microcosm oder dem World Wide Web.

Die gesamte Entwicklung der hypermedialen Patientenakte, in erster Linie die Anforderungsanalyse, die Bereitstellung von Datenmaterial sowie die Bewertung des entwickelten Systems erfolgt in Zusammenarbeit mit dem Mathias-Spital in Rheine.

1.3 Überblick

Dieser Abschnitt beschreibt den Aufbau der vorliegenden Diplomarbeit.

Kapitel 1. In diesem Kapitel werden zunächst in Abschnitt 1.1 die Motivation und der Kontext sowie in Abschnitt 1.2 die Aufgabenstellung dieser Diplomarbeit erläutert. Der Abschnitt 1.3 beinhaltet einen Überblick über den Aufbau der Arbeit.

Kapitel 2. In diesem Kapitel werden zunächst in Abschnitt 2.1 Aspekte der traditionell auf Papierbasis verwalteten und computerunterstützten Patientenakte vorgestellt. In Abschnitt 2.2 werden kurz die wesentlichen Charakteristika von Hypertext, Multimedia und Hypermedia präsentiert und zwei Modelle, das *Dexter-Hypertext-Modell* [HS94] zur Beschreibung von Hypertext- und das *Amsterdam Hypermedia Model* [HBvR94] zur Beschreibung von Hypermediasystemen vorgestellt. In Abschnitt 2.3 werden Gründe für die gewählte Form der Spezifikation der Patientenakte als Hypermediasystem erläutert und mögliche dadurch zu erreichende Ziele präsentiert. Der letzte Abschnitt 2.4 dieses Kapitels befaßt sich mit Hypermedia-Entwicklungsmethoden. Es werden von Balasubramanian entwickelte Richtlinien zum Entwurf von Benutzerschnittstellen von Hypermediasystemen [Bal94], die Methode *RMM* [ISB95] und die im Vorfeld der Diplomarbeit als zu verwendende ausgewählte Methode *OOHDM* [SRB95a] vorgestellt und verglichen.

Kapitel 3. In diesem Kapitel werden die im Rahmen dieser Diplomarbeit ermittelten fachlichen und DV-technischen Anforderungen vorgestellt, die das zu realisierende System, die hypermediale Patientenakte, erfüllen muß.

Kapitel 4. In diesem Kapitel werden in Abschnitt 4.1 zunächst die Grundlagen des durchgeführten Entwicklungsprozesses vorgestellt. Abschnitt 4.2 behandelt das entworfene Konzeptmodell. Hier werden die Subsysteme dieses Modells präsentiert und die Klasse `Patient` sowie die Beziehung `AnforderungPatient` beispielhaft vorgestellt. Der nächste Abschnitt 4.3 befaßt sich mit dem Navigationsmodell. Dieses wird allgemein anhand einer Beschreibung der spezifizierten Sichten und speziell durch konkrete Beispiele präsentiert. In Abschnitt 4.4 wird die verwendete Notation zur Beschreibung der Benutzerschnittstellentransformationen vorgestellt und anhand zweier Beispiele erläutert. Der letzte Abschnitt 4.5 behandelt Aspekte der Umsetzung der bisher konstruierten Modelle in eine Implementation. Zunächst wird die gewählte Vorgehensweise, eine Implementierung auf Basis von Java-Applets im Intranet-Rahmen beschrieben, dann wird auf Aspekte der Datenspeicherung durch Nutzung der JDBC-Klassen eingegangen und zuletzt wird der Aspekt der Integration von externen Anwendungen betrachtet.

Kapitel 5. In diesem Kapitel wird eine Bewertung des spezifizierten Systems und des Prototypen vorgenommen. Dies geschieht durch Überprüfung der Anforderungen und Vergleich derselben mit den erzielten Ergebnissen. Außerdem wird OOHDM im praktischen Einsatz bewertet.

Kapitel 6. Dieses letzte Kapitel gibt zunächst in Abschnitt 6.1 eine Zusammenfassung dieser Diplomarbeit wieder. Anschließend wird in Abschnitt 6.2 ein Ausblick auf mögliche Fortsetzungen dieser Arbeit vorgenommen.

Die vorliegende Diplomarbeit umfaßt zwei Teile. Teil I beinhaltet die schriftliche Ausarbeitung und Teil II beinhaltet die Objekt- und Schnittstellenbeschreibungen sowie den kommentierten Quellcode des Prototypen, letzteren auf einer Diskette. Der Teil II wurde am Lehrstuhl X des Fachbereichs Informatik der Universität Dortmund hinterlegt und kann dort eingesehen werden.

Im Verlauf dieser Diplomarbeit konnten die Bereiche des Datenschutzes und der Datensicherheit im Rahmen medizinischer Software aufgrund ihrer Größe und Komplexität nicht näher betrachtet werden. Weiterführende Literatur findet sich auf der WWW-Seite *Sicherheitsempfehlungen zum Internet-Anschluß von Krankenhäusern* [PS96] der Deutschen Gesellschaft für Medizinische Informatik, Biometrie und Epidemiologie.

Kapitel 2

Grundlagen

In diesem Kapitel werden zunächst in Abschnitt 2.1 Aspekte der traditionell auf Papierbasis verwalteten und der computerunterstützten Patientenakte vorgestellt. Es werden Vor- und Nachteile der papierbasierten Patientenakte präsentiert, sowie Eigenschaften, die eine computerunterstützte Patientenakte haben sollte, vorgestellt. In Abschnitt 2.2 werden kurz die wesentlichen Charakteristika von Hypertext, Multimedia und Hypermedia präsentiert und zwei Modelle, das *Dexter-Hypertext-Modell* [HS94] zur Beschreibung von Hypertext- und das *Amsterdam Hypermedia Model* [HBvR94] zur Beschreibung von Hypermediasystemen vorgestellt. In Abschnitt 2.3 werden Gründe für die gewählte Form der Spezifikation der Patientenakte als Hypermediasystem erläutert und mögliche dadurch zu erreichende Ziele präsentiert. Der letzte Abschnitt 2.4 dieses Kapitels befaßt sich mit Hypermedia-Entwicklungsmethoden. Es werden von Balasubramanian entwickelte Richtlinien zum Entwurf von Benutzerschnittstellen von Hypermediasystemen [Bal94], die Methode *RMM* [ISB95] und die in dieser Diplomarbeit eingesetzte Methode *OOHDM* [SRB95a] vorgestellt und verglichen.

2.1 Patientenakte

In diesem Abschnitt werden Aspekte der traditionellen papierbasierten und der computerunterstützten Patientenakte vorgestellt.

2.1.1 Die traditionelle Patientenakte

Im Krankenhaus werden eine Vielzahl von diagnostischen, therapeutischen, pflegerischen und verwaltungstechnischen Leistungen erbracht, die in erster Linie den Gesundheitszustand der Patienten positiv beeinflussen sollen. Im Rahmen dieser Tätigkeiten fallen eine Vielzahl an Informationen an, die geeignet dokumentiert, verarbeitet und archiviert werden müssen. Traditionell wurde ein Großteil dieser Information mit Hilfe von Patientenakten in Papierform verwaltet.

Die genaue Zusammensetzung einer solchen Akte variiert sehr stark von Krankenhaus zu Krankenhaus, da jedes eigene Formulare verwendet. Im allgemeinen ist diese Dokumentensammlung heute eine Mischung aus informellen Texten, wie z.B. Aufnahmebögen und Entlassungsberichte, die von Hand geschrieben werden, aus teilweise computerunterstützten und genormten Formularen wie z.B. Leistungsanforderungen oder Anästhesiebögen, auf denen Felder zum Ankreuzen vorgegeben sind, und aus Computerausdrucken von Meßwerten.

Mit diesem System sind eine Reihe von Vor- und Nachteilen verbunden. Eine umfassende Behandlung dieser Aspekte findet der interessierte Leser im Buch von Ball und Collen [BC92]. Im folgenden werden in Anlehnung an McHugh [McH92] die wesentlichen Vorteile aufgeführt:

- *Es ist ein etabliertes System.* Das Papiersystem wird schon seit geraumer Zeit in allen Krankenhäusern verwendet. Das Personal ist mit den nötigen Bearbeitungsrichtlinien vertraut und eingearbeitet oder kann es in relativ kurzer Zeit werden.

- *Es ist für Dateneingaben leicht zu benutzen.* Die dafür nötigen Fähigkeiten sind Lesen und Schreiben, welche jeder Krankenhausangestellte besitzt.

- *Es kann leicht an unterschiedliche Bedürfnisse angepaßt werden bzw. beliebig erweitert werden.* Wenn sich Prozeduren und Formalitäten ändern, wird einfach ein anderes Formular verwendet und in die Akte eingefügt. Es lassen sich beliebig viele Formulare hinzufügen oder entfernen.

- *Es ist flexibel.* Einer handschriftlichen Notiz kann schnell noch etwas hinzugefügt werden, oder auf einen Vordruck können ohne großen Aufwand noch Anmerkungen niedergeschrieben werden.

- *Es bietet relativ guten Schutz gegen unautorisierten Zugriff.* Um die Patientenakte zu lesen oder sie zu verändern, muß man sie physisch vorliegen haben.

Dem gegenüber stehen die folgenden Nachteile:

- *Manuelle Dateneingabe durch Schreiben ist sehr zeitaufwendig.* Die Zeit des Pflegepersonals ist sehr knapp bemessen. Sehr viel Zeit, die besser für direkte Patientenpflege verwendet werden könnte, geht verloren durch:

  - den Zwang zu sorgfältiger schriftlicher Berichtdokumentation
  - die Verpflichtung zu redundanter Datenhaltung
  - Ineffizienzen bei Dateneingabeformaten
  - eine Beschränkung auf einzelnen Zugriff auf die Akte
  - die Notwendigkeit, Daten von einem Formular auf ein anderes zu übertragen
  - kaum vorhandene Unterstützung für das Gedächtnis und die Arbeitsorganisation

  Studien haben ergeben, daß Pflegepersonal bis zu 40 Prozent seiner Zeit mit Informationsbeschaffung und -eingabe verbringt [JG66, Ric70]. Teilweise kommt es aufgrund von fehlenden oder nicht auffindbaren Dokumenten zu unerwünschten Effekten wie die unnötige Durchführung von Untersuchungen oder das Treffen von Fehldiagnosen. Gleichzeitig ist eine Formularsammlung von nicht aufeinander abgestimmten Blättern, in vielen Fällen in hohem Maße redundant sowie teilweise schlecht lesbar, unvollständig oder unpräzise. Es gibt keine einheitliche Datendichte. Diese Faktoren erschweren dem Pflegepersonal den Umgang mit der Patientenakte und mindern damit die Behandlungsqualität.

- *Das System ist sperrig und somit schwer zu transportieren und teuer zu lagern und zu unterhalten.* Dadurch müssen Hospitäler große Räume als Archive zur Verfügung stellen, die angemessen temperiert und nicht feucht sind. Gleichzeitig ist es zeitaufwendig, alte Akten, die wieder benötigt werden, zu lokalisieren und zu holen.

- *Der Entwurf und Druck von Papierformularen ist teuer.* Jede Änderung im Design eines Formulars erfordert einen vollständigen Neuentwurf und -druck, was große Kosten verursacht.

- *Datensuche und -sammlung geschieht von Hand und ist damit aufwendig.* Die Durchsicht von Tausenden von Tabellen, z.B. zu statistischen Zwecken, und deren Zusammenfassung und Auswertung ist von Hand extrem aufwendig.

2.1.2 Die computerbasierte Patientenakte

In den sechziger Jahren wurde aufgrund der Entwicklung von leistungsfähigen Computern verstärkt die Möglichkeit betrachtet, Rechner zur Unterstützung der Verwaltung der Patientenakten in den Krankenhäusern heranzuziehen. Eines der ersten Systeme war PROMIS im Jahre 1969, welches eine computergespeicherte Version einer problemorientierten Patientenakte war [Wee69, FSLS80].

Es dauerte allerdings bis in die späten 70er und frühen 80er Jahre, bis die ersten wirklich computerbasierten Patientenakten in großen Krankenhäusern auftauchten. Beispiele hierfür sind nach Pryor [Pry92] das Health Evaluation through Logical Processing-System (HELP), das Regenstrief Medical Record System (RMRS) und The Medical Record (TMR).

Ein wesentlicher Nachteil dieser Systeme basierte auf der Prämisse einer zentralen Patientenversorgung, d.h., daß diese Krankenhäuser darauf ausgelegt waren, alle gesundheitlichen Probleme eines Patienten zu versorgen. Das bedeutete, daß man nicht in Erwägung zog, daß ein Patient auch gelegentlich woanders behandelt werden könnte. Mit der Dezentralisierung und Spezialisierung im Gesundheitswesen sowie einer generellen Verbesserung des Transportsystems wurde die Behandlung eines Patienten an mehreren Orten jedoch zum Regelfall.

Dies führte zur Situation, daß

- einerseits dieselben Patienteninformationen an verschiedenen Orten in verschiedenen Formaten vorlagen: *Redundanz*,
- andererseits spezielle Daten nur bei einem Arzt/Krankenhaus zu finden waren, ohne einfache Möglichkeit, diese an andere Institutionen weiterzuleiten: *verlorene Daten*.

Heutige Systeme wie pro7 [GS97] und OHS [ROK95] versuchen durch eine offene Architektur und Unterstützung von weltweiten Kommunikationsstandards wie HL7 [Pac94] die obengenannten Probleme zu lösen.

2.1.2.1 Eigenschaften

Die genannten Aspekte, zusammen mit anderen Problemen wie Datenintegritätsverletzungen und Datenunzugänglichkeit, machten deutlich, daß Qualitätsanforderungen erforderlich waren, um einer computerbasierten Patientenakte adäquate Grundlagen zu verleihen. Im folgenden werden auf der Grundlage verschiedener Beiträge [McD92, Mar92, McH92, Ull92, Buc92, Sen92, Dav92, AW92] die wesentlichen fachlichen und technischen Eigenschaften, die bei der Formulierung geeigneter Qualitätsanforderungen berücksichtigt werden müssen, stichwortmäßig aufgelistet:

Fachliche Eigenschaften:

- Funktion als finanzielle und legale Unterlage
- Funktion als Terminplaner und Informationsaustauschobjekt
- Grundlage für Forschung und Statistik durch Anzeige von Trends und Mustern und Möglichkeit zu direkter, intelligenter Analyse
- Unterstützung der Pflege und Qualität
- Steigerung der Produktivität des Personals und Reduzierung administrativer Kosten
- Vertraulichkeit der Patientendaten
- Bereitstellung von Patienteninformationen zur richtigen Zeit am richtigen Ort im richtigen Format (lesbar und gut organisiert)
- Möglichkeit zu eigener Zusammenstellung von Informationen

- Bereitstellung einer Benutzerhilfe
- Bereitstellung eines breiten Informationsspektrums
- Einfache Bedienung und schnelle Reaktion - Benutzerfreundlichkeit
- Anpassungsfähige Dateneingabe- und -ausgabemöglichkeiten
- vollständige Dokumentation des Pflegeprozesses
- Verwendung existierender Terminologien

Technische Eigenschaften:

- Dezentralisierung/Verteiltheit im kleinen (Krankenhaus) und im großen (verschiedene Institutionen)
- Zusammensetzung aus unterschiedlichen Datentypen (Multimedia)
- Nutzung von Datenbankstrukturen
- Teil eines integrierten Patientenpflegesystems
- Mehrbenutzerzugriff
- Plattformunabhängigkeit
- Adäquate Datenspeicherung (aktive Patienten online, inaktive Patienten offline)
- Backupmöglichkeiten
- Einbeziehung zukünftiger Entwicklungen
- Ermöglichung von Kommunikation nach außen
- Direkte Speicherung von Überwachungsdaten
- Passwortsicherheit
- Zugangsbeschränkungen (Sichten)

Qualitätsanforderungen, die diese Eigenschaften abdecken, werden in Kapitel 3 vorgestellt.

2.2 Hypertext, Multimedia und Hypermedia

Dieser Abschnitt befaßt sich mit grundlegenden Aspekten von Hypertext, Multimedia und Hypermedia, die im folgenden vorgestellt werden.

2.2.1 Hypertext

Hypertextsysteme treten als eine neue Art von komplexen Informationsverwaltungssystemen immer mehr in den Vordergrund. Sie stellen dem Benutzer im Gegensatz zur traditionell sequentiellen Informationsdarstellung eine neue nichtsequentielle Methode zur Verfügung.

Als Vater des Hypertexts gilt Vannevar Bush, der 1945 ein System namens Memex postulierte, daß in der Lage sein sollte, Bücher und Aufzeichnungen zu speichern und zu verbinden [Bus45]. T.Nelson prägte 1965 den Begriff Hypertext, mit dem er einen Körper bezeichnete, welcher aus auf komplexe Weise verbundenem Datenmaterial besteht, so daß er nicht einfach auf Papier darstellbar ist [Nel65]. Seitdem schritt die Entwicklung stetig voran. Über Systeme wie Augment/NLS [Eng63] oder Xanadu [Nel80] bis hin zu modernen Anwendungen, die sich mittlerweile fast alle zumindest im Bereich der Online-Hilfe Hypertext-Eigenschaften zunutze machen, reicht die Spanne von Hypertextsystemen. Ihnen allen gemein ist die Basiseigenschaft:

Hypertext besteht aus Textknoten und Verknüpfungen.

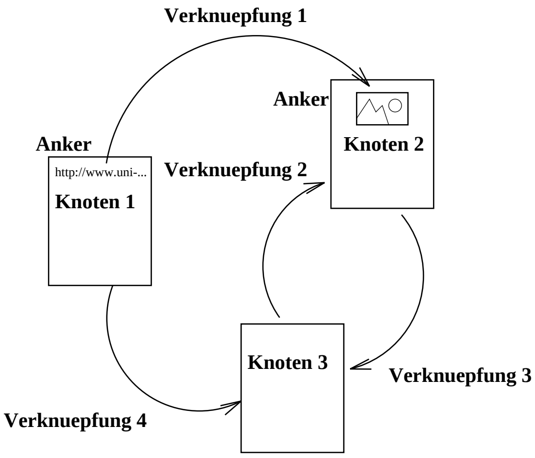


Abbildung 2.1: Knoten und Verknüpfungen.

2.2.1.1 Bestandteile

Hypertext hat eine Architektur. Die grundlegenden Elemente dieser Architektur sind *Knoten* und *Verknüpfungen*. Knoten repräsentieren Konzepte oder einzelne Ideen und Verknüpfungen Beziehungen zwischen diesen. Anker sind die Anfangs- und Endpunkte einer Verknüpfung in einem Knoten. Verknüpfungen verbinden verwandte Konzepte oder Knoten. Sie können bidirektional sein und somit Navigation in beide Richtungen unterstützen.

2.2.1.2 Dexter-Modell

Es gibt verschiedene Möglichkeiten, ein Hypertextsystem zu beschreiben. Das am häufigsten verwendete ist das von Halasz und Schwartz beschriebene Dexter-Hypertext-Referenzmodell, welches die wichtigen Abstraktionen eines Hypertextsystems spezifiziert [HS94]. Das Ziel dieses Modells ist es, eine systematische Basis zum Vergleich verschiedener Systeme zu liefern sowie Austausch- und Interoperabilitätsstandards zu entwickeln.

Das Dexter-Modell unterteilt ein Hypertextsystem in drei Schichten:

Laufzeitschicht. Hier werden die Hypertextpräsentation und die Dynamik der Benutzerinteraktion festgelegt. Zwischen der Laufzeitschicht und der Speicherschicht liegen als Schnittstelle Präsentationsspezifikationen, die von diesem Modell allerdings nicht konkret behandelt werden.

Speicherschicht. Hierauf liegt der Schwerpunkt des Dexter-Modells. Die Schicht modelliert eine Datenbank, die aus einer Hierarchie von Komponenten besteht. Diese Komponenten können Atome, Verknüpfungen oder zusammengesetzte Komponenten sein. Jede Verknüpfung hat Anker, die in jeweils einer Komponente enthalten sind und die als Ursprung oder Ziel benutzt werden können. Ein Anker ist innerhalb einer Komponente eindeutig spezifiziert.

In dieser Schicht wird keine Struktur innerhalb einer Komponente spezifiziert; der Schwerpunkt liegt hauptsächlich auf dem Mechanismus, durch den Komponenten und Verknüpfungen zu Hypertextnetzwerken geformt werden.

Komponenteninhaltsschicht. Diese Schicht beschreibt Inhalt und Struktur von Komponenten des Hypertextnetzwerks. Das Dexter-Modell behandelt diese Schicht nicht konkret, sondern betrachtet sie als außerhalb des Hypertextbeschreibungsmodells liegend. Eine Schnittstelle namens Verankerung, die zwischen Speicherschicht und Komponenteninhaltsschicht liegt, beschreibt den Adressierungsmechanismus innerhalb individueller Komponenten.

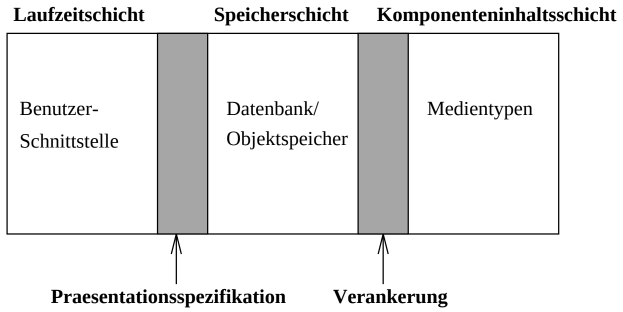


Abbildung 2.2: Das Dexter-Modell

2.2.1.3 Navigation

Hypertext ist ein Netzwerk von Informationsknoten, die miteinander verbunden sind, und die nichtsequentiell abgeschritten werden können. Deshalb muß ein Benutzer eines solchen Systems in die Lage versetzt werden, innerhalb der durch das System gegebenen logischen Grenzen des Informationsraums global und lokal frei zu navigieren. Dies bedeutet, daß der Benutzer sowohl die globale Gesamtorganisation des Systems als auch die unmittelbare lokale Umgebung seines gegenwärtigen Knotens übersichtlich präsentiert bekommen sollte. Dies wird ihm durch Werkzeuge ermöglicht.

2.2.1.4 Mögliche Anwendungen

Der Hypertextansatz eröffnet neue Möglichkeiten, komplexe Informationsmanagementsysteme zu entwickeln. Enzyklopädien, Lexika und Handbücher bieten sich für die Umsetzung in Hypertext an (siehe beispielsweise [Fri88b]). Die meisten der heute gebräuchlichen Anwendungen haben eine hypertextlich strukturierte Online-Hilfe. Hypermediasysteme sind ebenfalls dazu geeignet, Dokumentationen an Benutzerbedürfnisse und -perspektiven anzupassen [MT90]. Lernsysteme sind besonders als Hypertext geeignet, da diese Struktur Lernen durch Erforschen fördert [Fri88a, YSGM88]. Kreative Tätigkeiten werden ebenfalls durch die Möglichkeit, Ideen als Hypertext zu speichern und zu präsentieren, gefördert, da die entsprechenden Informationen oft aus Informationseinheiten bestehen, die durch Assoziationen verknüpft sind [Hal88, SHT89]. Weiterhin wird Gruppenarbeit unterstützt [IT90]. Es gibt noch viele weitere Anwendungsgebiete, die aber zu zahlreich sind, um hier alle explizit aufgeführt zu werden. Aus diesem Grund folgen noch einige exemplarische Verweise für unterschiedliche Bereiche.

- Entscheidungsunterstützungssysteme: [Bie93]
- Softwareentwicklung: [BMG+89, BR87]
- Simulation und Modellierung: [SL89]
- Gesetz und Rechtssprechung: [YW89]
- World Wide Web : [BL92]

2.2.2 Multimedia

Schulmeister [Sch96] definiert Multimedia als „interaktive Form des Umgangs mit symbolischem Wissen in einer computergestützten Interaktion “. Dies umfaßt die Erfüllung folgender Kriterien:

- Die Daten von unterschiedlichen Medien treten integriert auf.

- Die Daten werden vom Rechner verarbeitet und manipuliert.
- Für den Benutzer ist Multimedia ein multisensorischer Eindruck, eine multiple Repräsentation von interpretierbaren Informationen.
- Entscheidend für die Unterscheidung von sequentiellen multiplen Medien und Multimedia ist die Interaktion des Benutzers mit der Software.
- Die von Multimedia präsentierten Informationen sind symbolische Ausdrucksformen, symbolisches Wissen, das seinen Wert erst im interpretierenden Zugriff des Benutzers gewinnt; Multimedia-Informationen sind Anlässe für eigene kognitive Konstruktionen.

In diesem Abschnitt werden die geläufigsten Medientypen, die i.a. durch den Begriff Multimedia zusammengefaßt werden, kurz vorgestellt.

2.2.2.1 Text

Dies ist der traditionellste und geläufigste Medientyp. Text ist immer noch die verbreitetste Methode, Informationen in einem Computersystem darzustellen. Jeder moderne Computer ist in der Lage, ohne großen Aufwand Text wiederzugeben.

2.2.2.2 Bilder, Grafiken und Animationen

In der heutigen Zeit sind Computer, die Grafikausgaben unterstützen, die Norm. Deshalb verwenden die meisten modernen Programme Bilder und Grafiken. Der Vorteil besteht in einer großen Aussagekraft, der Nachteil ist ein großer Speicherplatzbedarf. Ein anderes Problem sind die vielen existierenden Formate (GIF, JPEG, BMP, EPS, etc.). Animationen sind ein Zwischenschritt zwischen stehenden Bildern und Videosequenzen. Der Vorteil von Animationen gegenüber Video ist, daß diese i.a. leichter und billiger zu produzieren sind sowie eine Abstraktion eines Sachverhalts realisieren. Durch letzteres wird eine unübersichtliche Detailfülle bei der Darstellung verhindert.

2.2.2.3 Video

Eine Videosequenz besteht lediglich aus einer Serie von Standbildern, die mit einer Geschwindigkeit von 30 Bildern pro Sekunde dargestellt werden müssen. Dies stellt hohe Anforderungen an die Hardware des Computers. Ein weiteres Problem ist die Existenz von zwei inkompatiblen Formaten.

Analog: Analoges Video entspricht dem Video im Fernsehen. Es besteht aus durchgehenden Farblinien, die angezeigt werden und wieder verschwinden, wenn sie durch neue ersetzt werden.

Digital: Digitales Video besteht aus einzelnen Punkten, die, jeweils zu einem vollen Bildschirm zusammengefaßt, auf einem digitalen Monitor dargestellt werden.

Um Videodaten in einem vernünftigen Rahmen speichern zu können, müssen diese komprimiert werden. Auf diese Techniken kann hier aus Platzgründen nicht näher eingegangen werden.

2.2.2.4 Audio

Audio- gibt es genau wie Videodaten in analoger und digitaler Form. Schallplatten und Kassetten speichern Audiodaten in einem analogen Format, CD-Audios und Digitale Audio-Bänder in einem digitalen.

Computer können digitale Audiodaten auf Weisen verarbeiten, die mit analogen Signalen nicht durchführbar sind. Allerdings verbraucht digitales Audio fast genausoviel Speicherplatz wie Video. Das Gerät, welches analoge Audiodaten in digitale verwandelt, nennt sich Digitizer (oder Sampler).

Statt reelle Geräusche zu digitalisieren, ist es möglich, mit spezialisierten Chips Audio direkt zu generieren. Hierfür gibt es im Gegensatz zu Bildern und Video einen einzelnen weltweiten Standard - MIDI. Damit kann Audio auf einem Synthesizer erzeugt und auf einem Computer verarbeitet und gespeichert werden. Ähnlich wie Videodaten sollten Audiodaten komprimiert werden, um den Speicherplatzverbrauch in einem vernünftigen Rahmen zu halten.

2.2.2.5 Timing und Synchronisation

Eine weitere Anforderung an Multimediadarstellungen auf dem Computer ist die Fähigkeit, kontinuierliche synchronisierte Medien in Realzeit darzustellen. Ein Beispiel hierfür ist eine Nachrichtensendung, bestehend aus einem Videofilm des Sprechers und der Audioausgabe seiner Stimme.

Um solche Zeitabhängigkeiten zu modellieren, gibt es verschiedene Möglichkeiten. Sie können auf Momenten oder Intervallen basieren. Ein Moment ist ein Zeitpunkt mit der Länge 0, wie z.B. 11,00 Uhr. Ein Zeitintervall wird durch zwei Momente begrenzt, und hat die Länge a zwischen diesen beiden Momenten. Zwei Intervalle können auf verschiedene Weise miteinander in Beziehung stehen. Es ist in der Informationstechnologie immer noch ein schwieriges Problem, diese Beziehungen effizient zu behandeln.

Es gibt zur Modellierung von zeitbasierten Medien sowohl graph- als auch skriptbasierte Repräsentationen. Der Vorteil von graphbasierten Repräsentationen liegt in der bildlichen Darstellung der Synchronisation. Petrinetze [Rei82] sind ein Beispiel hierfür.

2.2.2.6 Zusammenfassung

Multimediadarstellung auf dem Computer stellt ausgiebige Leistungsanforderungen an die Soft- und Hardware. Im Gegensatz zur traditionellen Grafikbearbeitung beinhaltet eine einzelne Multimediapräsentation mehrere verschiedene Datenflüsse, von denen jeder einzelne spezielle Anforderungen an Verarbeitung und Präsentation hat. Das System muß in der Lage sein, gespeicherte Daten zu dekomprimieren und aufgezeichnete Daten zu komprimieren, um mit den massiven Datenmengen fertigzuwerden. Information muß synchronisiert werden. Dieses erfordert eine sorgfältige Koordinierung der erforderlichen Verarbeitungsprozesse.

2.2.3 Hypermedia

In diesem Kapitel wird der Unterschied zwischen Hypertext und Hypermedia erläutert. Es werden spezielle Probleme angesprochen und ein Modell, das Amsterdam Hypermedia Modell, vorgestellt.

2.2.3.1 Hintergrund

Der Begriff Hypermedia verbindet die verschiedenen, unter dem Begriff Multimedia eingeordneten Datentypen mit den inhaltsbasierten Informationsassoziationsmöglichkeiten, die durch Hypertext angeboten werden. Es reicht nicht aus, einer Multimediapräsentation eine hypertextähnliche Navigationsstruktur aufzusetzen, um ein Hypermediasystem zu erschaffen. Die Problematik liegt in den Unterschieden zwischen statischen und dynamischen Daten. Die Hauptprobleme bei der Konstruktion von Hypermediaapplikationen lassen sich wie folgt zusammenfassen [HBvR94, Bal94]

Navigation in Hypermedia. Auch in einem Hypermediaraum besteht das Problem des Lost--In-Hyperspace. Aus diesem Grund muß der Anwender mit Navigationshilfen wie Karten unterstützt werden.

Ablaufkontrolle. Der Benutzer einer Hypermediaanwendung sollte in der Lage sein, den Ablauf einer dynamischen Informationseinheit zu steuern.

Verknüpfungen in Hypermedia. Problematisch sind Sprünge in eine dynamische Informationseinheit. Hierbei gibt es zwei Möglichkeiten, Verknüpfungen zwischen dynamischen Objekten zu ermöglichen: Behandlung einer Sequenz als einen diskreten Knoten, wobei ein Sprung in diese Informationseinheit nicht möglich ist, oder Behandlung als Satz von Knoten, wobei ein Sprung in diese Einheit zu jedem Teilknoten der Sequenz geschehen kann.

Effiziente Hypermedia-Viewer. Ein Hypermedia-Viewer sollte es dem Benutzer ermöglichen, Geschwindigkeit und Anordnung der Präsentation selbst zu wählen. Weiterhin sollte das System dem Benutzer eine angemessene Reaktionszeit liefern. Als Grundeigenschaft sollte jeder Viewer in der Lage sein, dem Benutzer jederzeit einen Ausstieg zu ermöglichen.

Produktionsqualität. Das System muß in der Lage sein, Video- und Audiosequenzen in angemessener Qualität zu präsentieren.

Synchronisation und Timing. Für Video- und Audiodaten in Hypermediasystemen sollte eine Synchronisation stattfinden.

2.2.3.2 Anforderungen an Hypermediabeschreibungen

Die wichtigsten Konstrukte und Semantiken, die ein Modell zur Beschreibung von Hypermedia benötigt, werden im Folgenden kurz vorgestellt [HBvR94].

Temporäre Information. Temporäre Synchronisation zwischen Komponenten muß durch das Modell spezifiziert werden. Hierfür gibt es drei Möglichkeiten. Die erste ist die Synchronisation innerhalb einer Komponente (*versteckte Struktur*), die zweite die Synchronisation in der Verknüpfungsstruktur zwischen Konponenten (*separate Struktur*) und die dritte eine Kombination der anderen beiden Möglichkeiten (*zusammengesetzte Struktur*).

Verknüpfungen. Dies sind wohl die wichtigsten Elemente eines Hypertextsystems. Sie legen logische Navigationsmechanismen in einem Dokument fest. Mittels des Dexter-Modells können allerdings keine Verknüpfungen innerhalb einer Komponente beschrieben werden. Außerdem ist in Multimediasystemen die Festlegung von Verknüpfungen aufgrund von Fragmentierungsproblemen recht komplex.

Ein weiteres Problem bei gegenwärtigen Hypertextsystemen ist, daß nicht explizit festgelegt wird, welche Informationen weiterhin präsentiert werden, wenn einer Verknüpfung gefolgt wird. Zur Lösung dieses Problems muß entweder eine Methode definiert werden, die es erlaubt, Teile einer Komponente für Verknüpfungszwecke als Mini-Komponente zu behandeln, oder es muß ein generalisierteres Verknüpfungsverhalten definiert werden. Eine Lösung ist die Einführung von Verknüpfungskontexten.

Hochstufige Präsentationsspezifikationsattribute. Es ist für Hypermediadokumente notwendig, Präsentationsinformationen auf einer höheren als der Komponentenebene zu spezifizieren. Diese globalen Attribute sollten als Default-Werte einer Klasse von Medien- oder Informationstypen zugeordnet sein oder als Satz von Basisattributen existieren.

2.2.3.3 Das Amsterdam Hypermedia Modell

Ein Modell, welches versucht, die Natur von hyperstukturierten Multimediapräsentationen zu beschreiben, ist das Amsterdam Hypermedia Model (AHM) [HBvR94]. Es ist das Ziel dieses Modells, ähnlich dem Dexter-Modell einen generellen Rahmen zu definieren, mit dem die Grundkonstrukte und Aktionen beschrieben werden können, die einem weiten Spektrum von Hypermediasystemen gemein sind.

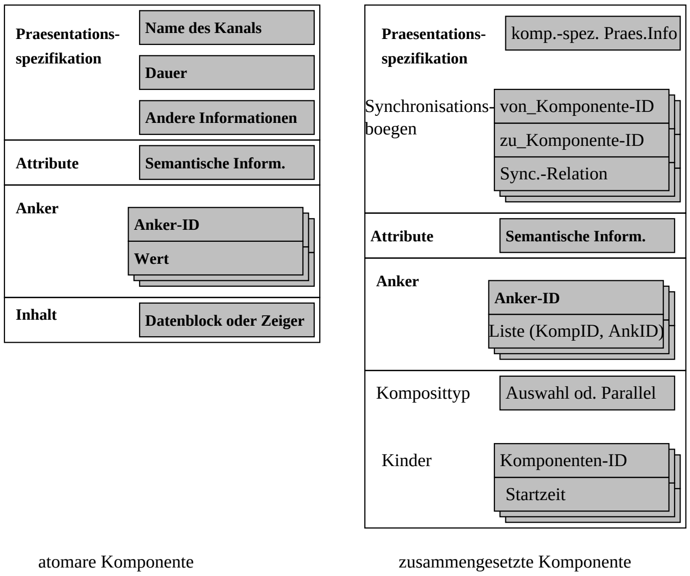


Abbildung 2.3: Komponenten im AHM

Vorstellung des Amsterdam Hypermedia Modells

Das AHM wurde entwickelt, um eine verständliche Basis für kombinierte Hyper- und Multimediaforschung zu liefern. Es wurde definiert, indem das Dexter-Hypertextmodell mit dem CMIF-Multimediamodell [vRJMB93] kombiniert wurde, und anschließend Erweiterungen hinzugefügt wurden, um jene Hypermediaanforderungen zu unterstützen, die dadurch nicht abgedeckt wurden. Es soll hier kurz vorgestellt werden.

AHM-Komponenten: Die Zusammensetzung einer AHM-Komponente wird in Abbildung 2.3 vorgestellt. Eine atomare Komponente besteht aus einer Präsentationsspezifikation, Attributen, Ankern und dem Inhalt. Die in der Präsentationsspezifikation festgelegten *Kanäle* werden weiter unten erläutert. Eine zusammengesetzte Komponente besteht ebenfalls aus einer Präsentationsspezifikation, die hier als weiteres Konstrukt *Synchronisationsbögen* beinhaltet, die weiter unten vorgestellt werden, sowie Attributen, Ankern, einer Festlegung eines Komposittypen, der die Art der Zusammensetzung festlegt, und einer Auflistung der Kindkomponenten.

Temporäre Beziehungen: Der Hauptmechanismus zur Definition von temporären Beziehungen im AHM ist das Konzept der zusammengesetzten Komponente. Eine zusammengesetzte Komponente besteht aus atomaren Komponenten und/oder weiteren zusammengesetzten Komponenten. Zwischen den Kindkomponenten besteht *grobe* oder *feine* Synchronisation, wobei letztere durch sogenannte *Synchronisationsbögen* spezifiziert wird. Ein solcher Synchronisationsbogen legt temporäre Einschränkungen zwischen zwei Komponenten fest, indem er für diese Komponenten ein Intervall, das die Zeit zwischen ihrer jeweiligen Startzeit festlegt, eine maximale und minimale Abweichung davon, und andere Attribute definiert (siehe [HBvR94]).

AHM Verknüpfungskontext: Um Informationen, die das Verhalten der Komponenten beschreiben, wenn eine von dieser Komponente wegführende Verknüpfung verfolgt wird, zu beschreiben, definiert AHM die Notation des Verknüpfungszusammenhangs. Ein Kontext ist eine (meist) zusammengesetzte Komponente, die eine Sammlung von Komposit- oder Atomkomponenten, die durch eine Verknüpfungsoperation beeinflußt werden, umfaßt. Es gibt Quell- und Zielkontexte. Der Kontextmechanismus gestattet es, verschiedene Anzeigeoptionen mit jeder Verknüpfung zu assoziieren. Der Vorteil hiervon ist, daß nur ein Teil der Dokument-struktur durch das Abschreiten einer Verknüpfung verändert werden muß. Komponenten der Präsentation, die höher in der Kompositionshierarchie stehen, können aktiv bleiben.

AHM-Kanäle: Codierung von hochstufigen Präsentationsattributen: Globale Präsentationsattribute können mit sogenannten *Kanälen* festgelegt werden. Kanäle sind abstrakte Ausgabegeräte zum Abspielen eines Komponenteninhalts wie z.B. ein Fenster oder eine Audioausgabe. Jedem dieser Kanäle können nun voreingestellte Präsentationswerte zugeordnet werden. Beim Abspielen eines Hypermediadokuments werden die Kanäle auf physische Ausgabegeräte abgebildet.

Durch die Benutzung von Kanälen kann ein- und dasselbe Dokument auf verschiedene Weise präsentiert werden, indem einfach die Kanäle neu spezifiziert werden. Dadurch wird Konsistenz gefördert. Flexibilität wird gefördert, indem das Überschreiben von individuellen Komponenten gestattet wird.

2.3 Die hypermediale Patientenakte

Eine der wesentlichen Zielsetzungen der Diplomarbeit besteht in der Entwicklung einer *hypermedialen Patientenakte*. Die Motivation, die hinter der Untersuchung dieser Form einer Patientenakte liegt, besteht in erster Linie in der Ausnutzung der in Abschnitt 2.2 genannten Vorteile von Hypertext, Multimedia und Hypermedia im Krankenhausbereich. Den genannten Vorteilen seien hier noch die folgenden krankenhausspezifischen Gründe hinzugefügt:

- Ein vernünftig implementiertes Hypermediasystem ist durch seine Konzeption benutzerfreundlich, indem es die natürliche Denkweise des menschlichen Gehirns unterstützt, nämlich Informationseinheiten logisch miteinander zu verknüpfen und in keiner festgelegten Reihenfolge zu besuchen. Der Aspekt der Benutzerfreundlichkeit darf vor allem in der Krankenhausumgebung nicht vernachlässigt werden.

- Zu einer Patientenakte gehören im Regelfall viele Bilddaten, seien es nun Röntgenfilme oder Ultraschallbilder. Es wäre wünschenswert, die Ansicht dieser Bilder und Filme direkt über dasselbe Computersystem zu ermöglichen, mit dem auch andere Patientendaten abgerufen oder Anforderungen verschickt werden können. Vor allem bei der Visite kann dies die Arbeit von Ärzten und Pflegepersonal sehr erleichtern.

- Viele Krankenhäuser nutzen vor allem im Verwaltungsbereich schon existierende Anwendungen verschiedenster Couleur, seien es einfache Textverarbeitungsprogramme, Datenbanken oder spezialisierte Verwaltungsanwendungen. Ein Hypermediasystem ist im Idealfall in der Lage, auf bestehende Anwendungen aufzusetzen und Datendokumente verschiedenster Herkunft miteinander zu verknüpfen. Dadurch entsteht eine gewisse Zeit- und Kostenersparnis, indem verhindert wird, daß alte Systeme komplett ersetzt werden müssen.

Mit der Einführung einer hypermedialen Patientenakte können zudem folgende Ziele erreicht werden:

- Es gibt durch die oben schon genannte Benutzerfreundlichkeit eine erhöhte Akzeptanz durch das Personal. Dadurch wird die Effizienz der täglichen Arbeit erhöht.

- Die Kommunikation zwischen einzelnen Leistungsstellen einerseits und Einzelpersonen andererseits wird durch eine Netzwerklösung, die sich für eine Hypermediaanwendung anbietet, und die ein E-Mail-System umfaßt, gefördert. Bei entsprechender Anbindung an weltweite Kommunikationsstrukturen wie das Internet ist es möglich, weltweit zu kommunizieren.

2.4 Hypermedia-Entwicklungsmethoden

Für die Entwicklung spezifischer Ausprägungen von Software-Systemen, beispielsweise für Informationssysteme, Realzeitsysteme oder parallele Anwendungen, existieren eine Vielzahl von Methoden. Demgegenüber gibt es nur wenige Methoden, die an die spezifischen Anforderungen von Hypermedia-Anwendungen angepaßt sind. In diesem Abschnitt werden die wichtigsten Hypermedia-Entwicklungsmethoden, die im Rahmen von Literatur- und Internet-Recherchen identifiziert wurden, vorgestellt, verglichen und bewertet. Dies umfaßt neben der in der Aufgabenstellung vorgegebenen Methode OOHDM die Richtlinien von Balasubramanian und die Methode RMM. Um den Rahmen der Diplomarbeit nicht zu sprengen, wurde auf die Diskussion der Methode HDM, dem Vorgänger von OOHDM, verzichtet. Der interessierte Leser sei auf die entsprechende Literatur [GSP93] verwiesen.

2.4.1 Richtlinien nach Balasubramanian

In diesem Kapitel werden die von Balasubramanian in [Bal94] festgelegten Richtlinien zum Entwurf von Benutzerschnittstellen für Hypermediasysteme vorgestellt.

Balasubramanian formuliert in dem technischen Bericht [Bal94] Richtlinien zum Entwurf von Benutzerschnittstellen für ein auf einem kognitiven Modell basierendes Hypermediarahmenwerk. Er geht davon aus, daß das Rahmenwerk Knoten und Verknüpfungen in verschiedene semantische Typen einteilt [RT90]. Eine solche Klassifizierung ist bei der Entwicklung einer Entwurfsmetapher und Benutzerschnittstelle sehr hilfreich. Im folgenden werden diese Richtlinien kurz vorgestellt:

Identifikation der Metapher. Die Wahl einer guten Metapher unterstützt sowohl den Benutzer durch Wiedererkennung existierender Strukturen, als auch den Entwickler eines Hypermediasystems durch Bereitstellung eines festen Rahmenwerks, welches diesem Grenzen zwecks Sicherung der Konsistenz setzt. Die hier vorgeschlagene unterliegende Metapher ist ein generelles menschliches kognitives Modell zur komplexen Problemlösung. Dieses Modell basiert auf einem semantischen Netzwerk, in dem Konzepte durch Assoziationen miteinander verbunden sind.

Identifikation aller Systemobjekte. In dem oben vorgeschlagenen Rahmenwerk werden sowohl Knoten als auch Verknüpfungen als Objekte betrachtet. Knoten werden in sechs semantische Typen unterteilt: *Detail, Kollektion, Vorschlag, Zusammenfassung, Thema* und *Beobachtung*. Verknüpfungen werden in zwei Hauptklassen eingeteilt: ***konvergente*** und ***divergente*** Verknüpfungen. Konvergente Verknüpfungen werden weiter in *Spezifikations-, Mitglied-, Assoziations-, Pfad-, Alternativ-* und *Inferenz*-Verknüpfungen, divergente Verknüpfungen in *Ausführungs-, Oppositions-, Spekulations-, Verzweigungs-, Lateral-* und *Extrapolations*-Verknüpfungen unterteilt. Die Betonung dieses Rahmenwerks liegt auf der Assoziation von Semantik mit Knoten und Verknüpfungen. Weiterhin wird die Verwendung von zusammengesetzten Knoten und Vorlagen propagiert, da diese Konstrukte ein schnelleres Verständnis des durchschnittlichen Benutzers des Modells und der Metapher fördern.

Identifikation aller Aktionen der Objekte. Es muß eine Festlegung aller möglichen Aktionen und Funktionen der Knoten- und Verknüpfungstypen erfolgen. Aktionen können generisch, explizit oder kontrollierend sein.

Generische Aktionen: Dies sind Aktionen, die Knoten modifizieren, hinzufügen oder löschen.

Explizite Aktionen: Diese Aktionen benötigen einen expliziten Qualifikator oder Modifikator. Ein Beispiel hierfür ist das Auflisten einer Knotenmenge mit einer bestimmten Eigenschaft.

Kontrollfunktionen: Kontrollfunktionen stellen Übersichts- und Zoomfunktionen, Vor- und Rückwärtsnavigationsmöglichkeiten, Pfade und andere Steuerungskontrollen zur Verfügung.

Identifikation aller Modifikatoren, die Untermengen von Objekten auswählen. Sämtliche Knoten und Verknüpfungen werden innerhalb des Rahmenwerks durch ihren semantischen Typ identifiziert. Diese Typinformation kann als Modifikator oder Qualifikator für ein solches Objekt gesehen werden.

Identifikation aller strategischen Wahlmöglichkeiten. Strategische Wahlmöglichkeiten umfassen Benutzerinteraktionsmöglichkeiten mit dem System, die das Ziel haben, eine spezielle Aufgabe zu erfüllen. Dies können Übersichtsanfragen, Navigationsmöglichkeiten, Editierwerkzeuge, Anzeigeoptionen und andere sein. Im optimalen Fall eines Hypermediasystems sollten strategische Wahlmöglichkeiten überflüssig sein, da jeder Knoten in sich abgeschlossen und vollständig ist.

Identifikation lateraler Klassifikationen. Die Fähigkeit von Hypermediasystemen, eine laterale Verknüpfung zu einer anderen Version eines Knotens oder einer Anmerkung dazu herzustellen, kann als laterale Klassifikation betrachtet werden.

Identifikation von Objekt- und Objektteilformaten. Der Autor eines Hypermediasystems muß ein Format für Knoten, Verknüpfungen, Übersichtsdiagramme, Indizierungsmechanismen, Abfragemöglichkeiten und Abfrageergebnisse definieren.

Identifikation von Objektlisten. Es ist notwendig, Listen von semantisch zusammenhängenden Knoten, Verknüpfungen und Vorlagen, zuletzt besuchten Knoten oder zuletzt modifizierten Knoten und Verknüpfungen zu erstellen.

Identifikation von reaktiven Auswahlmöglichkeiten für eine Objektliste. Einige Möglichkeiten hierfür sind die direkte Interaktion mit einem Knoten über ein Übersichtsdiagramm, eine Vorschau eines Knotens und davon wegführender Verknüpfungen, das Markieren eines Textausschnitts und das Hinzufügen neuer Verknüpfungen.

Identifikation gemeinsam genutzter Prozesse. Die Fähigkeit, Objekte zu erschaffen oder sie aufzulisten, ist im gesamten System unabhängig von der Art des Objekts möglich. Diese Prozesse können als gemeinsam genutzte Prozesse gesehen werden, die je nach Art des benutzten Objekts unterschiedlich ablaufen. Dieser gemeinsam genutzte Prozess des Erzeugens kann erweitert werden, um Vorlagen zu erschaffen.

Identifikation aller Benutzerinteraktionszustände. Der Entwickler muß alle möglichen Interaktionszustände, Navigationspfade, Listenanforderungen, Anfragen und Rückverfolgungen identifizieren. Der Benutzer sollte jederzeit über den gegenwärtigen Interaktionszustand informiert sein. Sämtliche Benutzerinteraktionen können in einer Historie abgespeichert werden.

Identifikation notwendiger Hilfe. Kontextsensitive Hilfe ist zur Vermeidung von Desorientation in einer Hypermediaumgebung essentiell. Aus diesem Grund müssen alle Knoten- und Verknüpfungstypen durch Beispiele erläutert werden. Hilfe kann auch in Form von Rückmeldungen in Bezug auf den gegenwärtigen Interaktionszustand, als in einem bestimmten Zustand zur Verfügung stehende reaktive Auswahlmöglichkeiten oder als navigationelle Hinweise existieren.

Identifikation aller Fehlerzustände. Es sollten alle möglichen Fehlerzustände identifiziert werden. Beispielsweise sollten Menüpunkte oder Auswahlmöglichkeiten, die in einem bestimmten Interaktionszustand nicht möglich sind, auch nicht angezeigt werden. Fehlermeldungen sollten präzise und konstruktiv gehalten sein, damit der Benutzer über den exakten Grund des Fehlers und mögliche Schritte zu seiner Behebung informiert wird.

Identifikation des Bildschirmlayouts. Obwohl die Bildschirmlayoutentwicklung anwendungsabhängig ist, müssen grafische Entwurfs- und Benutzbarkeitsprinzipien hierbei berücksichtigt werden. Beispielsweise sollte die Menüleiste strategische Auswahlmöglichkeiten beinhalten, die in verschiedenen Systemzuständen konsistent sind.

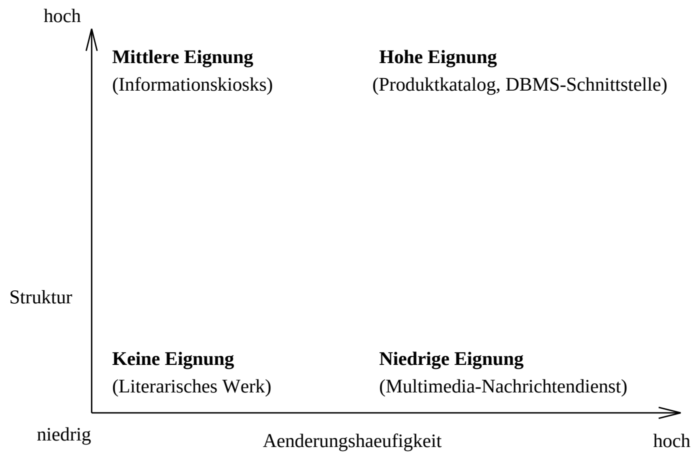


Abbildung 2.4: Nützlichkeit von RMM für verschiedene Systeme

2.4.2 RMM

In diesem Kapitel wird die Methode RMM zum Entwurf von Hypermediaanwendungen vorgestellt, indem zunächst ein Überblick gegeben, das zugrundeliegende Datenmodell vorgestellt und anschließend die einzelnen Entwicklungsschritte näher erläutert werden. RMM wird in [BBI96, ISB95, IKK96] ausführlich vorgestellt.

2.4.2.1 Überblick

An dieser Stelle wird die Methodologie RMM (Relationship Management Methodology) vorgestellt, welche von Isakowitz, Stohr und Balasubramanian [ISB95] zum Zweck des Entwurfs und der Konstruktion von Hypermediasystemen entwickelt wurde. Der Name *Relationship Management* unterstreicht die Sicht von Hypermedia als Vehikel zur Verwaltung von Beziehungen zwischen Informationsobjekten.

Abbildung 2.4 beschreibt die Nützlichkeit von RMM für verschiedene Systeme, die nach dem Grad ihrer Strukturierung und Änderungshäufigkeit klassifiziert werden.

RMM konzentriert sich im Softwareentwicklungszyklus auf die Entwurfs-, Entwicklungs- und Konstruktionsphasen, wie in Abbildung 2.5 dargestellt. Die Phasen Eins bis Sieben werden nach der Beschreibung des zugrundeliegenden Datenmodells RMDM näher erläutert, wobei die Phasen Vier bis Sieben nur sehr kurz zusammengefaßt werden.

2.4.2.2 Relationship Management Data Model (RMDM)

Ein Datenmodell, das ein generelles Hypertextsystem beschreibt, detailliert dessen Struktur. Es ist aber bei der Modellierung von Hypermediaanwendungen nur von geringem Nutzen, da die Modellierung spezieller Anwendungen spezielle Beschreibungsmittel und -konzepte erfordert. RMDM stellt hier eine Sprache zur Beschreibung der Navigationsobjekte und Navigationsmechanismen in Hypermediaanwendungen zur Verfügung. RMDM ist der Grundstein von RMM.

Abbildung 2.6 stellt die Modellierungsprimitive von RMDM dar. Die *Bereichsprimitive* modellieren Informationen über den Anwendungsbereich. Das *Ausschnittprimitiv* wird benutzt, um Attribute einer Entität logisch zu gruppieren. Beispielsweise könnte eine Entität *Professor* die Attribute *Name, Alter, Bild, Interessen* und *Biografie* umfassen. Diese Attribute könnten in einen *generellen* Ausschnitt, der die Attribute *Name, Alter* und *Bild*, und in einen Ausschnitt *Forschungsgebiete*, der die Attribute *Name* und *Interessen* enthält, gruppiert werden. Je nach Kontext der Anwendung

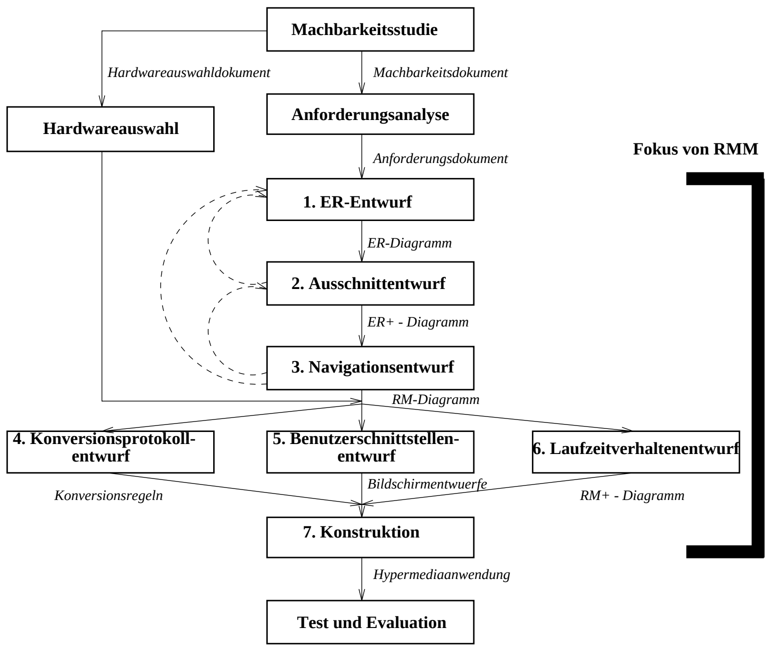


Abbildung 2.5: RMM im Softwareentwicklungszyklus

kann dem Benutzer einer der beiden Ausschnitte oder auch beide zusammen präsentiert werden. Die *Zugangsprimitive* beschreiben Navigation in RMDM. Das P repräsentiert eine Bedingung, die festlegt, welche Instanzen einer Entität zugänglich sind.

2.4.2.3 ER-Entwurf

In diesem Entwicklungsschritt wird der Informationsbereich der Anwendung durch ein Entity-Relationship-Diagramm dokumentiert. Die hier dargestellten Entitäten und Beziehungen stellen die Grundlage der Hypermedia-Anwendung dar. Das Ziel dieses Schrittes ist es, Verknüpfungen zwischen Objekten explizit zu definieren, da diese die Hauptpfade zwischen den einzelnen Informationsobjekten reflektieren. Abbildung 2.7 stellt exemplarisch einen ER-Entwurf für ein Fakultätsinformationssystem dar.

2.4.2.4 Ausschnittentwurf

In diesem Schritt, der nur für Hypermedia-Anwendungen relevant ist, wird festgelegt, wie die in einer Entität vorhandenen Informationen einem Benutzer präsentiert und zugänglich gemacht werden. Konkret bedeutet dies, daß die Attribute einer Entität in sinnvolle, nicht disjunkte Untermengen unterteilt werden. Jede Entität besitzt einen *Hauptausschnitt*, der als Default-Wert für Verknüpfungen, die zu der Entität führen, benutzt wird.

In dem Auschnittdiagramm, welches in diesem Schritt konstruiert und als ER+-Diagramm bezeichnet wird, wird auch die Navigation zwischen Auschnitten durch uni- und bidirektionale Verknüpfungen modelliert. Diese Verknüpfungen zwischen Ausschnitten werden als *strukturelle* Verknüpfungen bezeichnet, um sie von *assoziativen* Beziehungen im ER-Diagramm zu unterscheiden. Der Unterschied besteht darin, daß strukturelle Verknüpfungen Informationseinheiten innerhalb derselben Entität verbinden, während assoziative Beziehungen verschiedene Entitäten, die im allgemeinen zu verschiedenen Entitätsklassen gehören, verknüpfen.

In diesem Schritt müssen vier Aufgaben bearbeitet werden:

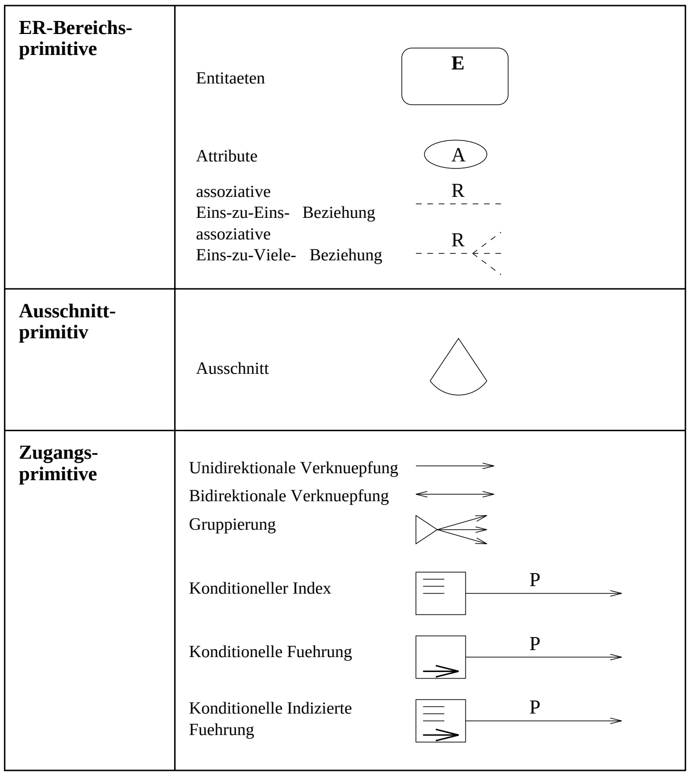


Abbildung 2.6: RMM: Primitive

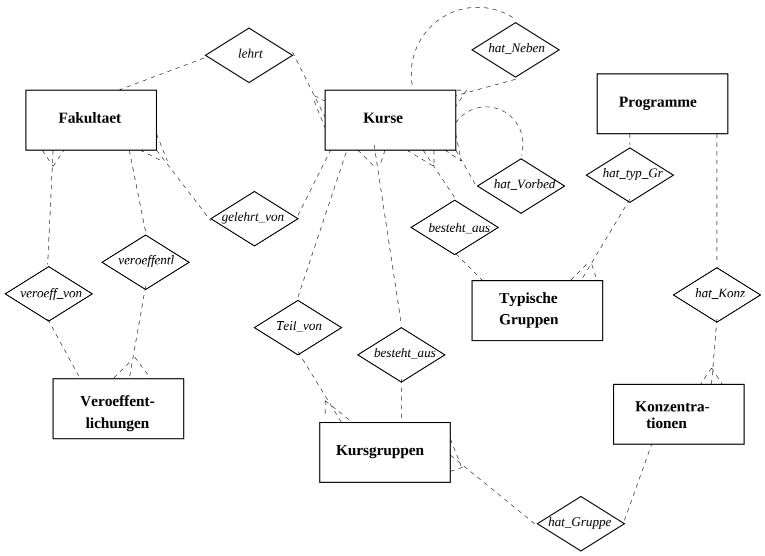


Abbildung 2.7: RMM: Beispiel für den ER-Entwurf

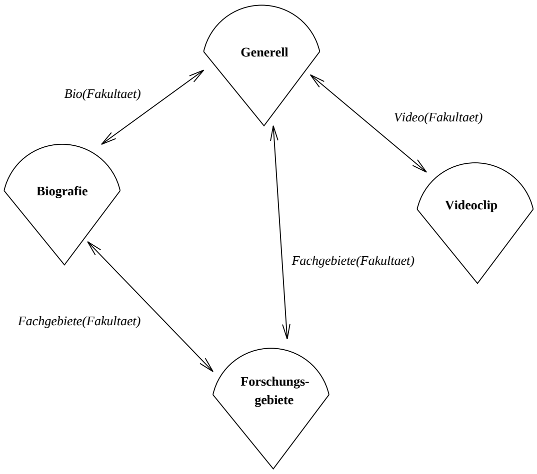


Abbildung 2.8: RMM: Beispiel für den Ausschnittentwurf

1. Unterteilung einer Entität in Ausschnitte
2. Bestimmung eines Ausschnitts als Kopfausschnitt
3. Verbinden der Ausschnitte
4. Benennung der Verknüpfungen

Ein Beispiel für den Ausschnittentwurf findet sich in Abbildung 2.8.

2.4.2.5 Navigationsentwurf

In diesem Entwicklungsschritt werden die Pfade als Grundlage für die Navigation entworfen. Dies geschieht durch sorgfältige Analyse der assoziativen Beziehungen, die im ER+-Diagramm spezifiziert wurden. Wenn über eine assoziative Beziehung navigiert werden soll, wird sie durch eine oder mehrere der RMDM-Zugangsstrukturen ersetzt. Da RMM für sich häufig verändernde Datenbereiche konzipiert wurde, werden die navigationellen Pfade in generischen Begriffen spezifiziert. Aus diesem Grund werden Verknüpfungen nicht fest zwischen Instanzen von Entitäten definiert, sondern durch Bezugnahme auf Eigenschaften von Entitäten und Beziehungen festgelegt.

Zunächst werden in diesem Schritt Navigationsmöglichkeiten zwischen Entitäten festgelegt, die auf den assoziativen Beziehungen aus dem ersten Modell basieren. Anschließend werden hochstufige Zugangsstrukturen durch Gruppierung von Interessengebieten definiert. Zuletzt können für Entitäten verschiedene Eingangsausschnitte festgelegt werden.

Nach Beendigung dieser Phase ist das ER+-Diagramm der letzten Phase in ein RMDM-Diagramm umgewandelt worden, welches alle Zugangsstrukturen des Systems beschreibt.

Während dieses Schrittes muß der Entwickler

1. die zugänglichen Informationskomponenten und Beziehungen,
2. die vorhandenen Gruppierungen und
3. die in jedem einzelnen Fall benutzten Zugangsstrukturen

identifizieren.

Ein Beispiel für ein RMDM-Diagramm des Navigationsentwurfs findet sich in Abbildung 2.9.

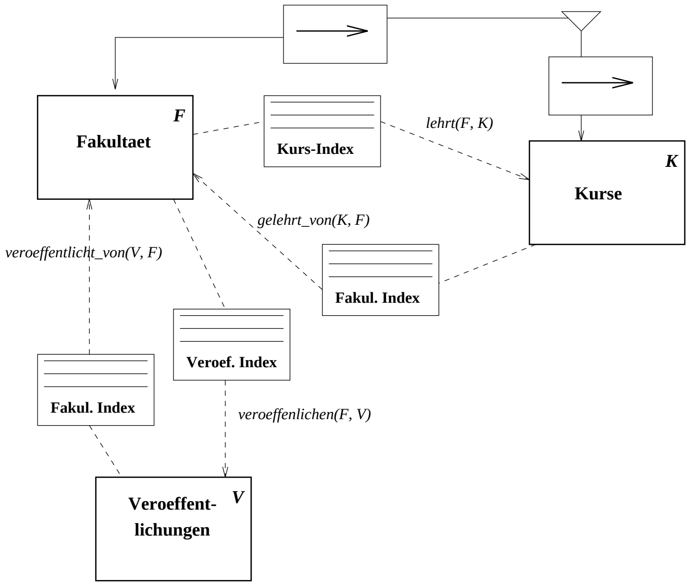


Abbildung 2.9: Beispiel für ein RMDM-Diagramm nach dem Navigationsentwurf

2.4.2.6 Weitere Phasen

Im folgenden werden die weiteren Phasen von RMM, die explizit unterstützt werden, kurz erläutert:

Konversionsprotokollentwurf: Während dieses Schrittes wird eine Menge von Konversionsregeln benutzt, um jedes Element des RMDM-Diagramms in ein Objekt der Zielplattform umzuwandeln. Dieser Schritt wird zur Zeit von den Programmierern von Hand ausgeführt.

Benutzerschnittstellenentwurf: In diesem Schritt wird für jedes Objekt, welches im RMDM-Diagramm erscheint, ein Bildschirmlayout entworfen. Dies beinhaltet Button-Layouts, Erscheinungsbilder von Knoten und Indizes und Lokalisierung von navigationellen Hilfen.

Laufzeitverhaltenentwurf: Entscheidungen über Verknüpfungsabschreitungen, Historien, Rückverfolgung und Navigationsmechanismen werden während dieses Schrittes getroffen. Unter Berücksichtigung der Änderungshäufigkeit der Daten wird ebenfalls während dieses Schrittes festgelegt, ob Knoteninhalte schon während der Konstruktion der Anwendung festgelegt oder während der Laufzeit dynamisch berechnet werden.

Konstruktion und Test: Diese Phase entspricht derjenigen in traditionellen Softwareentwicklungsmodellen. In Hypermediaanwendungen muß speziell Wert auf gründliches Testen aller Navigationspfade gelegt werden.

2.4.3 OOHDM

In diesem Abschnitt wird die Methode OOHDM zum Entwurf von Hypermediaapplikationen vorgestellt. Zunächst wird ein Überblick gegeben, anschließend werden die einzelnen Entwicklungsphasen näher betrachtet. OOHDM wird in den Artikeln [SRB95b, RS94, SRB95a, RSLC95, BS94] ausführlich behandelt.

2.4.3.1 Überblick

OOHDM ist ein Akronym für Object-Oriented Hypermedia Design Method. Die Methode wurde von Schwabe, Rossi und Barbosa entwickelt und basiert auf der Methode HDM [GSP93]. OOHDM

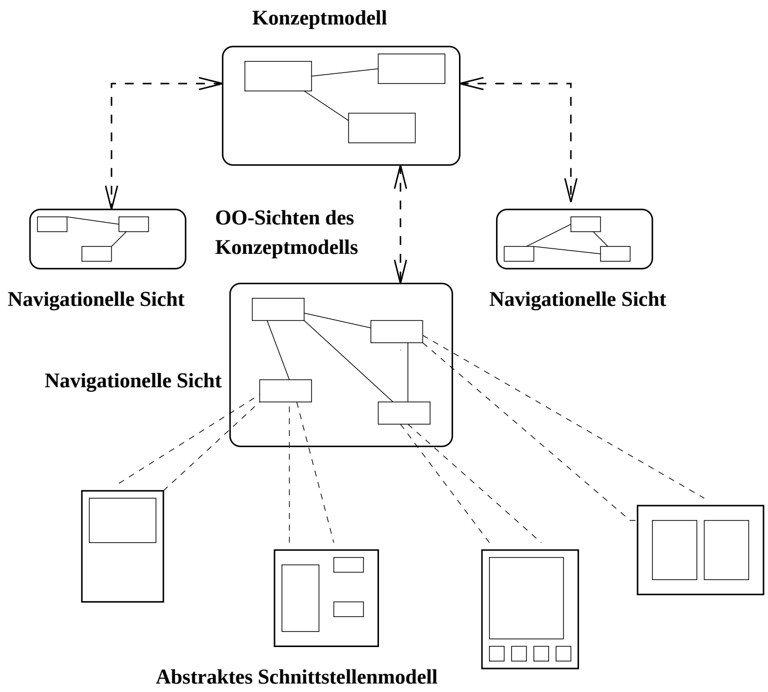


Abbildung 2.10: Verhältnis der Entwurfsmodelle von OOHDM

ist ein modellbasierter Ansatz zur Konstruktion von Hypermediaapplikationen. OOHDM setzt sich aus vier Phasen zusammen: Konzeptentwurf, Navigationsentwurf, abstrakter Schnittstellenentwurf und Implementation. Das Vorgehen ist inkrementell, iterativ und prototypbasiert. Während jeder Phase außer der letzten wird ein Satz von objektorientierten Modellen, die unterschiedliche Entwurfsaspekte beschreiben, konstruiert oder erweitert. In Abbildung 2.10 wird das Verhältnis der Entwurfsmodelle untereinander grafisch dargestellt. Abbildung 2.11 faßt Produkte, Mechanismen und Ziele der Phasen zusammen.

2.4.3.2 Konzeptentwurf

In diesem Schritt wird ein konzeptionelles Modell des Problembereichs konstruiert. Das Konzeptmodell setzt sich aus Objekten, Klassen, Beziehungen und Subsystemen zusammen. Es besteht aus einem Satz von Objekten und Klassen, die durch Beziehungen miteinander verbunden und in Subsystemen zusammengefaßt sind. Die Notation ist OMT (Object Modeling Technique) [RBP+91] entlehnt. Rechtecke repräsentieren Klassen mit Namen- und Attributsinformationen darin, Ovale fassen Subsysteme zusammen, Linien stellen Beziehungen dar, wobei ein Dreieck an einer Linie oder eine Raute am Ende einer Linie eine Vererbungsbeziehung bzw. eine Aggregation symbolisieren. Abbildung 2.12 stellt die Modellierungsprimitive der ersten zwei Entwurfsphasen dar. Ein Beispiel für den Konzeptentwurf ist in Bild 2.13 zu sehen. Das Beispiel enthält drei Hauptklassen: `Stadt`, `StadtBild` und `Land` sowie zwei Subsysteme: `Stadtgeschichte` und `AllgemeineInformation`. `Länder` und `Städte` sind Aggregationen von `Städten` bzw. `StadtBildern`.

Im folgenden werden die Primitive des Konzeptentwurfs kurz vorgestellt:

Objekte und Klassen: Diese Konstrukte sind aus der objektorientierten Entwicklung bekannt und werden hier nicht näher erläutert.

Subsysteme: Subsysteme dienen der Zusammenfassung von logisch zusammengehörigen Klassen und helfen, den Systementwurf übersichtlicher zu gestalten.

Phase	Produkte	Mechanismen	Entwurfsziele
Konzept-Entwurf	Klassen, Subsysteme, Beziehungen, Attibutsperspektiven	Klassifikation, Komposition, Generalisierung, Spezialisierung	Modellierung der Semantik des Anwendungsbereichs
Navigations-Entwurf	Knoten, Verknuepfungen, Navigationelle Kontexte, Nav. Transformationen	Abbilden der Konzept- auf die Navigationsobjekte	Beruecksichtigung von Benutzerprofil und Aufgabe; Betonung auf kognitiven Aspekten
Abstrakter Schnitstellenentw.	Abstrakte Schnittstellenobjekte, Reaktionen auf externe Ereignisse, Schnittstellentransformationen	Abbilden der Konzept- auf die wahrnehmbaren Objekte	Modellierung wahrnehmbarer Objekte; Implement. gewaehlter Metaph., Beschr. der Schnittst. fuer nav. Objekte
Implementation	Funktionierende Anwendung	Die durch die Zielumgebung bereitgestellten	Performanz, Vollstaendigkeit

Abbildung 2.11: Zusammenfassung von OOHDM

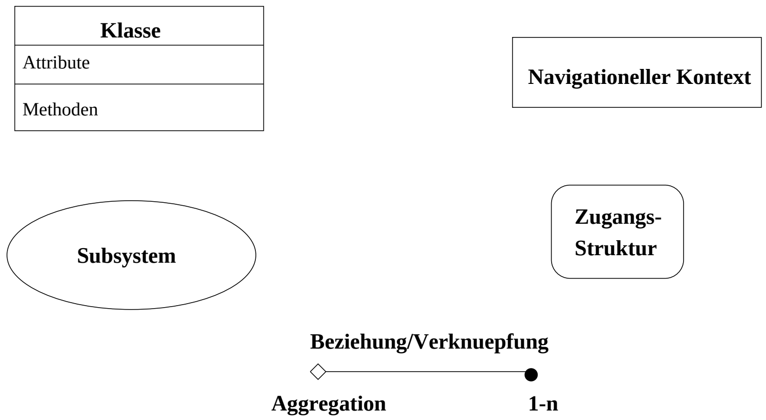


Abbildung 2.12: Modellierungsprimitive von OOHDM

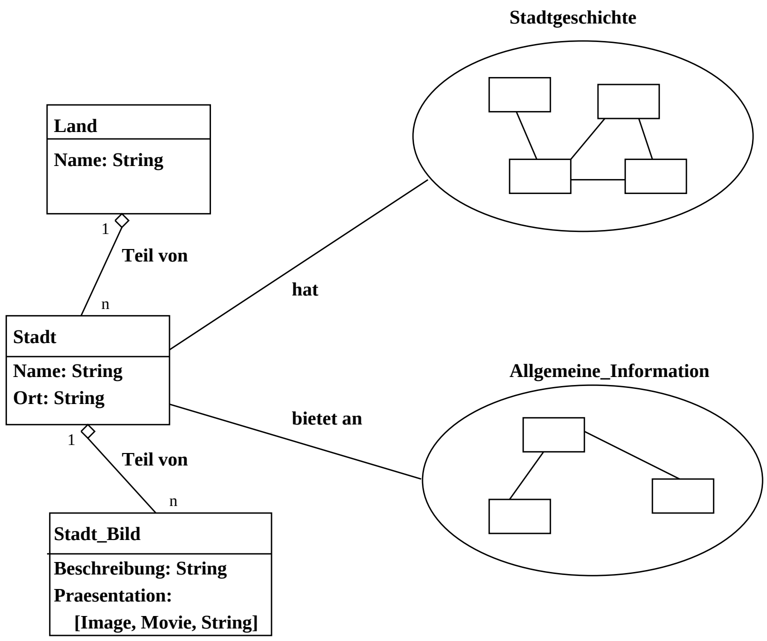


Abbildung 2.13: Beispiel für den Konzeptentwurf mit OOHDM

Beziehungen: Beziehungen drücken in dem Modell Verwandschaften zwischen Problembereichsobjekten aus und werden in den Navigationssichten auf Verknüpfungen abgebildet. Die Definition von Beziehungen geschieht unabhängig von navigationaler Semantik. Beziehungen werden auch als Klassen definiert und beinhalten daher Attribute und Verhalten. Sie sind in Hierarchien organisiert. Kardinalitätseinschränkungen können festgelegt werden.

Attribute und Attributsperspektiven: Klassenattribute sind typisiert und repräsentieren intrinsische oder konzeptionelle Eigenschaften von Objekten. Der Typ (oder die Klasse) eines Attributes repräsentiert einen Datentyp wie Integer oder String oder einen Medientyp wie Movie oder Image, der in zukünftigen Systemen eingesetzt wird. Für ein Attribut kann es mehrere Darstellungsformen geben, wobei je nach Kontext eine davon ausgewählt wird.

Abstraktionskonstrukte: In OOHDM werden zwei aus der Objektorientierung allgemeine bekannte Abstraktionskonstrukte zur Behandlung der Komplexität der Klassenhierarchie verwendet: Aggregation und Generalisation/Spezialisierung.

Modelldokumentation: Modelldokumentation ist ein sehr häufiges Problem bei der Konstruktion von großen Systemen, besonders im Hypermediabereich. OOHDM propagiert die Benutzung von Karten, die vergleichbar mit CRC-Karten [WB90] sind. Es gibt Klassen-, Beziehungen- und Subsystemkarten zur Modelldokumentation. Die Abbildung 2.14 stellt eine Klassen- und eine Subsystemkarte dar. Beispielhaft soll hier kurz die Klassenkarte vorgestellt werden.

Eine Klasse besitzt einen eindeutigen Namen, welcher im Feld `Klassenname` vermerkt wird. Das Feld `Erbt von` beinhaltet eine eventuell existierende Oberklasse. Das Feld `Attribute` umfaßt die Attribute der Klasse. Das Feld `Methoden` dokumentiert die Methoden. Das Feld `Verwandt mit` zählt die Beziehungen und Klassen, zu denen diese Beziehungen führen, auf. Wenn die Klasse Teil einer Aggregation oder eines Subsystems ist, wird dies im Feld `Teil von` vermerkt. Das Feld `Kommentar` beinhaltet einen Kommentar zu dieser Klasse. Die Felder `Trace fwd` und `Trace bck` verweisen auf Klassen in vorhergehenden bzw. nachfolgenden Entwicklungsschritten.

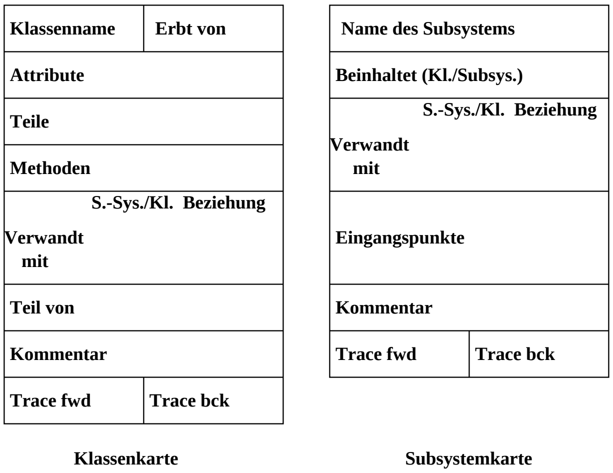


Abbildung 2.14: Karten zur Beschreibung des Konzeptmodells

2.4.3.3 Navigationsentwurf

Bei OOHDM wird das Navigationsmodell vom Konzeptschema abgeleitet, indem Komponenten definiert werden, die als logische Fenster auf die im Konzeptschema festgelegten Klassen fungieren. So können verschiedene Hypermediaapplikationen, jeweils repräsentiert durch eine Sicht, aus demselben Schema abgeleitet werden, wobei jede Sicht die Bedürfnisse einer bestimmten Benutzergruppe unterstützt.

Die Konstruktion eines Navigationsmodells aus dem Konzeptmodell beinhaltet die Definiton von Knotenklassen, Verknüpfungen, Zugangsstrukturen, navigationellen Kontexten, Kontextklassen und Navigationskarten für jede navigationelle Sicht.

Die Primitive des Navigationsmodells können in Anlehnung an [SRB95a] folgendermaßen beschrieben werden:

Knotenklassen: Knoten sind die grundlegenden Informationsbehälter in Hypermediaanwendungen. Ihre Struktur hängt von der Applikationssemantik ab, das heißt den speziellen Interessen der Anwender.

Eine Knotenklasse wird aus einer oder mehreren Konzeptklassen gebildet, formt also ein zusammengesetztes Objekt aus den Attributen und Methoden einer oder mehrerer Konzeptklassen. Wichtig ist hier die Spezifikation von Ankern, die die Verknüpfungen der Knotenklasse charakterisieren. Die Attribute einer Knotenklasse besitzen genau einen Typ, daß heißt, daß Attribute, die im Konzeptentwurf mehrere mögliche Typen hatten, hier auf einen Typ abgebildet werden müssen. Knotenklassen können in Vererbungshierarchien eingeordnet werden, die im Allgemeinen die Klassenhierarchie des Konzeptschemas wiederspiegeln.

Verknüpfungen: In OOHDM implementieren Verknüpfungen Beziehungen, die im Konzeptschema definiert wurden. Somit sind Verknüpfungen die navigationelle Verwirklichung von Beziehungen. Verknüpfungsklassen haben Verknüpfungsattribute und Verhalten, Quell- und Zielobjekte und Kardinalität. Verknüpfungsattribute beschreiben Eigenschaften der Verknüpfung selbst und sind bei der Definition von n-ären Verknüpfungen nützlich. In solchen Fällen kann sich die Verknüpfung wie ein Knoten verhalten und fungiert als intermediäres Objekt während der Navigation. Verknüpfungsklassen können genau wie Knotenklassen in Hierarchien organisiert werden, in denen abstrakte Klassen gemeinsames Verhalten vorgeben, und konkrete Klassen erst Struktur hinzufügen und letztendlich Verhalten spezialisieren.

Zugangsstrukturen: Zugangsstrukturen fungieren als Indizes oder Verzeichnisse und können unter anderem die Navigation über n-äre Verknüpfungen, wo eine Auswahl zum Zeitpunkt der

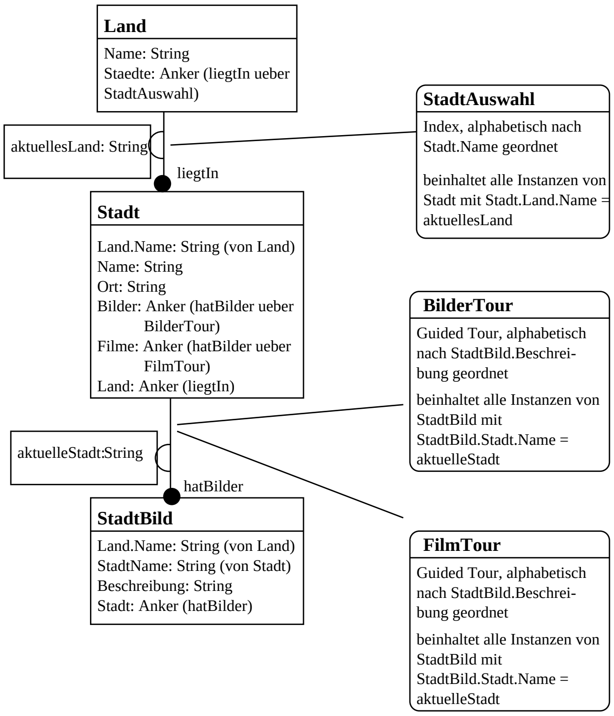


Abbildung 2.15: Beispiel für Klassen, Verknuepfungen und Zugangsstrukturen

Navigation getroffen wird, realisieren. Sie werden ebenfalls als Klassen modelliert und durch eine Menge von Selektoren und eine Menge von Zielobjekten charakterisiert. Selektoren gelten für Attribute der Zielobjekte. Die Zielmenge kann nach einer vorgegebenen Datenstruktur (geordnete Liste, Menge von Icons, etc.) organisiert werden. Ferner kann die Selektion durch Vorgabe von Prädikaten weiter eingeschränkt werden. Eine Zugangsstruktur kann einen navigationellen Kontext definieren. Sie wird im Rahmen dieser Diplomarbeit als Rechteck mit abgerundeten Kanten dargestellt.

Navigationelle Kontexte: Ein navigationeller Kontext ist eine Menge von Knoten, Verknüpfungen, Kontextklassen und anderen verschachtelten navigationellen Kontexten, die ein gemeinsames Navigationsverhalten haben. Für jeden navigationellen Kontext müssen das Navigationsverhalten sowie Ein- und Ausgänge festgelegt werden. Ein Kontextschema faßt grafisch alle navigationellen Kontexte für jede Sicht zusammen. Durch die Definition eines flexiblen Navigationsverhaltens für einen Kontext, wie beispielsweise die Möglichkeit, von jedem Element eines Kontexts jedes andere Element dieses Kontexts zu erreichen, werden Knotenklassen zusätzliche Anker hinzugefügt. Dieses Vorgehen wird durch die Definition von Kontextklassen dokumentiert.

Kontextklassen: In manchen Situationen müssen die Informationen in einem Knoten um kontextspezifische Zusatzinformationen ergänzt werden. Kontextklassen komplementieren die Definition einer navigationellen Klasse mit kontextabhängigen Informationen. Das heißt, daß hier einem navigationellen Objekt Attribute und Anker hinzugefügt werden, die nur in einem spezifischen Kontext sichtbar oder navigierbar sind. Abbildung 2.16 stellt Kontexte und Kontextklassen dar.

Navigationskarten: Navigationskarten liefern eine Notation zur Festlegung von Knotengruppen, wobei hier Mengen von Knoten gemeint sind, die gleichzeitig angezeigt werden können, und zur Festlegung des Effekts, der eintritt, wenn einer Verknüpfung aus dem aktuellen Knoten gefolgt wird. Die Notation verwendet eine XOR- und AND-Verschachtelung, die der durch

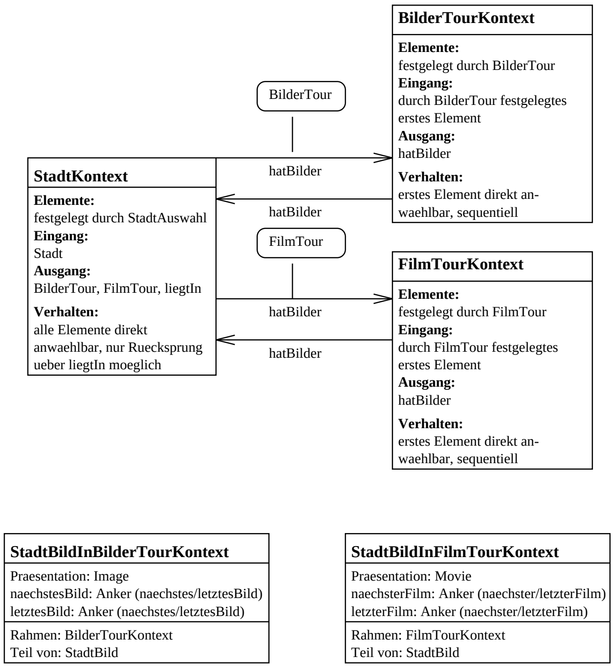


Abbildung 2.16: Beispiel für Kontextklassen und Navigationelle Kontexte

Statecharts [HPSS87] propagierten Notation entspricht.

Ein Beispiel für den Navigationsentwurf findet sich in den Abbildungen 2.15 und 2.16. Diese Beispiel führt das im Konzeptentwurf vorgestellte Beispiel fort. Die Knotenklasse `Land` besitzt einen Anker zur Knotenklasse Stadt, der über die 1-n-Verknüpfung `liegtIn` navigiert. Die Auswahl des angewählten Knotens erfolgt über die Zugangsstruktur `StadtAuswahl`, die alle `Städte` des aktuellen `Landes`, welches in dem Verknüpfungsattribut `aktuellesLand` gespeichert ist, in Form eines alphabetisch nach dem Attribut `Stadt.Name` geordneten *Indexes* beinhaltet. Analog kann von der `Stadt` zu 1-n `StadtBildern` navigiert werden. Hierbei gibt es zwei Möglichkeiten, der Verknüpfung zu folgen:

- über die Zugangsstruktur `BilderTour` oder
- über die Zugangsstruktur `FilmTour`.

Beide Zugangsstrukturen sind vom Typ *Guided Tour*. Der Unterschied zwischen den beiden Zugangsstrukturen wird durch die Definition von entsprechenden Kontexten und Kontextklassen verdeutlicht. Die Knotenklasse `Bild` besitzt zwei Kontextklassen, eine für jeden der Kontexte `StadtBildInBilderTourKontext` und `StadtBildInFilmTourKontext`, die die oben genannten entsprechenden Zugangsstrukturen realisieren. Im erstgenannten Kontext wird das Attribut `Präsentation` auf den Typ `Image`, im zweiten Fall auf den Typ `Movie` abgebildet. Dazu besitzen beide Kontextklassen zusätzliche Anker `nächstesBild` und `letztesBild`, die ein sequentielles Abschreiten der Elemente der Kontexte ermöglichen.

2.4.3.4 Abstrakter Schnittstellenentwurf

In diesem Schritt wird auf eine implementationsunabhängige Weise eine Spezifikation des Erscheinungsbilds der Anwendung, der Beziehungen zwischen Navigations- und Schnittstellenobjekten, der Schnittstellentransformationen und der Synchronisation zwischen Schnittstellenobjekten vorgenommen.

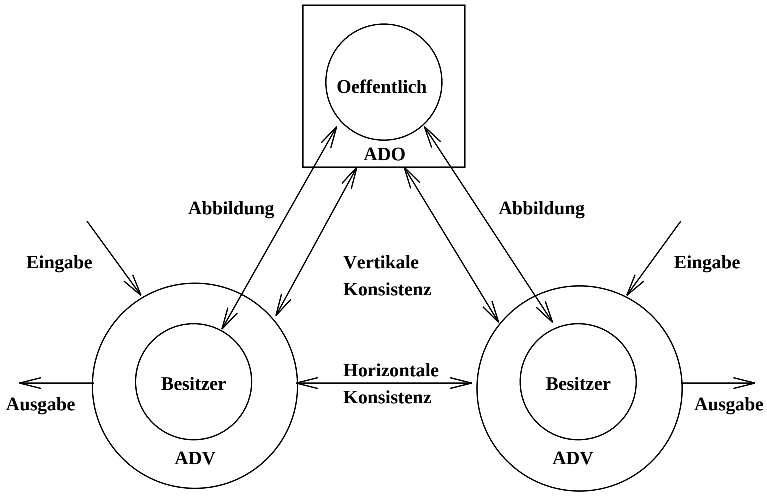


Abbildung 2.17: Beziehung zwischen ADV und ADO

Das Abstrakte Schnittstellenmodell wird durch die folgenden Aktionen konstruiert:

Definition von ADVs für jedes navigationelle Objekt. Die Abkürzung ADV bedeutet *Abstract Data View* [CL95]. ADVs sind Objekte, da sie einen Zustand und eine Schnittstelle besitzen. ADVs sind abstrakt in dem Sinne, daß sie nur die Schnittstelle und den Zustand repräsentieren und nicht die Implementation. In einer typischen Anwendung liegt eine Menge von Abstrakten Datenobjekten (ADOs) vor, die Datenstrukturen und Funktionen innerhalb der Anwendung verwalten, und eine Menge von ADVs, die Schnittstellenaspekte zum Benutzer realisieren. Bei OOHDM fungieren navigationelle Objekte wie Knoten, Verknüpfungen und Zugangsstrukturen als ADOs.

Typischerweise besitzt ein ADO ein oder mehrere ADVs, die für die Außenwelt einen Aspekt seines Zustands repräsentieren. Jede ADV muß zu ihrem besitzenden ADO konsistent sein, genauso wie alle ADVs eines ADOs untereinander. Diese Arten von Kosistenz werden vertikale und horizontale Konsistenz genannt. Abbildung 2.17 zeigt die Beziehung zwischen ADV und ADO. Eine ADV reagiert auf externe Eingaben, indem sie Funktionalität ihres besitzenden ADOs aufruft, somit seinen Zustand verändert und Ausgaben an die Umwelt macht.

Ein ADV wird hier als Schnittstellenobjekt, welches aus einer Menge von Attributen, die seine Präsentationseigenschaften festlegen, und aus einer Menge von Ereignissen, die es behandeln kann, gesehen. Attributswerte können als Konstanten definiert werden, um einzelne Erscheinungsformen wie Position oder Farbe festzulegen. Eine reservierte Variable, *WahrnehmungsKontext*, im folgenden als *WK* abgekürzt, wird benutzt, um Modifikationen des Wahrnehmungsraums durchzuführen. Wenn ein Objekt wahrnehmbar gemacht werden soll, wird es zu *WK* hinzugefügt und umgekehrt. Die ADV wird aus Primitiven, die typische Schnittstellenobjekte wie Buttons, Fenster, Textfelder und andere umfassen, und anderen ADVs modelliert.

Bei OOHDM fungieren Navigationsobjekte wie Knoten, Verknüpfungen und Zugangsstrukturen als ADOs, während die assoziierten ADVs ihr Erscheinungsbild gegenüber dem Benutzer festlegen.

Spezifikation eines Konfigurationsdiagramms. Konfigurationsdiagramme [CHB92] stellen externe benutzerinitiierte Ereignisse, die ein ADV behandelt, die Methoden, die ein ADV bereitstellt, und die Kommunikation zwischen ADVs und ADOs dar. Mit einem Konfigurationsdiagramm werden die statischen Beziehungen zwischen ADO, ADV und Benutzer modelliert. Ein Konfigurationsdiagramm kann auch die Verschachtelungsstruktur von zusammengesetz-

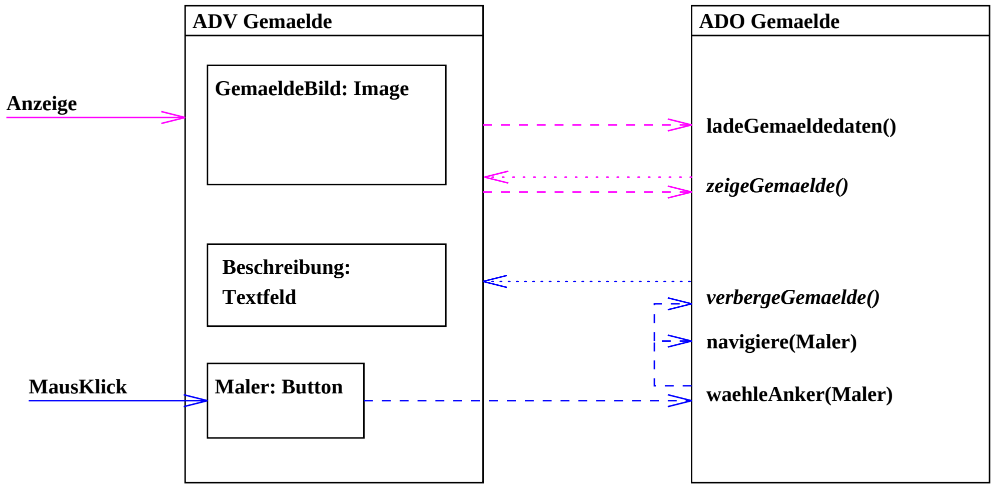


Abbildung 2.18: Beispiel für ein Konfigurationsdiagramm

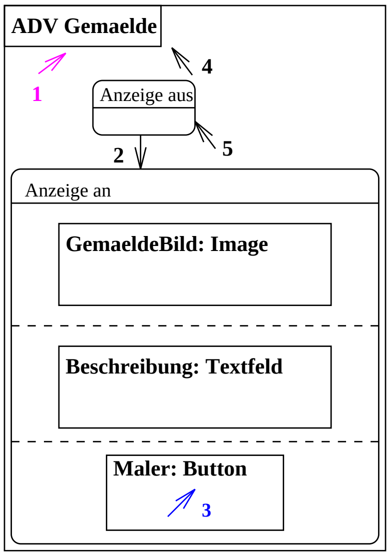
ten ADVs zeigen. Im Rahmen dieser Diplomarbeit werden Methoden. die die Benutzerschnittstelle modifizieren, kursiv geschrieben, um sie von Methoden, die keinen direkten Einfluß auf die Schnittstelle haben, zu unterscheiden. Ein Beispiel für ein Konfigurationsdiagramm findet sich in Abbildung 2.18.

Im Kontext von OOHDM wird mit Konfigurationsdiagrammen die Art festgelegt, auf die der Benutzer mit der Hypermediaapplikation interagiert, insbesondere aber welche Schnittstellenobjekte Navigation auslösen.

Spezifikation eines ADVCharts für jede ADV. ADVCharts [CCCL94] beschreiben Synchronisationsaspekte, die bei Multimediadaten häufig vorkommen und modellieren die Dynamik der Schnittstelle. Wichtig ist hierbei, daß die Attribute und Anker der ADO direkt durch Schnittstellenobjekte dargestellt und die ausgelöste Funktionalität dynamisch durch Transitionsbeschreibungen festgelegt werden. Eine Transition wird definiert, indem ein Ereignis, eventuell existierende Vorbedingungen für dieses Ereignis und eine oder mehrere Nachbedingungen festgelegt werden. Im Unterschied zu Statecharts können für ADVCharts Aktionen als Nachbedingungen definiert werden, die hier als Methoden spezifiziert werden (siehe [RSLC95], Abschnitt 4.2). Ein ADVChart und ihre Transitionen werden beispielhaft in Abbildung 2.19 für eine ADV Gemaelde gezeigt, die aus einem Bild, einer Beschreibung und einem navigationsauslösenden Button, der zum Maler des Bildes führt, besteht. Das Ereignis *Anzeige* bewirkt den Aufruf der Methoden `ladeGemaeldedaten` und `zeigeGemaelde`. Die Methode **zeigeGemaelde** bewirkt die Anzeige der gesamten ADV, was durch die Addition von `Gemaelde` zum Wahrnehmungsraum *WK* dokumentiert wird. Das Ereignis *Mausklick* mit dem Fokus auf dem Button `Maler` bewirkt den Aufruf der Methode `wähleAnker(Maler)`, die die ADV `Gemaelde` durch den Aufruf der Methode `verbergeGemaelde` aus dem Wahrnehmungskontext ausblendet und gleichzeitig Navigation zur ADV `Maler` verursacht.

2.4.3.5 Implementierung

Indem die Navigations- und Abstrakten Schnittstellenobjekte - die wahrnehmbaren Objekte und ihre Transformationen - auf konkrete Objekte einer gewählten Anwendungsumgebung abgebildet werden, macht der Autor das System lauffähig. Tatsächlich kann das durch die Schritte 1-3 konstruierte Modell auf direkte Weise in existierenden Hypermediasystemen wie Hypercard, Toolbook, KMS, Guide, Microcosm, HTML/Java und anderen implementiert werden.



Transitionen

1:
Ereignis: Anzeige
Vorbedingungen:
Nachbedingungen: ladeGemaeldedaten(); *zeigeGemaelde()[2]*

2:
Ereignis: zeigeGemaelde()
Vorbedingungen:
Nachbedingungen: WK = WK + Gemaelde

3:
Ereignis: Mausklick
Vorbedingungen: Fokus(Maler)
Nachbedingungen: waehleAnker(Maler)

4:
Ereignis: waehleAnker(Maler)
Vorbedingungen:
Nachbedingungen: *verbergeGemaelde()[5];* navigiere(Maler)

5:
Ereignis: verbergeGemaelde()
Vorbedingungen:
Nachbedingungen: WK = WK - Gemaelde;

Abbildung 2.19: Beispiel für eine ADVChart

2.4.4 Bewertung der Methoden

In diesem Kapitel werden die drei vorgestellten Methoden zur Entwicklung von Hypermediasystemen bewertet und verglichen. Eine Bewertung von OOHDM im praktischen Einsatz folgt in Kapitel 5.

2.4.4.1 Richtlinien nach Balasubramanian

Balasubramanian formuliert Richtlinien, die zwar wertvoll und nützlich für den Entwurf von Hypermediasystemen sind, aber in ihrer Gesamtheit kein formales Modell konstruieren, sondern informelle Konstruktionshinweise liefern. Viele der von dem Autor genannten Punkte finden sich allerdings in den verschiedenen Entwicklungsphasen der anderen beiden Methoden wieder.

2.4.4.2 RMM und OOHDM

RMM und OOHDM erheben beide den Anspruch, einen mehrstufigen formalen Prozeß zum Entwurf von Hypermediasystemen bereitzustellen. Beide sind in Phasen unterteilt, in denen jeweils ein Modell, welches einen Aspekt des zu konstruierenden Systems repräsentiert, spezifiziert wird. Beide Methoden haben denselben Nachteil, daß sie noch nicht einheitlich und zusammenfassend dokumentiert sind und nicht alle Aspekte der Konstruktion ausführlich und zusammenhängend erläutern. Ferner gibt es noch keine Entwicklungswerkzeuge, die die Methoden durchgängig unterstützen.

Beide Methoden decken ähnliche Bereiche des Softwareentwicklungsprozesses ab. Es findet zunächst ein Entwurf eines allgemeinen Informationsbereichs statt. Dann wird zu diesem Modell ein Navigationsmodell entworfen, daß die Navigationsmöglichkeiten in dem spezifizierten Informationsraum festlegt. Als letztes wird die Benutzerschnittstelle spezifiziert.

Die folgende Tabelle vergleicht und bewertet die beiden Methoden anhand der Umsetzung der drei oben genannten Aspekte Informationsbereichs-, Navigations- und Schnittstellenentwurf.

	RMM		OOHDM	
Entwurf des Informationsbereichs	ER-Entwurf	+	Konzeptentwurf	+
	Scheibenentwurf	+		
Navigationsentwurf	Navigationsentwurf	+	Navigationsentwurf	**0**
Schnittstellenentwurf	Konversionsprotokollentwurf	-	Abstrakter Schnittstellenentwurf	+
	Benutzerschnittstellenentwurf	-		
	Laufzeitverhaltenentwurf	-		
Implementation	Implementation		Implementation	

Abbildung 2.20: Vergleich und Bewertung von RMM und OOHDM

Entwurf des Informationsbereichs: Beide Methoden benutzen hier etablierte Methoden und Konstrukte zur Darstellung des Informationsbereichs: aus Objekten, Klassen, Entitäten und Beziehungen zusammengesetzte Diagramme. Dieser Ansatz ist gut als Grundlage für das zu konstruierende Hypermediasystem geeignet. Die Phase *Konzeptentwurf* von OOHDM und die Phasen *ER-Entwurf* und *Ausschnittentwurf* von RMM decken diesen Bereich ab, wobei letztere schon Aspekte des Navigationsentwurfs umfaßt. Wie oben schon erwähnt, gehen beide Methoden nicht explizit auf die Definition von Funktionalität ein.

Navigationsentwurf: Beide Methoden verwenden Erweiterungen der im Informationsbereichsentwurf benutzen Konstrukte, um Navigation darzustellen. Die wichtigste Erweiterung ist das Hinzufügen von Zugangsstrukturen und bei OOHDM navigationellen Kontexten, die Auswahlmöglichkeiten und Navigationsregeln für das System reflektieren. RMM hat hier den Vorteil einer einheitlichen und gut dokumentierten Notation, während OOHDM die komplexen Zusammenhänge zwischen Knotenklassen, Verknüpfungen, Zugangsstrukturen und navigationellen Kontexten etwas unübersichtlich darstellt. Allerdings verwirklicht OOHDM durch die Benutzung von zusammengesetzten Klassen und navigationellen Kontexten die durch das *Amsterdam Hypermedia Model* geforderte Behandlung von Verknüpfungskontexten. Die Phase *Navigationsentwurf* von OOHDM sowie die Phasen *Ausschnittentwurf* (teilweise) und *Navigationsentwurf* von RMM realisieren diesen Aspekt.

Schnittstellenentwurf: Hier hat RMM den großen Nachteil, keine formale Notation zur Beschreibung von Schnittstellenobjekten und Transformationen bereitzustellen. OOHDM deckt diesen Aspekt sehr umfassend durch seine Verwendung von ADVs, ADVCharts und Konfigurationsdiagrammen ab. Die Umsetzung des Schnittstellenentwurfs in OOHDM realisiert den durch das *Amsterdam Hypermedia Model* geforderten Aspekt der Synchronisation durch diese formellen Beschreibungen. Die Phase *Abstrakter Schnittstellenentwurf* von OOHDM und die Phasen *Konversionsprotokollentwurf, Benutzerschnittstellenentwurf* und *Laufzeitverhaltenentwurf* (teilweise) von RMM behandeln diesen Bereich.

2.4.4.3 Zusammenfassung

OOHDM unterscheidet sich von RMM in zwei wesentlichen Punkten. OOHDM betont erstens den Navigations- und Schnittstellenentwurf. Zweitens benutzt OOHDM während des ganzen Entwurfsprozesses Objekte als Konstrukte und fördert somit die Wiederverwendung von existierenden Komponenten.

Weiterhin realisiert OOHDM die wesentlichen in Kapitel 2.2.3.2 formulierten Anforderungen an Hypermediabeschreibungen:

Temporäre Information. OOHDM spezifiziert Synchronisation zwischen Komponenten während des Abstrakten Schnittstellenentwurfs durch die Verwendung von ADVs (siehe Kapitel 2.4.3.4). Innerhalb jedes Knotens wird durch explizite Festlegung der Transitionen zwischen Zuständen der ADVs und Primitive die gesamte Synchronisation für einen Knoten definiert.

Verknüpfungen. OOHDM behandelt durch seinen objektorientierten Ansatz Verknüpfungen genau wie Knoten als Objekte, die eigene Attribute und Methoden besitzen. Verknüpfungen innerhalb einer Komponente lassen sich in den Phasen Konzeptentwurf und Navigationsentwurf durch Aggregation beschreiben, die während der Festlegung der navigationellen Transformationen im Navigationsentwurf und während des Abstrakten Schnittstellenentwurfs als Oberflächentransformationen spezifiziert werden. Weiterhin ermöglicht es OOHDM durch seine Eigenschaft, die während des Navigationsentwurfs definierten Knoten aus Attributen mehrerer Konzeptklassen zusammensetzen zu können und durch die Möglichkeit, Knoten zu navigationellen Kontexten mit gleichen Navigationsregeln für alle Elemente zu definieren, explizit festzulegen, welche Informationen bei Navigation zwischen Knoten beibehalten werden.

Präsentationsspezifikationsattribute höherer Ordnung. Die Phase des Abstrakten Schnittstellenentwurfs von OOHDM ermöglicht es, Präsentationsattribute für ADVs festzulegen, die ganze Knoten, Teilkomponenten oder Oberflächenprimitive definieren können. Die objektorientierte Verschachtelung von ADVs ist hier sehr hilfreich.

RMM und OOHDM sind im Prinzip beide gut geeignet, um ein formales Modell einer Hypermediaanwendung zu konstruieren, wobei OOHDM den entscheidenden Vorteil bei seiner Behandlung des Schnittstellenentwurfs hat. Insofern hat es sich als sinnvoll erwiesen, daß die Methode OOHDM zur Entwicklung der hypermedialen Patientenakte vorgegeben und eingesetzt wurde. Allerdings sollten beide Methoden zusammenfassend dokumentiert werden, um Mißverständnisse, Unklarheiten und Widersprüche, die teilweise noch vorhanden sind, auszuräumen.

Kapitel 3

Anforderungen

In diesem Kapitel werden die im Rahmen dieser Diplomarbeit ermittelten fachlichen und DV-technischen Anforderungen vorgestellt, die das zu realisierende System, die hypermediale Patientenakte, erfüllen muß. Diese Anforderungen basieren auf den in Kapitel 2.1.2.1 ermittelten Eigenschaften einer computerunterstützten Patientenakte. Die Anforderungen sind den von Xenos und Christodoulakis in [MX95] vorgeschlagenen 11 Qualitätsanforderungen an computerunterstützte Patientenakten zugeordnet.

Akzeptanz

Die Akzeptanz der hypermedialen Patientenakte durch die Benutzer, also Ärzte, Pflegepersonal und Verwaltung, wird dadurch erhöht, daß folgende Anforderungen bei der Konstruktion des Systems berücksichtigt werden:

Anwenderfreundliche Benutzeroberflächengestaltung. Diese fachliche Anforderung beinhaltet mehrere zu berücksichtigende Unterpunkte.

- *Wiedererkennung.* Aus Gründen der Gewohnheit und Wiedererkennung ist es sinnvoll, die Benutzeroberfläche einer computerunterstützten Patientenakte an dem Aussehen einer papiernen Patientenakte auszurichten. Möglichkeiten hierzu sind die grafische Darstellung von farbigen Reitern, die in der konventionellen Patientenakte als Signale dienen und die Beibehaltung des Layouts von existierenden Papierformularen bei der Realisierung in der computerunterstützten Patientenakte.
- *Ergonomie und Präsentation.* Es ist wichtig, die Benutzeroberfläche des gesamten Systems nach ergonomischen Richtlinien zu entwerfen. Dieses umfaßt unter anderem die Verwendung von Metaphern wie einer Schreibtisch- oder Patientenaktenmetapher, die Benutzung von aufeinander abgestimmten Bildschirmfarben sowie eine übersichtliche Einteilung des Bildschirms und eine sinnvolle Gruppierung von Informationen. Weiterhin ist dafür zu sorgen, daß die Informationen der Patientenakte stets aktuell und multimedial präsentiert werden. Ferner macht eine geeignete Verknüpfung der Informationseinheiten, wie sie in einem Hypermediasystem vorliegt, das System für den Benutzer bedienungsfreundlicher. Die Verknüpfung der Informationseinheiten kann an den Arbeitsabläufen der Benutzer orientiert sein.
- *Dateneingabe und Datenausgabe.* Eine weitere Anforderung zur Benutzeroberfläche fordert verschiedene Datenein- und Datenausgabemöglichkeiten. Neben der Tastatureingabe, die aber aus Platz- und Zeitgründen in der Krankenhausumgebung durchaus ungeeignet sein kann, müssen Alternativmethoden betrachtet werden. Möglichkeiten dafür sind mausgesteuerte Auswahlmenüs, Spracheingabe und direkte Eingabe durch elektronische Überwachungsgeräte. Der hier relevante Vorteil einer hypermedialen Umsetzung

der Patientenakte liegt darin, daß sie im Idealfall in der Lage ist, eine Vielfalt von verschiedenen Daten, aus denen sich die papierne Patientenakte zusammensetzt, gleichzeitig auf einem Bildschirm ausgeben zu können. Diese Daten bestehen aus Texten (formell, informell, Formular), Bildern (Röntgen, Ultraschall), Filmen (Röntgenfilme), Signalen (EKG, Puls) und Ton (gesprochene Diagnose, EKG). Eine hypermediale Patientenakte muß in der Lage sein, in vernünftigen Maßen mehrere dieser Datentypen gleichzeitig und ohne Zeitverzögerung (zwecks Effektivität) auf einem Bildschirm darstellen zu können und eine Bearbeitung dieser zu ermöglichen.

- *Hilfesystem und Nachschlagewerk.* Eine Anforderung, die sich besonders gut durch den Einsatz von Hypermediakonzepten realisieren läßt, ist das Bedürfnis nach einem umfassenden und einfach zu bedienenden Hilfesystem. Das Hilfesystem einer hypermedialen Patientenakte soll in der Lage sein, einerseits dem Benutzer schnell und lesbar kontextsensitive Hilfe zur Verfügung zu stellen, beispielsweise durch Einblendung eines Hilfe-Fensters durch Tastendruck, andererseits muß es auch als Online-Nachschlagewerk in Form eines Lexikons mit Querverbindungen, die durch Hypertextverknüpfungen realisiert werden, dienen können. Ferner ist dem Phänomen des 'Lost in Hyperspace' durch Navigationshilfen wie Karten zu begegnen. Weiterhin kann das System sowohl dem Arzt als auch dem Pflegepersonal als Online-Nachschlagewerk bei rechtlichen und fachlichen Fragen dienen. Eine Anbindung an Online-Versionen von Dienstvorschriften, Enzyklopädien und rechtlichen Unterlagen, die als Hypermediaanwendungen vorliegen, ist eine wünschenswerte Arbeisterleichterung für das Personal.

Verwendung der Terminologie des Anwendungsbereichs. Dies ist eine fachliche Anforderung. Im Krankenhausbetrieb existiert, wie in vielen anderen spezialisierten Bereichen des öffentlichen Lebens, eine spezialisierte Fachterminologie. Bei der Konstruktion einer computerunterstützten Patientenakte muß darauf geachtet werden, daß diese Terminologie bei der Systemein- und ausgabe genutzt wird, um die Integration des Systems in den alltäglichen Betrieb zu erleichtern. Leider ist es im Moment so, daß es noch keine standardisierte Einheitsterminologie in diesem Bereich gibt, obwohl zahlreiche Bestrebungen in diese Richtung zielen. Die Entwicklung von immer mehr und besseren Anwendungen für diesen Bereich unterstützt die Entwicklung einer einheitlichen Terminologie.

Anwenderorientierte Umsetzung des Ist-Zustands. Dies ist eine fachliche Anforderung an den Entwicklungsprozeß. Um die administrativen Tätigkeiten des Personals effizient zu unterstützen, müssen diese zunächst analysiert werden. Dies geschieht dadurch, daß die Informatiker die Benutzergruppen in die Entwicklung mit einbeziehen, indem sie die alltäglichen Arbeitsabläufe analysieren. Wo es sinnvoll ist, übernehmen die Informatiker existierende Arbeitsabläufe und setzen sie direkt oder unter Berücksichtigung der gewählten Metapher in dem System um.

Vereinfachung und Automatisierung von Routinetätigkeiten. Eine automatisierte Routinetätigkeit, die nach sorgfältiger Analyse in dem System umgesetzt wurde, erleichtert dem Personal die tägliche Arbeit und führt zu einer Verbesserung der Akzeptanz und Steigerung der Effizienz. Diese fachliche Anforderung wird ausführlicher unter *Effektivität* vorgestellt.

Zeitliche Effektivität. Ein großes Problem, vor dem eine Implementierung einer hypermedialen Patientenakte steht, ist die technische Anforderung nach zeitlicher Effektivität. Dieser Punkt wird ebenfalls unter *Effektivität* vorgestellt.

Zugänglichkeit

Eine computerunterstützte Patientenakte wird zugänglich, wenn sie einen einfachen und schnellen Zugriff auf alle vorhandenen Daten ermöglicht und über eine einleuchtende und verständliche Benutzerführung verfügt. Dies wird durch die unter *Akzeptanz* vorgestellten fachlichen Anforderungen nach einer anwenderfreundlichen Benutzeroberflächengestaltung und nach Verwendung der

Terminologie des Anwendungsbereichs sowie die durch die unter *Effektivität* vorgestellte technische Anforderung nach zeitlicher Effektivität erreicht. Eine weitere Anforderung, die diesen Punkt fördert, ist die technische Anforderung nach

Transparenz. Den Endbenutzer, sei er nun Arzt, Pfleger oder Bürokraft in der Krankenhausverwaltung, muß nicht interessieren, auf welchem Rechner die Daten gespeichert sind, die er gerade bearbeitet. Das System sollte so gestaltet sein, daß Aspekte der Lokalisierung von Daten den Benutzer nicht belasten. Eine vernünftige Netzwerklösung auf Basis etablierter Systeme lösen dieses Problem.

Verwaltbarkeit

Die Verwaltbarkeit der Patientenakte wird verbessert, indem die Organisation der Arbeit, bei der die Akte genutzt wird, durch das System unterstützt wird. Dies geschieht durch die unter *Akzeptanz* vorgestellten fachlichen Anforderungen nach einer anwenderfreundlichen Benutzeroberflächengestaltung und nach Verwendung der Terminologie des Anwendungsbereichs, durch die unter *Effektivität* vorgestellte fachliche Anforderung nach Vereinfachung und Automatisierung von Routinetätigkeiten sowie die ebenfalls unter *Effektivität* präsentierte technische Anforderung nach zeitlicher Effektivität. Weitere Anforderungen, die die Verwaltbarkeit unterstützen, sind:

Unterstützung der Terminplanung. Dies ist eine fachliche Anforderung. Die Behandlung eines Patienten in einem modernen Krankenhaus ist in der Regel sorgfältig geplant und durchorganisiert. Damit diese reibungslos durchgeführt werden kann, muß eine angemessene Terminplanung stattfinden. Eine sinnvolle Unterstützung der Terminvereinbarung zwischen verschiedenen Stationen und Abteilungen durch das System ist zudem eine große Arbeitserleichterung. Hier kann eine netzwerkbasierte computerunterstützte Patientenakte wertvolle Hilfe leisten. Dies kann in Form einer durch das System verwalteten Terminplanung, die jederzeit von Hand korrigiert werden kann, geschehen.

Administrative Unterstützung. Eine der wichtigsten fachlichen Anforderungen an ein Krankenhausinformationssystem ist die konkrete Unterstützung des Personals bei seinen Tätigkeiten. Dies soll im Falle dieses Systems dadurch geschehen, daß es durch Unterstützung von administrativen Tätigkeiten des Personals den Zeitaufwand dafür verkürzt und so dem Personal mehr Zeit für eigentliche pflegerische Tätigkeiten bereitstellt. Konkret bedeutet dies hier, die Dokumentation und Anforderungserstellung zu erleichtern.

Direkte Berechnung der finanziellen Aspekte des Krankenhausaufenthalts. Diese fachliche Anforderung bezieht sich auf die Berechnung der individuellen Leistungen für einzelne Patienten, die Berechnung für einzelne Stationen und die Jahreskalkulation des Krankenhauses im Ganzen. Durch lückenlose Dokumentation des gesamten Pflegeprozesses eines Patienten unter Einbeziehung der Fallpauschalen wird die Erstellung einer persönlichen Abrechnung und einer Gesamtkalkulation durch die Verwaltung wesentlich erleichtert. Eine durch die hypermediale Umsetzung unterstützte flexible Informationsverkettung der verschiedenen Abrechnungsaspekte ist hier hilfreich.

Unterstützung von Forschung und Statistik. Dies ist eine fachliche Anforderung. Es ist wünschenswert, sowohl Forschung als auch Statistik durch Anbindung der computerunterstützten Patientenakte an statistische Analyseprogramme zu unterstützen. Durch eine offene Gestaltung der hypermedialen Patientenakte werden hierfür die Voraussetzungen geschaffen (siehe auch Modularität und Offenheit unter *Ausdehnbarkeit und Flexibilität*).

Integration von externen Anwendungen. Dies ist eine fachliche Anforderung. Da sich in der Krankenhausumgebung im Allgemeinen schon Anwendungen verschiedenster Couleur im Einsatz befinden, ist es wünschenswert, diese Anwendungen direkt oder deren Dateiformate im Rahmen der hypermedialen Patientenakte bearbeiten und aufrufen zu können. Ein offenes

Hypermediasystem sollte in der Lage sein, diese verschiedenen Anwendungen entweder direkt miteinander zu verknüpfen oder zumindest einen Datenaustausch zwischen diesen zu ermöglichen (siehe auch Modularität und Offenheit unter *Ausdehnbarkeit und Flexibilität*).

Umfassende Patienteninformation. Durch eine Umsetzung der fachlichen Anforderung nach übersichtlicher und einheitlicher Organisation der Patientendaten ist der Anwender ständig über den gesamten Zustand eines Patienten informiert. Im Gegensatz zu einer papiernen Akte, deren Bestandteile oft in verschiedenen Formaten vorliegen und häufig im Krankenhaus verteilt sind, ermöglicht es eine nach etablierten, ergonomischen Richtlinien entworfene hypermediale Patientenakte (siehe anwenderfreundliche Benutzeroberflächengestaltung unter *Akzeptanz*), dem Personal alle relevanten Daten eines Patienten anschaulich zu präsentieren.

Adäquate Datenspeicherung. Dies ist eine technische Anforderung. Vor allem Multimediadaten verbrauchen sehr viel Speicherplatz. Aus Effizienz- und Kostengründen muß sichergestellt werden, daß insbesondere die Daten, die wirklich wichtig und aktuell sind, schnell aufrufbar sind. Das bedeutet eine vernünftige Archivierung von Patientendaten, die zur Zeit nicht für eine Behandlung benötigt werden. Konkret heißt das, daß Daten von Patienten, die sich gerade zur Behandlung im Krankenhaus befinden, online abrufbar sind, während die Daten von anderen Patienten auf sinnvolle und geschützte Weise archiviert werden.

Vertraulichkeit

Vertraulichkeit der in der Patientenakte organisierten Daten wird erreicht, indem sowohl umfassende Datenschutzmaßnahmen als auch eindeutige Zuordnungen von Maßnahmen zu Benutzern in dem System realisiert werden. Das geschieht durch Berücksichtigung der unter *Verwaltbarkeit* aufgeführten technischen Anforderung nach adäquater Datenspeicherung sowie der folgenden Anforderungen:

Datenschutz. Ein wichtiges und aktuelles Thema, nicht nur im Krankenhaus, ist die fachliche Anforderung nach Datenschutz. Um die Sicherheit und Integrität der durch das System gespeicherten Patientendaten zu gewährleisten und um von Ärzten und Pflegepersonal durchgeführte Maßnahmen eindeutig zuordnen zu können, ist es vonnöten, daß das System über adäquate Datensicherungs- und Zugriffsschutzmechanismen verfügt. Auch ist es nötig, Sicherheitskopien des Datenbestands regelmäßig und periodisch durchzuführen, um Datenverluste zu vermeiden. Beispiele hierfür sind die Einrichtung von Logins für verschiedene Benutzer und Benutzergruppen mit der Möglichkeit eines schnellen Benutzerwechsels, Systemidentifikation durch Keycards, Datenverschlüsselung bei Speicherung, häufige und mehrfache Backups des Systemdatenbestands bei sorgfältiger Aufbewahrung der Datenträger und Auswahl einer sicheren Plattform für das System, in dem Sinne, daß sie als verläßlich gegen Systemabstürze eingestuft wird.

Sichten. Diese fachliche Anforderung berücksichtigt, daß verschiedene Benutzer der Patientenakte verschiedene Bedürfnisse und Aufgaben haben. Die verschiedenen Benutzergruppen im Krankenhaus (Ärzte, Pflegepersonal, Verwaltung) sollen nur die Daten, die für die Ausübung ihrer täglichen Tätigkeiten wichtig sind, sehen. Das bedeutet für die zu konstruierende Patientenakte, daß verschiedene Sichten auf das System, für jede Benutzergruppe eine, implementiert werden müssen.

Legale Dokumentation. Dies ist eine fachliche Anforderung. Für jede bei der Behandlung eines Patienten durchgeführte Maßnahme existiert ein Verantwortlicher. Es muß im Rahmen einer computerunterstützten Dokumentation sichergestellt werden, daß jede eingetragene Maßnahme durch eine Benutzeridentifikation abgezeichnet ist. Dies kann automatisch durch einen Systemvermerk anhand des Logins geschehen.

Sicherheit. Nicht zu vergessen ist die technische Anforderung nach Sicherheit. Eine Implementierung der computerunterstützten Patientenakte auf der Basis eines verteilten Systems (siehe auch Verteiltheit und Dezentralisation unter *Ausdehnbarkeit und Flexibilität*) hat den Vorteil, daß durch die Tatsache, daß sich einerseits die wichtigen Daten, andererseits die Rechner selbst an physisch verschiedenen Orten befinden, eine größere Sicherheit als bei einem zentralen System erreicht wird. So kann ein Ausfall einer Teilkomponente, sei es nun ein Datenspeicher oder ein Terminalrechner, leicht kompensiert werden. Voraussetzung ist hier im Fall der Datenspeicher eine sorgfältige Archivierung und Aufbewahrung von Sicherheitskopien.

Effektivität

Ein System, welches eine computerunterstützte Patientenakte realisiert, ist dann als effizient zu bezeichnen, wenn es für das Personal eine spürbare Arbeitserleichterung bewirkt. Dies kann durch Umsetzung der unter *Akzeptanz* erwähnten, der unter *Verwaltbarkeit* beschriebenen Unterstützung der Terminplanung sowie der folgenden Anforderungen erreicht werden:

Vereinfachung und Automatisierung von Routinetätigkeiten. Diese fachliche Anforderung beinhaltet die folgenden Unterpunkte:

- *Pflegeplanung und -dokumentation.* In einem modernen Krankenhaus ist der gesamte Pflegeprozeß vollständig organisiert und dokumentiert. Eine computerunterstützte Patientenakte muß dieser Tatsache Rechnung tragen. Das System soll den gesamten Pflegeprozess elektronisch dokumentieren. Weiterhin soll das System die Pflegeplanung durch übersichtliche Benutzerführung und ein Bausteinsystem, bei dem die Auswahl und Eintragung von Standardmaßnahmen einfach möglich ist, unterstützen.
- *Diagnoseunterstützung und Therapieplanungssystem.* Ein weiterer wichtiger Punkt ist die Möglichkeit der hypermedialen Patientenakte, dem Arzt eine schnelle und effiziente Unterstützung bei der Diagnose und Therapie zu liefern. Dies kann auf ähnliche Weise wie bei der Pflegeplanung geschehen. Eine weitere Möglichkeit, den Arzt zu unterstützen, kann durch die schnelle, übersichtliche und unkomplizierte Bereitstellung von großen Datenmengen, die hypermedial präsentiert werden, geschehen. Letzteres kann beispielsweise durch eine direkte Anbindung an das World Wide Web umgesetzt werden.
- *Minimierung der Patientenverweildauer.* Es liegt sowohl im Interesse des Patienten als auch der Krankenhausverwaltung, daß die Verweildauer eines Patienten im Krankenhaus nicht länger als nötig ist. Daher ist es vonnöten, daß der Aufenthalt eines Patienten dort optimal geplant wird und verläuft. Ein effizientes System unterstützt die effiziente Planung und Durchführung des Aufenthalts durch die obigen Punkte Pflegeplanung und Diagnose-/Therapieunterstützung.
- *Verbesserte Patientenpflege.* Der Patient soll von der Tatsache profitieren, daß durch eine effiziente computerunterstützte Patientenakte mehr Zeit des Personals zur Verfügung steht.

Mehrbenutzersystem. Dies ist eine fachliche Anforderung. Die computerunterstützte Patientenakte soll in einer Krankenhausumgebung eingesetzt werden. Das bedeutet, daß viele Personen, sowohl Pflege- als auch Verwaltungspersonal, gleichzeitig auf die elektronisch gespeicherten Informationen der Patientenakte zugreifen müssen. Für das zu konstruierende System bedeutet dies, daß es in der Lage sein muß, mehreren Benutzern gleichzeitig den Zugriff auf die Patientenakte zu ermöglichen, und zwar mit angemessen schneller Reaktionszeit. Dies kann durch eine gute Netzwerkrealisierung erreicht werden.

Kommunikationsförderung. Eine wesentliche fachliche Anforderung, die eine netzwerkfähige computerunterstützte Patientenakte erfüllen kann, ist die nach Verbesserung der Kommunikation im Krankenhaus. Durch Bereitstellung eines E-Mail-Systems mit Anbindung an globale Kommunikationsstrukturen wie dem Internet wird nicht nur die Kommunikation zwischen einzelnen Personen und Abteilungen des Krankenhauses unterstützt, sondern es ergeben sich auch vor allem für Ärzte und Forschungspersonal neue Möglichkeiten der weltweiten Verständigung mit anderen Kollegen.

Erhöhung der Wirtschaftlichkeit. Eine der fachlichen Anforderungen an das System ist es, die Produktivität von Ärzten und Pflegepersonal zu steigern, indem diese bei administrativen Aufgaben durch das System unterstützt werden (siehe Administrative Unterstützung unter *Verwaltbarkeit*). Die Verwaltung unterstützt diese Forderung aus dem Grund, weil dadurch die Wirtschaftlichkeit des Krankenhauses verbessert wird und Ressourcen und Arbeitszeit besser eingesetzt werden.

Zeitliche Effektivität. Ein großes Problem, vor das sich eine Implementierung einer hypermedialen Patientenakte gestellt sieht, ist die tatsächliche technische Anforderung nach zeitlicher Effektivität. Die Zeit von Ärzten und Pflegepersonal ist in der Regel knapp bemessen. Aus diesem Grund darf ein System, welches das Personal effektiv bei seiner Arbeit unterstützen soll, den Benutzern keinen Nettozeitverlust durch umständliche Ein- und Ausgaben bescheren. Ein Szenario des täglichen Arbeitslebens, bei dem dieser Punkt akut wird, ist die Visite. Hier müssen Schwestern und Ärzte in der Lage sein, schnell und einfach die aktuellen Patientendaten abzurufen und neue Anweisungen des Arztes zu dokumentieren. Eine Lösung für dieses Problem ist die Nutzung von Notebooks oder PDAs (Personal Digital Assistants), die das Pflegepersonal bei der Visite anstelle der alten papiernen Patientenakte mitnimmt. Natürlich muß hier besonders auf Datenintegrität geachtet werden. Dies kann durch ständigen Kontakt (Funk, Infrarot, etc.) geschehen.

Direkte Überwachungsdatenspeicherung und -auswertung. Eine zu berücksichtigende technische Anforderung ist die direkte Speicherung und Auswertung von Patientenüberwachungsdaten. Vitalwerte und andere Werte (Ergebnisse von Laboruntersuchungen) müssen nicht mehr von Hand eingegeben werden, sondern können direkt von der Maschine ins System geleitet und dort analysiert, ausgewertet und in Datenbanken gespeichert werden. Dadurch entfallen längere Wartezeiten, die im Moment noch in Kauf genommen werden müssen, bis Ergebnisse von Untersuchungen auf der Station eintreffen.

Erweiterbarkeit und Flexibilität

Diese beiden Kriterien werden erfüllt, indem das System nach etablierten Softwareentwicklungsmethoden konzipiert und entwickelt wird, die diese Punkte berücksichtigen. Die folgenden technischen Anforderungen sowie die unter *Portierbarkeit und Universalität* erwähnte Anforderung nach Plattformunabhängigkeit unterstützen diesen Punkt:

Verteiltheit und Dezentralisation. Das System soll verteilt und dezentralisiert sein. Allein durch die Natur einer hypermedialen Patientenakte, die eine großräumige Verteiltheit impliziert (Terminals auf den Stationen, in der Verwaltung, Aufnahme, Labors, etc.), muß eine verteilte und dezentralisierte Lösung konstruiert werden. Durch die immer größer werdende Leistungsfähigkeit von Einzelplatzsystemen ist es heute möglich, angemessen multimediafähige Rechner, die über Netzwerke kommunizieren, als optimale Lösung für eine hypermediale Patientenakte anzusehen. Eine Netzwerklösung, die ihre interne Kommunikation über Standardschnittstellen (beispielsweise HL7 [Pac94]) abwickelt, ist außerdem in der Lage, über dieselben Standardschnittstellen extern zu kommunizieren. Dieses erfordert zudem, daß die Anbindungen an Datenbanken, die natürlich auch Multimediadaten beinhalten können, und andere Krankenhäuser realisiert werden.

Modularität und Offenheit. Eine weitere wichtige Anforderung ist die Konstruktion des Systems als modulares und offenes System. Die Vorteile eines modularen Systems, das über Standardschnittstellen kommuniziert, liegen in einer verbesserten Wartbarkeit, d.h. die Erweiterung, Korrektur und Weiterentwicklung des Systems wird unterstützt.

Integrität

Ein wichtiger Punkt vor allem in verteilten Systemen ist die Datenintegrität. Die folgende technische Anforderung berücksichtigt dies:

Konsistenz. Es muß bei der Konstruktion der hypermedialen Patientenakte sichergestellt werden, daß die Konsistenz der Daten erhalten bleibt, also keine zwei Schreibzugriffe gleichzeitig erfolgen. Dies geschieht durch eine vernünftige Netzwerkverwaltung (siehe Verteiltheit und Dezentralisation unter *Ausdehnbarkeit und Flexibilität*). Auch muß daß System ständig von verschiedenen Stellen eingehende neue Daten sofort integrieren und verwalten. Dies geschieht durch Speicherung in Datenbanken, mit denen über Standardschnittstellen kommuniziert wird.

Portierbarkeit und Universalität

Das System wird portierbar, indem es nach einer Entwicklungsmethode konstruiert wird, bei der die eigentliche Implementierung erst im allerletzten Entwicklungsschritt geschieht. Die unter *Ausdehnbarkeit und Flexibilität* erwähnte Anforderung nach Modularität unterstützt dies ebenso wie die technische Anforderung nach

Plattformunabhängigkeit. Das System muß plattformunabhängig konzipiert werden. Moderne Softwareentwicklungsmethoden berücksichtigen dies in der Regel dadurch, daß der Entwicklungsschritt, in dem das System tatsächlich implementiert wird, als letzter erfolgt. Das heißt, daß das System zunächst als abstraktes Modell fertiggestellt wird, so daß es letztendlich nur noch auf verschiedene Plattformen portiert werden muß. Dies wird durch die Auswahl einer geeigneten Entwicklungsmethode gewährleistet.

Nützlichkeit

Die hypermediale Patientenakte wird dann als nützlich angesehen, wenn sie eine Effizienzsteigerung und Kosteneinsparung bewirkt. Wie eine Effizienzsteigerung erreicht wird, wurde schon unter dem entsprechenden Punkt behandelt. Eine Kostenreduktion wird dadurch erreicht, indem Verwaltungsaufwand verringert wird und Ressourcen besser verwaltet werden. Die Unterstützung der Buchführung und Verwaltung durch das System sorgen für eine bessere Verteilung und Nutzung von Ressourcen.

Lesbarkeit

Die Daten, die durch die hypermediale Patientenakte dargestellt werden, werden lesbarer, wenn sie übersichtlich dargestellt werden und der Benutzer einen Einfluß auf die Art der Darstellung hat. Die Benutzeroberfläche, auf die unter *Akzeptanz* schon näher eingegangen wurde, ist dafür verantwortlich.

Kapitel 4

Entwicklungsprozeß

In diesem Kapitel werden in Abschnitt 4.1 zunächst die Grundlagen des durchgeführten Entwicklungsprozesses, die überarbeiteten Problembereichsobjekte, Anwendungsfälle und Akteure der Spezifikation der Projektgruppe 263 *PROMETHEUS* vorgestellt. Abschnitt 4.2 behandelt das entworfene Konzeptmodell. Hier werden die Subsysteme dieses Modells präsentiert und die Klasse `Patient` sowie die Beziehung `AnforderungPatient` beispielhaft vorgestellt. Der nächste Abschnitt 4.3 befaßt sich mit dem Navigationsmodell. Dieses wird allgemein anhand einer Beschreibung der spezifizierten Sichten und speziell durch konkrete Beispiele in Form der Navigationsklasse `Patient`, der Verknüpfung `AnforderungPatient`, der Zugangsstruktur `PatientenanforderungsAuswahl`, des navigationellen Kontexts `Patientenanforderungskontext`, der Kontextklasse `Patient im Neuaufnahmekontext` und letztlich der navigationellen Transformationen der Navigationsklasse `Patient` präsentiert. In Abschnitt 4.4 wird die verwendete Notation zur Beschreibung der Benutzerschnittstellentransformationen vorgestellt und anhand zweier Beispiele, der ADVs `Patient` und `Navigationsleiste Patient` erläutert. Der letzte Abschnitt 4.5 behandelt Aspekte der Umsetzung der bisher konstruierten Modelle in eine Implementation. Zunächst wird die gewählte Vorgehensweise, eine Implementierung auf Basis von Java-Applets im Intranet-Rahmen beschrieben, dann wird auf Aspekte der Datenspeicherung durch Nutzung der JDBC-Klassen eingegangen und zuletzt wird der Aspekt der Integration von externen Anwendungen betrachtet.

4.1 Basis des Entwicklungsprozesses

Als eine Grundlage für diese Diplomarbeit dienen die Ergebnisse der Projektgruppe 263 *PROMETHEUS*, die im Sommersemester 1995 und Wintersemester 1995/96 am Lehrstuhl X (Software-Technologie) des Fachbereichs Informatik an der Universität Dortmund stattfand. In dieser Projektgruppe ging es darum, ein rechnergestütztes Behandlungsmanagementsystem für den stationären Krankenhausbereich zu konzipieren und dieses prototypisch zu implementieren [BCF+96a].

Wie im Verlauf der Projektgruppe festgestellt wurde, sollte die Patientenakte im Mittelpunkt eines Behandlungsmanagementsystems (BMS) stehen. Sie ist eine zentrale Komponente zur Unterstützung der Diagnose, Therapie und Pflege von Patienten. Die Spezifikation einer computerunterstützten Patientenakte war eines der zentralen Themen von *PROMETHEUS*. Das von *PROMETHEUS* spezifizierte rechnergestützte BMS stellt allerdings kein Hypermediasystem dar, da die in Kapitel 2.3 vorgestellten Punkte nicht erfüllt werden:

Verknüpfung von lose zusammengehörigen Informationseinheiten. Es wurde bei der Spezifikation nicht vordringlich darauf geachtet, die verschiedenen Aspekte der Patientenakte nach logischen Kriterien zu verknüpfen.

Einbindung von Bilddaten. Die Spezifikation geht nicht auf die Einbindung von computergenerierten Bilddaten wie Röntgenbildern ein.

Einbindung von externen Werkzeugen. *PROMETHEUS* berücksichtigt die Möglichkeit, Teile der computerunterstützten Patientenakte über externe Anwendungen zu bearbeiten und in das System einzubinden, nur ansatzweise.

Um diese Nachteile auszugleichen, wurde bereits in [BCF+96a] die Anforderung gestellt, das von *PROMETHEUS* spezifizierte rechnergestützte BMS als Hypermediasystem zu entwickeln. Dies geschieht hier im Rahmen dieser Diplomarbeit, indem ein Systementwurf mit der Methode OOHDM auf der Grundlage der ermittelten und in Kapitel 3 vorgestellten Anforderungen durchgeführt wird. OOHDM geht allerdings nicht explizit auf die Anforderungsanalyse des zu spezifizierenden Systems ein. Aus diesem Grund wurden die Projektgruppenergebnisse überprüft und überarbeitet. Es wurde ein neues Anwendungsfallmodell erstellt, welches als informelle Grundlage für das Konzeptmodell der ersten Phase von OOHDM dient.

Diese Diplomarbeit stützt sich auf die Dokumente und Analysen, die in Zusammenarbeit mit dem Mathias-Spital in Rheine der Projektgruppe als Arbeitsmaterial dienten.

Die Projektgruppe 263 *PROMETHEUS* lieferte die folgenden wesentlichen Ergebnisse [BCF+96b]:

- Eine umfassende Ist-Analyse des Behandlungsmanagements
- Eine Spezifikation eines rechnergestützten Behandlungsmanagementsystems nach OOSE [Jac92] mittels:
  - Anwendungsfällen, Akteuren und einem Problembereichsmodell
  - einem Analyseobjektmodell
  - einem Prototypen

4.1.1 Das Problembereichsmodell

Die wichtigsten Objekte des von der Projektgruppe 263 *PROMETHEUS* erarbeiteten Problembereichsmodells, die für diese Diplomarbeit relevant sind, seien hier kurz alphabetisch und unkommentiert aufgelistet. Die vollständige Beschreibung dieser Objekte befindet sich in [BCF+96b].

`Anforderungen` (siehe unten)	`Maßnahme`
`Anforderungsergebnisse` (siehe unten)	`Patientenmappe`
`Arztbrief`	`Pflegebericht`
`Belegungsplan`	`Pflegemaßnahme`
`Blutgruppenbestimmungszettel`	`Planette`
`Einlage zur Krankengeschichte`	`Signalplan`
`Einweisung des Hausarztes`	`Spezielle Pflege`
`Einweisungsdiagnose`	`Stammblatt`
`Entlassungsdiagnose`	`Standardtherapieplan`
`Entlassungsurkunde`	`Terminplan`
`Externes Gutachten`	`Therapieplan`
`Fieberkurve`	`Verlegung`
`Krankengeschichte`	`Visiteverordnung`
`Kurzbericht`	`Vitalwerteintrag`

Die `Anforderungen` umfassen Formulare für `Physikalische Therapie`, `OP/Anästhesie`, `Apotheke`, `Echokardiographie`, `Essen`, `Konsiliarschein`, `Labor`, `Nuklearmedizin`, `Material`, `Lungenfunktionsprüfung`, `Röntgen` und `Sonographie`. Die `Anforderungsergebnisse` werden durch die entsprechenden Ergebnisformulare dargestellt.

4.1.2 Die Akteure

Die mit dem System arbeitenden Akteure, die die Anwendungsfälle initiieren, sind:

- Chefarzt
- Verwaltungsangestellter
- Oberarzt
- Stationsarzt
- Stationsschwester
- Personal bei einer Leistungsstelle (Labor, Röntgenabteilung, etc.)

4.1.3 Das Anwendungsfallmodell

Die folgenden Anwendungsfälle wurden für das System überprüft und neu festgelegt:

Patient neu aufnehmen	*Neuaufnahme Pflegepersonal*
Neuaufnahme Arzt	*Patient verlegen*
Patient entlassen	*Stammdaten bearbeiten*
Diagnosedaten bearbeiten	*Pflegedaten bearbeiten*
Therapiedaten bearbeiten	*Fieberkurve bearbeiten*
Anforderung für Station erstellen	*Anforderung für Patient erstellen*
Nachricht senden	*Terminplan Medizinisches Personal bearbeiten*
Terminplan Patient bearbeiten	*Terminplan Station bearbeiten*
Terminplan Leistungsstelle bearbeiten	*Termin vereinbaren*
Anforderungsergebnis Station liefern	*Anforderungsergebnis Patient liefern*
Abrechnung Station erstellen	*Abrechnung Patient erstellen*
Patientenakte aus Archiv laden	*Patientenakte archivieren*
Externes Dokument integrieren	

Exemplarisch seien hier zwei wichtige Anwendungsfälle vorgestellt. Die vollständige Beschreibung der überarbeiteten Anwendungsfälle befindet sich in [Pro97].

Patient neu aufnehmen. Der Anwendungsfall *Patient neu aufnehmen* besteht aus den folgenden Anwendungsfällen: *Neuaufnahme Pflegepersonal*, *Neuaufnahme Arzt*.

Neuaufnahme Pflegepersonal. Der Patient kommt auf der Station an. Die Schwester überprüft, ob die Akte des Patienten im Archiv gespeichert ist.

- Wenn dies der Fall ist, lädt sie die alte Akte des Patienten ins System.
- Wenn dies nicht der Fall ist, legt sie eine neue Patientenakte an.

Dann nimmt die Schwester die Stammdaten des Patienten neu auf oder korrigiert die eventuell vorhandenen alten Daten (Anwendungsfall *Stammdaten bearbeiten*). Die mitgeführten Dokumente des Patienten werden ins System eingegeben. Die Schwester bearbeitet bei Bedarf die Krankengeschichte (Anwendungsfall *Diagnosedaten bearbeiten*). Sie weist dem Patienten ein Bett zu. Anschließend wird der Stationsarzt benachrichtigt (*Nachricht senden*), daß er den Patienten aufnehmen muß (Anwendungsfall *Neuaufnahme Arzt*).

Neuaufnahme Arzt. Der Arzt untersucht den neu angekommenen Patienten und bearbeitet dabei die Krankengeschichte (Anwendungsfall *Diagnosedaten bearbeiten*). Er erstellt eine Einweisungsdiagnose (Anwendungsfall *Diagnosedaten bearbeiten*). Er erstellt für den Patienten einen Standardtherapieplan (Anwendungsfall *Therapiedaten bearbeiten*). Er erstellt eine Kurzanamnese (Anwendungsfall *Pflegedaten bearbeiten*). Er kann bei Bedarf eine Anforderung für den Patienten erstellen (Anwendungsfall *Anforderung für Patient erstellen*).

Anforderung für Patient erstellen. Der Benutzer will eine Anforderung für einen Patienten vornehmen. Er ruft eine Liste mit möglichen Anforderungen auf und wählt eine davon aus. Diese wird angezeigt und er füllt die nötigen Felder aus. Anschließend schickt er die Anforderung ab. Der Anwendungsfall *Nachricht senden* wird ausgelöst. Bei manchen Unterfällen wird der Anwendungsfall *Termin vereinbaren* ausgelöst. Mögliche Anforderungen sind EKG-Anforderung, Konsiliarschein, Röntgen-Anforderung, Echokardiographieanforderung, Nuklearmedizinanforderung, Sonographie-Anforderung, Lungenfunktionsprüfung, Anforderung physikalische Therapie, OP-Anforderung mit Anästhesie, Apothekenanforderung, Essensanforderung und Laboranforderung.

4.2 Konzeptentwurf

In diesem Kapitel wird die durchgeführte Entwicklung des Konzeptentwurfs der hypermedialen Patientenakte dokumentiert und das mittels der von OOHDM propagierten erweiterten OMT-Notation entworfene Datenmodell, verdeutlicht durch grafische Darstellungen, anhand von Beispielen vorgestellt. Die vollständige Spezifikation ist in [Pro97] nachzulesen.

Wie in Kapitel 2.4.3 beschrieben, wird das Konzeptmodell in einer erweiterten OMT-Notation dokumentiert. Die Elemente der Notation bezeichnen Subsysteme, Klassen, Objekte mit Attributen und Methoden sowie Beziehungen. Die Dokumentation erfolgt durch Diagramme sowie eine Beschreibung der Elemente. Die Notation wurde in Kapitel 2.4.3 in Abbildung 2.12 vorgestellt.

Die Entwicklung des Konzeptmodells geschah in zwei Iterationen, wobei die zweite Iteration nach einer Präsentation und anschließender Rücksprache mit dem Personal des Mathias-Spitals in Rheine durchgeführt wurde. Als Ergebnis der Rücksprache wurde das Konzeptmodell um die Klasse `Medizinische Datenbank` erweitert, die Standardpflegemaßnahmen, -diagnosen und -therapien beinhaltet, die einfach und direkt in die Dokumentation der entsprechenden Aspekte der Patientenakte eingesetzt werden können. Dies geschah aufgrund des Wunsches des medizinischen Personals, die Eingabe von Daten, die für ungeübte Benutzer mittels Tastatur aufwendig sein kann, zu erleichtern.

Das Klassenmodell wurde auf der Grundlage der in Kapitel 4.1 vorgestellten Anwendungsfälle und Problembereichsobjekte spezifiziert, wobei dieser Prozeß informell durchgeführt wurde.

Das Klassenmodell bildet in vereinfachender Weise die hierarchische Struktur eines Krankenhauses ab, wie in Abbildung 4.1 dargestellt. Die oberste Organisationseinheit ist das `Krankenhaus`. Dieses ist eine Aggregation der Klassen `Verwaltung`, 1-n `Stationen` und 1-n `Leistungsstellen`. Auf einer `Station` arbeiten 1-n `Schwestern` und 1-n `Ärzte`, die die gemeinsame Oberklasse `Medizinisches Personal` haben. Eine `Station` ist außerdem mit 1-n `Patienten` belegt.

Als durchgängige Fallbeispiele, anhand deren die einzelnen Entwicklungsschritte erläutert werden, werden die Klasse `Patient` und die Beziehung `AnforderungPatient` näher betrachtet. Diese werden in den folgenden Phasen wieder aufgegriffen.

4.2.1 Subsysteme

Die Klassen wurden zu den in Abbildung 4.2 dargestellten Subsystemen zusammengefaßt, die im folgenden erläutert werden:

Stationssystem: Das Subsystem *Stationssystem* umfaßt die direkt die Station betreffenden Klassen `Station`, `Schwester` und `Arzt` sowie die zugeordneten `Terminpläne`. Es wird in Abbildung 4.3 grafisch dargestellt.

Patientensystem: Das in Abbildung 4.4 dargestellte Subsystem *Patientensystem* umfaßt die Klasse `Patient`, bestehend aus den Klassen `Diagostik`, `Pflege`, `Therapie` und `Fieberkurve`, die selbst Aggregationen sind, einen zugeordneten `Terminplan` sowie eine

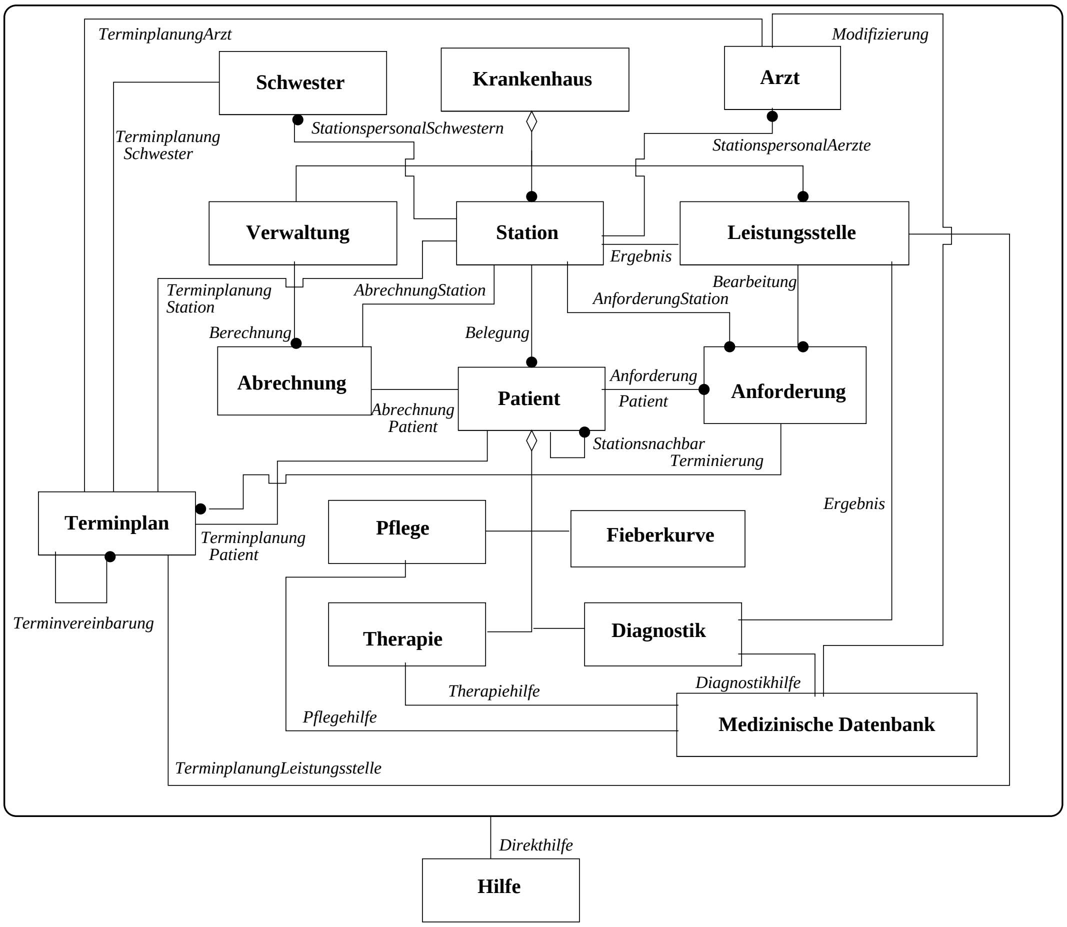


Abbildung 4.1: Übersicht des Konzeptentwurfs

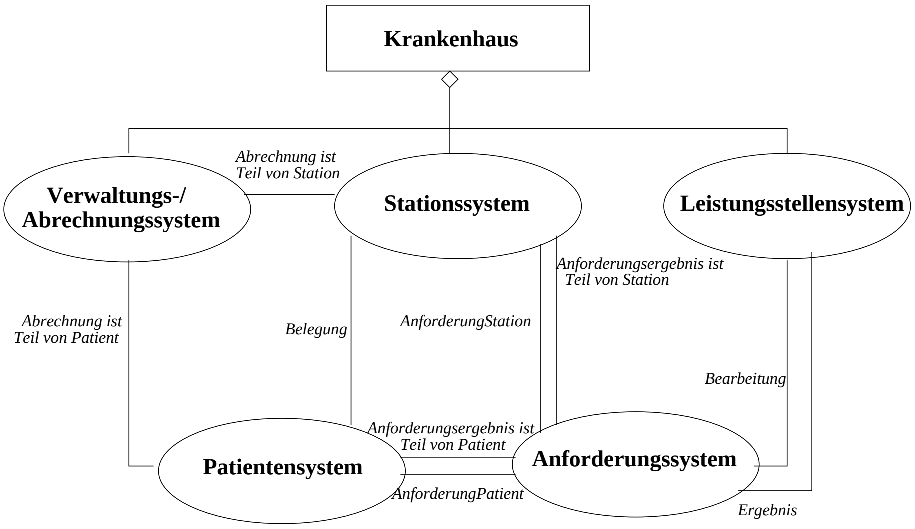


Abbildung 4.2: Die Subsysteme

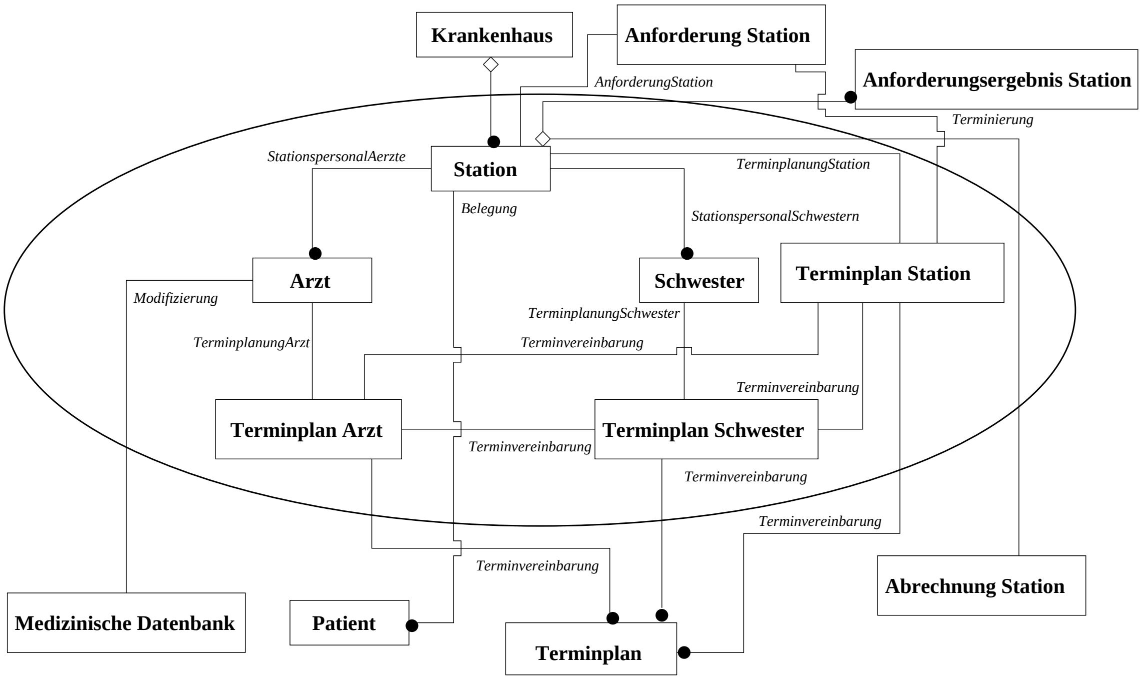


Abbildung 4.3: Das Subsystem Stationssystem

Klasse `Medizinische Datenbank`, die Pflege-, Therapie- und Diagnoseunterstützung durch Vorgabe von Standardmaßnahmen und -einträgen liefert.

Anforderungssystem: Das in Abbildung 4.5 dargestellte Subsystem *Anforderungssystem* umfaßt die möglichen Anforderungen und Anforderungsergebnisse des Systems. Die Oberklasse `Anforderung` besitzt zwei Unterklassen `Anforderung Station` und `Anforderung Patient`, die wiederum Oberklassen weiterer `Anforderungen` sind. Die `Anforderungen Station` sind:

- `Materialanforderung`
- `Laboranforderung`
- `Apothekenanforderung`
- `Essensanforderung`

Die `Anforderungen Station` `Laboranforderung`, `Apothekenanforderung` und `Essensanforderung` sind Sammelanforderungen mehrerer `Anforderungen Patient`, was hier durch eine Aggregationsbeziehung ausgedrückt wird. Die `Anforderungsergebnisse Station` sind:

- `Materialanforderungsergebnis`
- `Laboranforderungsergebnis`
- `Apothekenanforderungsergebnis`
- `Essensanforderungsergebnis`

Bei den Ergebnissen ist nur das `Laboranforderungsergebnis` als Aggregation aus mehreren `Anforderungen Patient` modelliert, da die Größe der einzelnen Ergebnisse der anderen Anforderungen keine eigene Klasse rechtfertigen. Die `Anforderungen Patient` sind:

- `EKG-Anforderung`
- `Nuklearmedizinanforderung`
- `Sonografieanforderung`

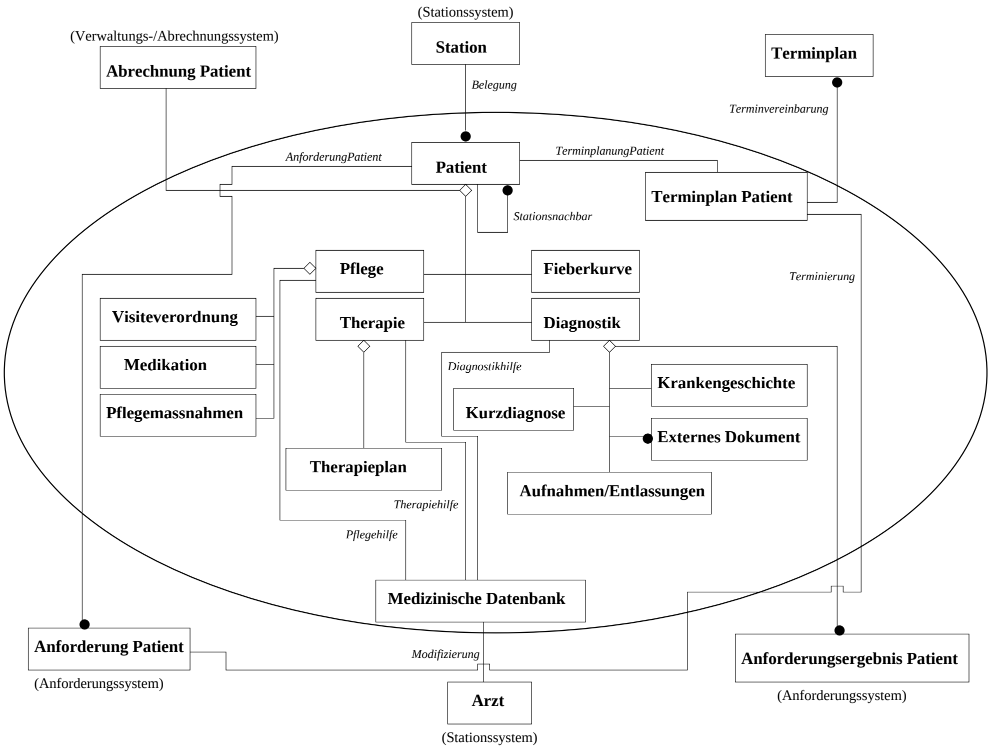


Abbildung 4.4: Das Subsystem Patientensystem

- `Lungenfunktionsprüfung`
- `Anforderung Physikalische Therapie`
- `OP/Anästhesieanforderung`
- `Röntgenanforderung`
- `Konsiliarschein`
- `Laboranforderung Patient`
- `Apothekenanforderung Patient`
- `Essensanforderung Patient`

Die letzten drei `Anforderungen` sind die oben schon erwähnten Teilanforderungen einer `Anforderung Station`. Die `Anforderungsergebnisse Patient` sind:

- `EKG-Anforderungsergebnis`
- `Nuklearmedizinanforderungsergebnis`
- `Sonografieanforderungsergebnis`
- `Lungenfunktionsprüfungsergebnis`
- `Anforderung Physikalische Therapieergebnis`
- `OP/Anästhesieanforderungsergebnis`
- `Röntgenanforderungsergebnis`
- `Externes Gutachten`
- `Laboranforderungsergebnis Patient`

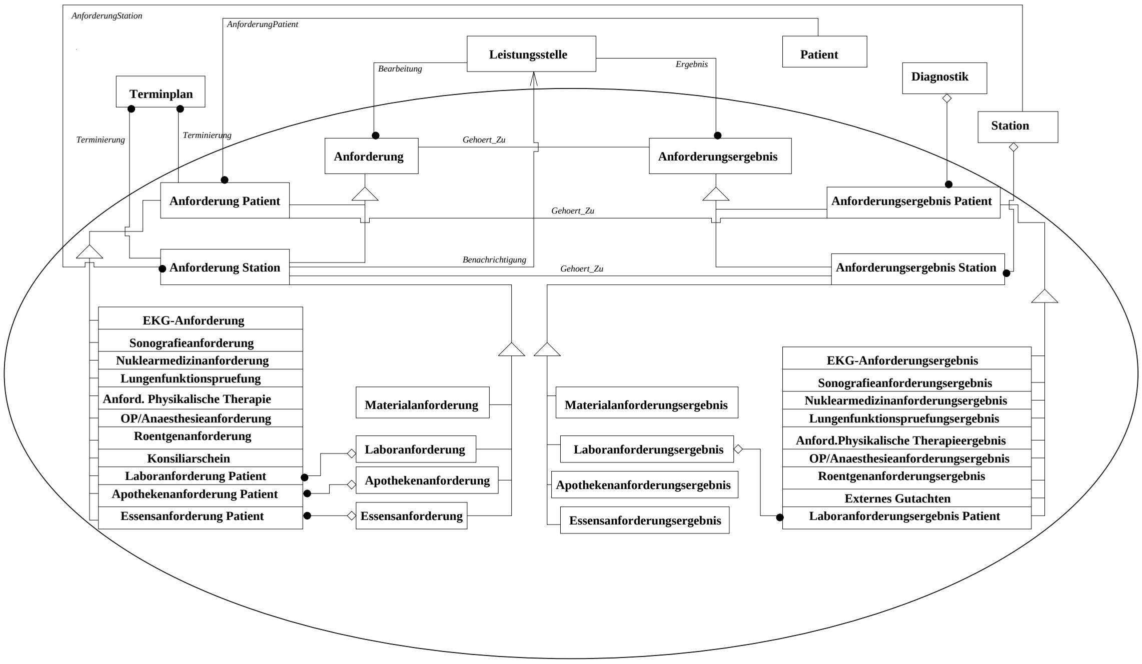


Abbildung 4.5: Das Subsystem Anforderungssystem

Leistungsstellensystem: Dieses in Abbildung 4.6 präsentierte Subsystem umfaßt alle `Leistungsstellen` des `Krankenhauses` sowie einen zugeordneten `Terminplan`. Dieses sind die folgenden:

- `Innere Abteilung`
- `Nuklearmedizin`
- `Abteilung OP/Anästhesie`
- `Abteilung Physikalische Therapie`
- `Radiologie`
- `Apotheke`
- `Labor`
- `Küche`
- `Materialstelle`
- `Externe Stelle`

Die `Leistungsstellen` bearbeiten die `Anforderungen` für `Stationen` und `Patienten` und liefern die entsprechenden `Anforderungsergebnisse`.

Verwaltungs-/Abrechnungssystem: Dieses Subsystem wurde im Rahmen dieser Diplomarbeit nur rudimentär spezifiziert und umfaßt die `Verwaltung` und die `Abrechnungen` für `Patienten` und `Stationen`, wobei die `Abrechnung Station` eine Aggregation aller `Abrechnungen Patient` einer `Station` ist. Dieses Subsystem wird in Abbildung 4.7 grafisch dargestellt.

4.2.2 Klassen und Beziehungen

Aus Platzgründen ist es nicht möglich, an dieser Stelle alle Konzeptklassen detailliert vorzustellen. Darum werden hier beispielhaft die Klasse `Patient` und die Beziehung `Belegung` vorgestellt, da

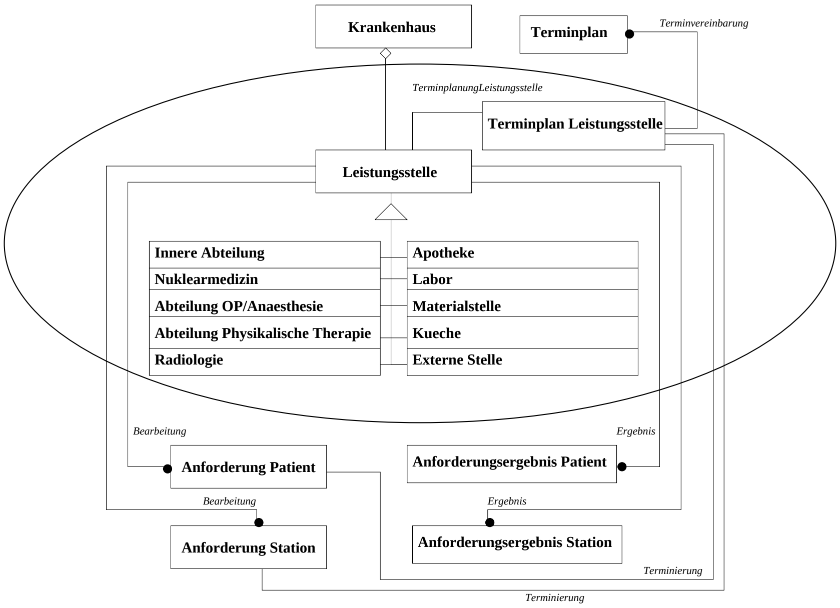


Abbildung 4.6: Das Subsystem Leistungsstellensystem

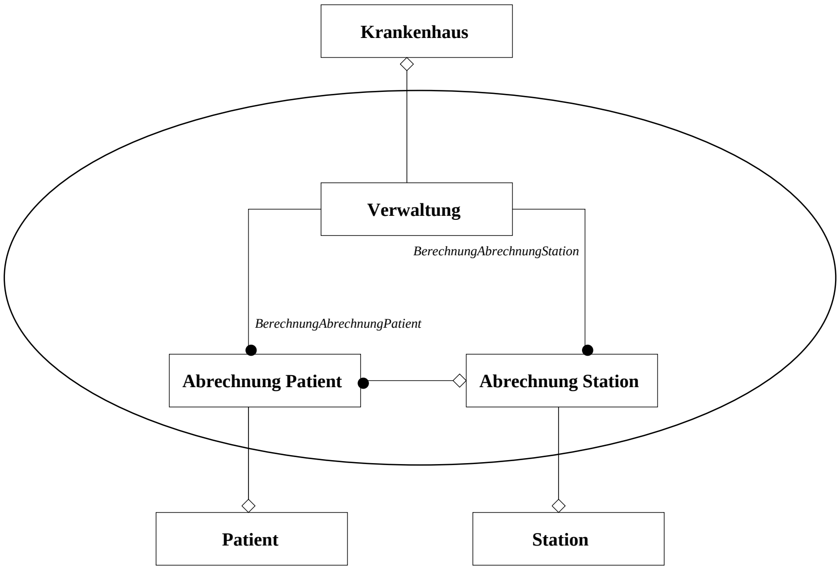


Abbildung 4.7: Das Subsystem Verwaltungs-/Abrechnungssystem

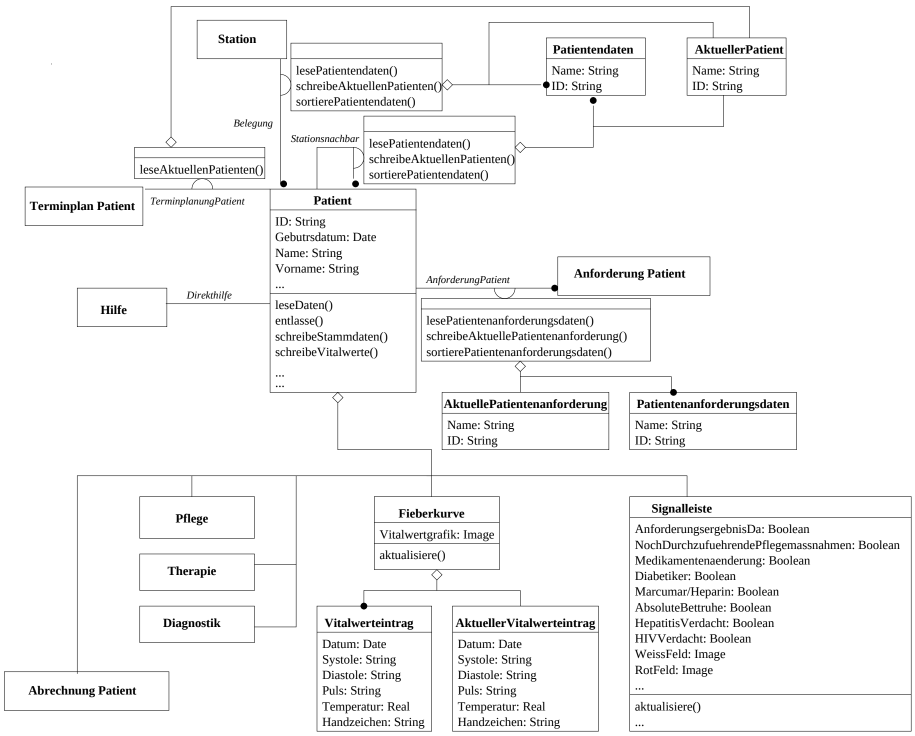


Abbildung 4.8: Umgebung der Klasse Patient

diese Elemente wichtig und repräsentativ für das System sind. Die Notation erfolgt in Anlehnung an die Form der in Kapitel 2.4.3 vorgestellten Klassenkarten. Aus Verständnisgründen werden in den Beispielen die Kommentare direkt nach den einzelnen Charakteristika eingefügt. Eine genaue Vorstellung aller Konzeptobjekte und -beziehungen sowie ihrer Charakteristika und der verwendeten Notation findet sich in [Pro97].

Die Abbildung 4.8 stellt die direkte Umgebung der Klasse `Patient` und Beziehung `Belegung` dar, wobei die Attribute und Methoden sowie die Teilklassen `Abrechnung Patient`, `Pflege`, `Diagnostik` und `Therapie` aus Platzgründen nicht vollständig aufgeführt sind.

4.2.2.1 Klasse Patient

Klassenname: Patient

Kommentar zu Klassenname: Hier steht der Name der Klasse im Konzeptentwurf.

Subsystem: Patientensystem

Kommentar zu Subsystem: Hier werden die 0-n Subsysteme aufgelistet, zu denen die Klasse gehören kann. In diesem Falle ist es das Subsystem Patientensystem.

Erbt von:

Kommentar zu Erbt von: Hier wird eine eventuell existierende Oberklasse der Klasse aufgeführt.

Attribute:

ID: String	Geburtsdatum: Date
Name: String	Vorname: String
AnkunftAllein: Boolean	Ankunftsart: String
Aufnahmedatum: Date	LetztesAufnahmedatum: Date
Krankenkasse: String	AnkunftSanka: Boolean
EinweisenderArzt: String	Hausarzt: String
Aufnahmezustand: String	InkontinenzUrin: Boolean
InkontinenzStuhl: Boolean	Dekubitus: Text
Kontrakturen: Text	Konfession: String
Krankensalbung: Date	HandzeichenAufnahme: String
Sonstiges: Text	WichtigeMedikamente: 1-n Strings
Kostform: String	Brille/Kontaktlinsen: Boolean
Gehhilfen: Boolean	Hörgerät: Boolean
Zahnprothesen: Boolean	Patienteninformationen: String
Entlassungsprobleme: String	Unverträglichkeiten/Allergien: 1-n Strings
Blutgruppe: String	LR: String
ET: String	Entlassungsdatum: Date
Verlegungsdatum: Date	Verlegungsort: String
Straße: String	Wohnort: String
PLZ: String	Telefonnummer: String
AngehörigenName: String	AngehörigenStraße: String
AngehörigenWohnort: String	AngehörigenPLZ: String
AngehörigenTelefonnummer: String	Zimmer: String
Station: String	Legende: Image

Kommentar zu Attribute: An dieser Stelle werden die Attribute der Klasse mit Namen und Typ aufgelistet. Die Attribute der Klasse `Patient` reflektieren die Einträge des Formulars `Stammblatt` des Mathias-Spitals in Rheine, welches Teil der Dokumentation der Projektgruppe 263 *PROMETHEUS* ist (siehe [BCF+96b]) und in Abschnitt 4.1.1 als Problembereichsobjekt festgelegt wurde.

Besteht aus:

- `Pflege`
- `Therapie`
- `Diagnostik`
- `Fieberkurve`
- `Signalleiste`
- `Abrechnung Patient`

Kommentar zu Besteht aus: Falls die Klasse eine Aggregation ist, werden hier die Bestandteile des Aggregats vorgestellt. Im Fall der Klasse `Patient` sind dies die oben genannten Klassen.

Methoden:

- leseDaten()
- aktualisiereSignalleiste()
- aktualisiereVitalwerte()
- schreibeVitalwerte()
- schreibeStammdaten()

Kommentar zu Methoden: Hier werden die Methoden der Klasse spezifiziert. Die hier vorgestellten Methoden besitzen keine Parameter, dies ist aber keine Bedingung. Anmerkung: Methoden, die die Benutzerschnittstelle modifizieren, werden erst im abstrakten Schnittstellenentwurf festgelegt. Es folgt eine kurze Beschreibung der Methoden. Die Methode `leseDaten` liest alle zum `Patienten` zugehörigen Daten, einschließlich der Daten der Klassen `Signalleiste` und `Fieberkurve`, in den Hauptspeicher. Die Methoden `aktualisiereStammdaten` und `aktualisiereVitalwerte` überprüfen die Korrektheit neuer Eingaben der Attribute des `Patienten` (Stammdaten) oder der `Vitalwerte` und bringen im Fall der Korrektheit diese auf den neuesten Stand. Die Methoden `schreibeVitalwerte` und `schreibeStammdaten` speichern die entsprechenden Daten.

Beziehung zu:

- TerminplanungPatient `Terminplan`
- Belegung `Station`
- AnforderungPatient `Anforderung Patient`
- Stationsnachbar 1-n `Patienten`
- Direkthilfe `Hilfe`

Kommentar zu Beziehung zu: An dieser Stelle werden die Beziehungen der Klasse mit ihrer Kardinalität, sofern diese größer als 1 ist, sowie die Zielklassen dieser Beziehungen aufgelistet. Die Klasse `Patient` hat Beziehungen zu den Klassen `Terminplan, AnforderungPatient`, 1-n anderen `Patienten` und der `Hilfe`.

Teil von:

Kommentar zu Teil von: Wenn die Klasse Teil einer Aggregation ist, wird hier das Aggregat festgelegt.

Kommentar: Die Klasse `Patient` ist die zentrale Klasse des Systems. Die Attribute beschreiben die Stammdaten des Patienten, wie sie auf dem Formular Stammdaten aus dem Mathias-Spital in Rheine aufgeführt werden.

Die Klasse `Patient` ist eine Aggregation aus den Klassen `Pflege, Therapie, Diagnostik, Fieberkurve, Signalleiste` und `Abrechnung Patient`.

Die Methode `leseDaten` liest alle zum `Patienten` zugehörigen Daten, einschließlich der Daten der Klassen `Signalleiste` und `Fieberkurve`, in den aktiven Speicher. Die Methoden `aktualisiereStammdaten` und `aktualisiereVitalwerte` überprüfen die Korrektheit neuer Eingaben der Attribute des `Patienten` (Stammdaten) oder der `Vitalwerte` und bringen im Fall der Korrektheit diese auf den neuesten Stand. Die Methoden `schreibeVitalwerte` und `schreibeStammdaten` speichern die entsprechenden Daten.

Die Klasse hat Beziehungen zu einem Objekt der Klasse `Terminplan`, welche die Terminplanung des `Patienten` wiedergibt, zur `Station` in Form der `Belegung`, zur Klasse `Anforderung Patient`, die die möglichen `Anforderungen` eines `Patienten` zusammenfaßt, zu 1-n anderen `Patienten`, die auf derselben `Station` liegen, und zur Klasse `Hilfe`.

Die Umgebung der Klasse `Patient` ist in Abbildung 4.8 zu sehen.

Kommentar zu Kommentar: Hier werden die Eigenschaften der vorgestellten Klasse näher erläutert.

Trace fwd:

- `Patient` (Verwaltungssicht)
- `Patient` (Schwesternsicht)
- `Patient` (Arztsicht)

- `Patient` (Besuchersicht)

Kommentar zu Trace fwd: An dieser Stelle werden aus der hier vorgestellten Konzeptklasse abgeleitete Navigationsklassen und die navigationellen Sichten, zu denen sie gehören, erwähnt. In diesem Fall existiert in jeder der vier obengenannten navigationellen Sichten eine Navigationsklasse `Patient`.

Trace bck:

- `Blutgruppenbestimmungszettel`
- `Patientenmappe`
- `Planette`
- `Stammblatt`

Kommentar zu Trace bck: Hier werden Problembereichsobjekte präsentiert, aus denen die vorgestellte Konzeptklasse spezifiziert wurde. In Falle der Klasse `Patient` sind dies die obengenannten Problembereichsobjekte.

4.2.2.2 Beziehung AnforderungPatient

Beziehungsname: AnforderungPatient

Verbundene Klassen:

- `Patient`
- 1-n `Anforderungen Patient`

Kommentar zu Verbundene Klassen: An dieser Stelle werden die Konzeptklassen, die die Beziehung miteinander verbindet, sowie die Kardinalität der Beziehung vorgestellt. Die hier vorgestellte Beziehung `AnforderungPatient` verbindet die Klasse `Patient` mit 1-n Instanzen der Klasse `Anforderung Patient`.

Attribute:

Besteht aus:

- `AktuellePatientenanforderung`
- 1-n `Patientenanforderungsdaten`

Kommentar zu Besteht aus: Hier werden Klassen vorgestellt, die der Beziehung direkt zugeordnet sind. Im Falle dieser Beziehung sind dies die Klassen AktuellePatientenanforderung und `Patientenanforderungsdaten`, die benötigt werden, um die konkrete Instanz von `Anforderung Patient`, die in einem gegebenen Zustand mit der Klasse `Patient` verbunden ist, festzulegen.

Methoden:

- lesePatientenanforderungsdaten()
- schreibeAktuellePatientenanforderung()
- sortierePatientenanforderungsdaten()

Kommentar zu Methoden: An dieser Stelle werden Methoden der Beziehung vorgestellt. Die hier genannten Methoden ermöglichen die Auswahl einer konkreten Instanz von `Anforderung Patient`.

Erbt von:

Oberklasse von:

Kommentar: Die Beziehung `AnforderungPatient` stellt die Möglichkeit dar, für einen `Patienten` verschiedene `Anforderungen` vorzunehmen. Das zugeordnete Objekt `AktuellePatientenanforderung` stellt die aktuell ausgewählte `Anforderung Patient` dar, die in einer Liste aus den Objekten `Patientenanforderungsdaten`, die alle möglichen `Anforderungen Patient` (`EKG-Anforderung,Sonografieanforderung, Nuklearmedizinanforderung, Lungenfunktionsprüfung, Röntgenanforderung, OP/Anästhesieanforderung, Anforderung Physikalische Therapie, Konsiliarschein, Apothekenanforderung Patient, Laboranforderung Patient` und `Essensanforderung Patient`) zusammenfaßt, eingetragen ist.

Die Methode `lesePatientenanforderungsdaten` lädt die Liste der `Patientenanforderungsdaten` in den Hauptspeicher. Die Methode `schreibeAktuellePatientenanforderung` spezifiziert die Instanz der Zielklasse `Anforderung Patient`. Die Methode `sortierePatientenanforderungsdaten` sortiert die Liste der `Anforderungen Patient` nach vorgegebenen Kriterien.

Kommentar zu Kommentar: Hier werden die Eigenschaften der vorgestellten Beziehung näher erläutert.

Trace fwd:

- `AnforderungPatient` (Schwesternsicht)
- `AnforderungPatient` (Arztsicht)

Kommentar zu Trace fwd: An dieser Stelle werden Verknüpfungen des Navigationsentwurfs mitsamt der Sicht, zu der sie gehören, erwähnt, die aus der hier vorgestellten Beziehung des Konzeptentwurfs hervorgehen. Hier ist dies die gleichnamige Verknüpfung `AnforderungPatient`, die sowohl in der Schwesternsicht als auch in der `Arztsicht` existiert.

4.3 Navigationsentwurf

In diesem Kapitel wird die Vorgehensweise beim Navigationsentwurf der hypermedialen Patientenakte geschildert. Es werden die Sichten und die dort verwendeten Konstrukte vorgestellt und im Fall der Schwesternsicht näher erläutert. Die aus dem Konzeptentwurf bekannten Beispielklassen `Patient` und `AnforderungPatient` werden hier wiederaufgenommen und erläutert.

OOHDM propagiert die Spezifikation von verschiedenen Sichten über das beim Konzeptentwurf konstruierte Modell. Sichten auf das System reflektieren die Aufgaben und Fähigkeiten der unterschiedlichen Benutzer des im Entwurfsprozeß befindlichen Systems. Das bedeutet in diesem Fall, daß für jede Benutzergruppe der hypermedialen Patientenakte eine Sicht spezifiziert wird.

Es wird festgelegt, daß es vier Arten von Benutzern gibt, die mit dem System arbeiten:

- **Schwester** auf einer Station
- **Arzt** auf einer Station
- **Verwaltungsangestellter**
- **Personal** einer Leistungsstelle

Für die Zwecke dieser Diplomarbeit wird die Tatsache, daß es verschiedene Arten von Ärzten und Schwestern auf einer Station gibt, nicht weiter berücksichtigt. Es wird festgelegt, daß die Aufgabenbereiche für die jeweilige Berufsgruppe in Bezug auf die Patientenakte ähnlich genug sind, um eine Zusammenfassung zu einer einzigen Sicht zu rechtfertigen. Zusätzlich zu den obengenannten Sichten wird noch eine Besuchersicht spezifiziert. Dies stellt ein Informationssystem dar, das einem Besucher des Krankenhauses in grafisch ansprechender und informativer Weise das Krankenhaus und seine Abteilungen vorstellt.

Damit werden fünf verschiedene navigationelle Sichten spezifiziert:

- *Schwesternsicht*
- *Arztsicht*
- *Verwaltungssicht*
- *Leistungsstellensicht*
- *Besuchersicht*

Die Spezifikation einer Sicht erfolgt, indem Klassen und Objekte des Konzeptmodells, die für den Benutzer der Sicht wichtig sind, nach aufgabenorientierten Kriterien neu gruppiert, zu Knotenklassen zusammengesetzt und verknüpft werden, wobei für den Aufgabenbereich des jeweiligen Benutzers irrelevante oder nicht zugängliche Datenobjekte nicht berücksichtigt werden. Die verwendeten Konstrukte wurden in Kapitel 2.4.3 vorgestellt.

OOHDM legt nicht genau fest, wie Aggregationen in Bezug auf die Navigation behandelt werden. Für diese Diplomarbeit wird festgelegt, daß jede Aggregation einen diskreten Knoten darstellt, welcher einem zusammengesetzten Knoten des in Kapitel 2.2.3.3 vorgestellten Amsterdam Hypermedia Models entspricht. Aus diesem Grund wurden mehrere Aggregationen des Konzeptmodells aufgelöst, um die Existenz von mehreren Knoten für manche Konzeptklassen zu dokumentieren.

4.3.1 Schwesternsicht

Die *Schwesternsicht* stellt für diese Diplomarbeit die zentrale Sicht dar. Diese Sicht vereint Aspekte der Stationsverwaltung und Patientendokumentation. Aus diesem Grund wird diese Sicht hier am vollständigsten dokumentiert. Eine umfassende Spezifikation aller Sichten befindet sich im Anhang [Pro97]. Die *Schwesternsicht* umfaßt die im Konzeptmodell definierten Subsysteme *Stationssystem*, *Patientensystem* und *Anforderungssystem* (siehe Abbildung 4.2 und Kapitel 4.2.1). Die Abbildung 4.9 stellt eine Übersicht der Knotenklassen und Verknüpfungen der Schwesternsicht dar.

4.3.1.1 Knoten und Verknüpfungen

Die Knotenklasse `Station` stellt zusammen mit der Knotenklasse `Patient` die wichtigste Organisationseinheit in dieser Sicht dar. Die Knotenklasse `Station` besitzt Anker zur Klasse `Anforderung Station` mit den Unterklassen `Materialanforderung, Laboranforderung, Essensanforderung` und `Apothekenanforderung` sowie den korrespondierenden `Anforderungsergebnissen Station`. Ferner ist der `Station` eine Instanz des `Terminplans` zugeordnet. Von der `Station` aus gibt es eine Verknüpfung zur Knotenklasse `Schwester`, welche persönliche `Notizen` des gerade mit dem System arbeitenden Benutzers **Schwester** umfaßt, sowie eine weitere Verknüpfung zum `Terminplan` der `Schwester`. Die letzte Verknüpfung der `Station` gibt die Tatsache wieder, daß diese mit 1-n `Patienten` belegt ist.

Die Knotenklasse `Patient` ist ein zentrales Element des Systems. Im Unterschied zum Konzeptentwurf ist die Aggregation aus `Pflege, Diagnostik` und `Therapie` aufgelöst worden. Diese Klassen bilden hier eigene Knotenklassen. Die `Fieberkurve` ist ebenso wie die `Signalleiste` als Bestandteil des `Patienten` verblieben. Die Klasse `Anforderungsergebnis Patient` wurde aus der Aggregation `Diagnostik` entfernt und als eigene Knotenklasse definiert, um ihre Signifikanz zu unterstreichen. Die zugehörigen `Anforderungen Patient` haben eine Verknüpfung zur Knotenklasse `Patient`. Dem `Patienten` ist eine Instanz der Klasse `Terminplan` zugeordnet.

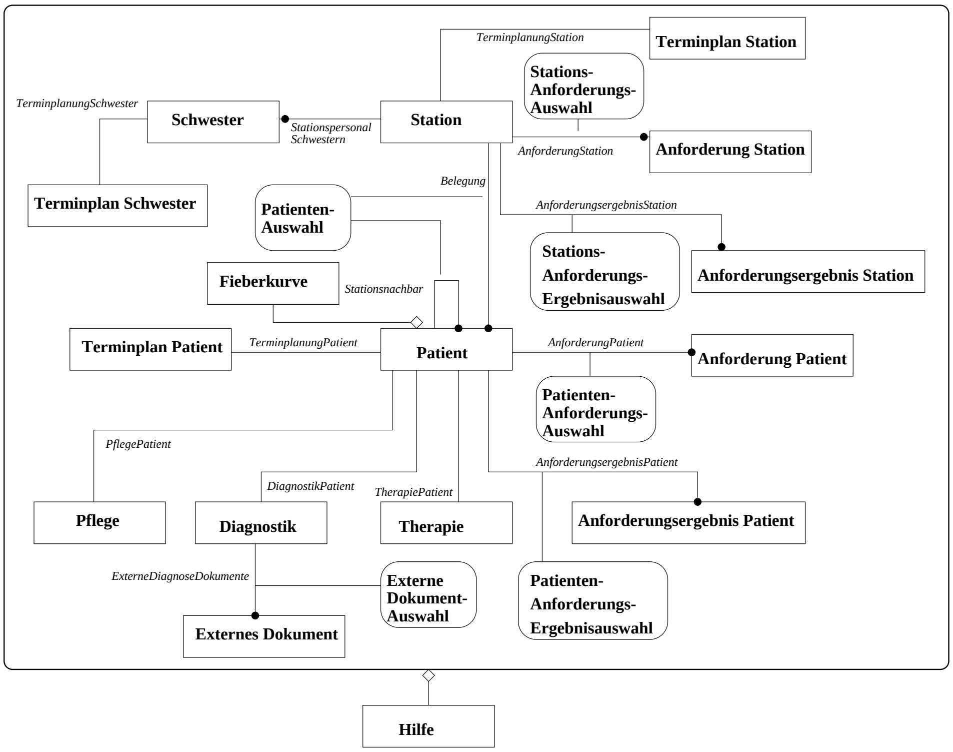


Abbildung 4.9: Die Schwesternsicht

Die Knotenklasse `Pflege` ist ein Aggregat und setzt sich aus den Klassen `Anmerkungen Pflege`, `Pflegemaßnahmen`, `Visiteverordnung` und `Medikation` zusammen. Ihr wurde weiterhin die Klasse `PflegeHilfe` der Konzeptklasse `Medizinische Datenbank` zugeordnet, die in ihrer Konzeptform im Navigationsentwurf nicht existiert.

Die Knotenklasse `Diagnostik` ist ebenfalls ein Aggregat und setzt sich aus den Klassen `Aufnahmen/Entlassungen`, `Einweisungsdiagnose/Kurzanamnese`, `Krankengeschichte` und `Entlassung` zusammen. Die Klasse `Externes Dokument` wurde aus der Aggregation entfernt und ist der Knotenklasse `Diagnostik` über die Verknüpfung `ExterneDiagnoseDokumente` zugeordnet. Dies reflektiert die Tatsache, daß systemexterne Dokumente nicht direkt im Rahmen des Systems angezeigt werden sollen, sondern in einem externen Fenster.

Die Klasse `Therapie` ist ein Aggregat und besteht aus der Klasse `Therapieplan`.

Die Knotenklasse `Anforderung Station` ist direkt aus dem Konzeptentwurf übernommen worden. Gleiches gilt für die Knotenklasse `Anforderungsergebnis Station`. Die beiden Knotenklassen bilden die Oberklassen für die Klassen `Materialanforderung`, `Laboranforderung`, `Apothekenanforderung` und `Essensanforderung` respektive die zugehörigen `Anforderungsergebnisse Station`.

Gleiches gilt für die Knotenklassen `Anforderung Patient` und `Anforderungsergebnis Patient`, mit dem Unterschied, daß hier einige Patientenattribute zwecks eindeutiger Zuordnung und Verbesserung der Übersichtlichkeit hinzugefügt worden sind. Die entsprechenden Unterklassen sind die Klassen `EKG-Anforderung`, `Konsiliarschein`, `Röntgenanforderung`, `Nuklearmedizinanforderung`, `Sonographie-Anforderung`, `Lungenfunktionsprüfung`, `Anforderung physikalische Therapie`, `OP-Anästhesieanforderung`, `Apothekenanforderung`, `Essensanforderung` und `Laboranforderung` sowie die zugehörigen `Anforderungsergebnisse Patient`.

Die Knotenklasse `Terminplan` wurde direkt aus dem Konzeptmodell übernommen. Jeweils eine Instanz dieser Klasse ist jeder Instanz der Klassen `Station`, `Patient` und `Schwester` zugeordnet.

Jeder Knotenklasse ist eine Instanz der Klasse `Hilfe` als Aggregat zugeordnet.

Die Verknüpfungen `Belegung, Stationsnachbar, AnforderungStation, AnforderungsergebnisStation, AnforderungPatient, AnforderungsergebnisPatient` und `ExterneDiagnoseDokumente` werden durch Zugangsstrukturen realisiert. Diesen Verknüpfungen sind im allgemeinen zwei Datenklassen zugeteilt. Im Falle der `AnforderungPatient` ist es das Objekt `AktuellePatientenanforderung`, welches Informationen über die ausgewählte `Anforderung Patient` beinhaltet, und eine Liste mit 1-n `Patientenanforderungsdaten`, die alle existierenden `Anforderungen Patient` beinhaltet und aus der die `Aktuelle Patientenanforderung` ausgewählt wird. Bei den anderen Verknüpfungen liegen analoge Strukturen vor.

Anmerkung: Die 1-zu-n-Verknüpfung `StationspersonalSchwestern` wird nicht durch eine Zugangsstruktur realisiert. Der Grund dafür ist, daß die Auswahl der `Aktuellen Schwester` nicht erst im Fall der Navigation entlang dieser Verknüpfung geschieht, sondern innerhalb des Knotens `Station` beim Auslösen der Methode `überprüfeLogin`.

In den Abschnitten 4.2.2.1 und 4.2.2.2 wurden die Konzeptklasse `Patient` und die Beziehung `AnforderungPatient` näher vorgestellt und erläutert. Die daraus hervorgehenden Klassen des Navigationsentwurfs, die Navigationsklasse `Patient` und die Verknüpfung `AnforderungPatient` der `Schwesternsicht` werden hier nun erneut etwas detaillierter betrachtet. Die Beschreibung der Elemente erfolgt wiederum in an die Klassenkarten angelehnter Form, wobei erläuternde Kommentare direkt den einzelnen Charakteristika der Beschreibung zugeordnet sind. Der vollständige Navigationsentwurf findet sich in [Pro97].

4.3.1.2 Navigationsklasse Patient

Klassenname: Patient

Erbt von:

Attribute:

StationsName: String (von `Station`)	StationsID: String (von `Station`)
ID: String	Geburtsdatum: Date
Name: String	Vorname: String
AnkuftAllein: Boolean	Ankunftsart: String
Aufnahmedatum: Date	Letztes Aufnahmedatum: Date
Krankenkasse: String	AnkunftSanka: Boolean
Einweisender Arzt: String	Hausarzt: String
Aufnahmezustand: String	Inkontinenz Urin: Boolean
Inkontinenz Stuhl: Boolean	Dekubitus: Text
Kontrakturen: Text	Konfession: String
Krankensalbung: Datum	HandzeichenAufnahme: String
Sonstiges: Text	WichtigeMedikamente: 1-n Strings
Kostform: String	Brille/Kontaktlinsen: Boolean
Gehhilfen: Boolean	Hörgerät: Boolean
Zahnprothesen: Boolean	Patienteninformationen: String
Entlassungsprobleme: String	Unverträglichkeiten/Allergien: 1-n Strings
Blutgruppe: String	LR: String
ET: String	Entlassungsdatum: Datum
Verlegungsdatum: Datum	Verlegungsort: String
Straße: String	Wohnort: String
PLZ: String	Telefonnummer: String
AngehörigenName: String	AngehörigenStraße: String
AngehörigenWohnort: String	AngehörigenPLZ: String
AngehörigenTelefonnummer: String	Zimmer: String
Station: String	Legende: Image

Station: Anker (`StationsbelegungPatienten`)
Patientenliste: Anker (`Stationsnachbar`) über `PatientenAuswahl`
TerminplanPatient: Anker (`TerminplanungPatient`)
Patientenanforderung: Anker (`AnforderungPatient`) über `PatientenanforderungsAuswahl`
Diagnostik: Anker (`DiagnostikPatient`)
Therapie: Anker (`TherapiePatient`)
Pflege: Anker (`PflegePatient`)
Patientenanforderungsergebnis: Anker (`AnforderungsergebnisPatient`) über `PatientenanforderungsergebnisAuswahl`

Kommentar zu Attribute: An dieser Stelle werden die Attribute der Navigationsklasse festgelegt. Navigationsklassen können aus Attributen verschiedener Konzeptklassen zusammengesetzt werden. Wenn Attribute von einer anderen als der gleichnamigen Konzeptklasse übernommen werden, wird dies ausdrücklich gekennzeichnet, wie in obigem Beispiel die Attribute `StationsName` und `StationsID`, die von der Konzeptklasse `Station` stammen. Ferner werden hier Anker der Klasse definiert. Ein Anker wird durch einen Namen, die Kennung 'Anker' und die zugehörige Beziehung beschrieben. Ein Beispiel hierfür ist der oben definierte Anker `Patientenanforderung`, der die Navigationsklasse `Patient` über die Beziehung `AnforderungPatient`, die unten näher beschrieben wird, mit der Navigationsklasse `Anforderung Patient` verbindet.

Besteht aus:

- `Fieberkurve`
- `Signalleiste`
- `Hilfe`

Kommentar zu Besteht aus: Hier werden die Bestandteile einer Aggegation dokumentiert. In dem konkreten Fall der Klasse `Patient` sind dies die Klassen `Fieberkurve, Signalleiste` und `Hilfe`.

Methoden:

- leseDaten()
- aktualisiereSignalleiste()
- aktualisiereVitalwerte()
- schreibeVitalwerte()
- schreibeStammdaten()

Kommentar zu Methoden: Die hier aufgelisteten Methoden entsprechen den für diese Klasse übernommenen Methoden des Konzeptentwurfs.

Navigationelle Kontexte:

- `Patientenkontext`
- `Neuaufnahmekontext`

Kommentar zu Navigationelle Kontexte: An dieser Stelle werden die navigationellen Kontexte, zu denen die Klasse gehört, aufgezählt. Im Fall der Klasse `Patient` sind dies die Kontexte `Patientenkontext` und `Neuaufnahmekontext`.

Teil von:

Kommentar zu Teil von: Wenn die beschriebene Klasse Teil einer Aggregation ist, wird dies hier vermerkt.

Kommentar:

Die Klasse `Patient` stellt den Patienten in der Schwesternsicht dar. Die Attribute stammen aus dem Konzeptentwurf und beschreiben die Stammdaten des `Patienten`. Hinzugekommen sind die Attribute `StationsName` und `StationsID`, die die Zugehörigkeit des `Patienten` beschreiben.

Der Anker `Station` ermöglicht die Navigation zur zugehörigen `Station`. Der Anker `Patientenliste` ermöglicht die Navigation zu einem anderen `Patienten` auf derselben `Station` über die Zugangsstruktur `PatientenAuswahl`. Der Anker `TerminplanPatient` ist Ausgangspunkt für die Navigation zum `Patiententerminplan`. Die Anker `Diagnostik, Therapie` und `Pflege` ermöglichen Navigation zu den gleichnamigen Klassen. Die Anker `Patientenanforderung` und `Patientenanforderungsergebnis` sind Ausgangspunkte für die Navigation zu den entsprechenden `Anforderungen` und `Anforderungsergebnissen` über die jeweiligen Zugangsstrukturen aus.

Die Klasse `Patient` besteht aus der `Fieberkurve`, der `Signalleiste` und einer `Hilfe`, wobei diese Klassen den gleichnamigen Konzeptklassen entsprechen und hier nicht erneut vorgestellt werden.

Die Klasse ist Teil der navigationellen Kontexte `Patientenkontext` und `Patientenanforderungskontext`.

Kommentar zu Kommentar: An dieser Stelle werden die Charakteristika der Klasse ausführlich erläutert.

Trace fwd:

- ADV `Patient` (Schwesternsicht)
- ADV `Neuaufnahme` (Schwesternsicht)

Kommentar zu Trace fwd: Hier werden die zu dieser Klasse definierten ADVs des abstrakten Schnittstellenentwurfs aufgezählt. Für die Navigationsklasse `Patient` sind dies die oben genannten ADVs `Patient` und `Neuaufnahme` der Schwesternsicht.

Trace bck:

- `Patient`

Kommentar zu Trace bck: An dieser Stelle werden die Konzeptklassen aufgelistet, aus denen diese Klasse hervorgeht. Hier ist es die gleichnamige Klasse.

4.3.1.3 Verknüpfung AnforderungPatient

Verknüpfungsname: AnforderungPatient

Erbt von:

Attribute:

Besteht aus:

- *AktuellePatientenanforderung*
- 1-n *Patientenanforderungsdaten*

Kommentar zu Besteht aus: An dieser Stelle werden direkt der Verknüpfung zugeordnete Klassen vorgestellt. Im Falle der hier vorgestellten Verknüpfung `AnforderungPatient` sind dies die Klassen `AktuellePatientenanforderung` und `Patientenanforderungsdaten`. Im allgemeinen ist dies eine Teilmenge der entsprechenden Klassen der korrespondierenden Beziehung des Konzeptentwurfs.

Methoden:

- lesePatientenanforderungsdaten()

- schreibeAktuellePatientenanforderung()
- sortierePatientenanforderungsdaten()

Kommentar zu Methoden: Hier werden von der Verknüpfung benutzten Methoden erwähnt. Ähnlich wie bei den Bestandteilen ist dies eine Teilmenge der entsprechenden Methoden der korrespondierenden Beziehung des Konzeptentwurfs. In dem hier beschriebenen Beispiel sind dies die obengenannten Methoden.

Quellklasse: *Patient*

Kommentar zu Quellklasse: Dies ist die Klasse, von der die Verknüpfung ausgeht. Prinzipiell sind alle Verknüpfungen dieses Entwurfs bidirektional, sofern sie nicht ausdrücklich als Einweg-Verknüpfung deklariert sind. Hier ist die Klasse `Patient` Quellklasse der Verknüpfung `AnforderungPatient`.

Zielklasse: *Anforderung Patient*

Kommentar zu Zielklasse: Siehe Kommentar zu Quellklasse. Die Zielklasse für die Verknüpfung `AnforderungPatient` ist die Klasse Anforderung Patient.

Kardinalität: 1-n

Kommentar zu Kardinalität: Die Kardinalität legt fest, ob eine Instanz der Quellklasse mit einer oder 1-n Instanzen der Zielklasse verknüpft ist. Hier sind es 1-n Instanzen von `Anforderung Patient`, die mit einem `Patienten` verknüpft sind.

Zugangsstrukturen:

- *PatientenanforderungsAuswahl*

Kommentar zu Zugangsstrukturen: An dieser Stelle werden Zugangsstrukturen, die die Auswahl einer 1-n-Verknüpfung konkretisieren, aufgezählt. Hier ist es die Zugangsstruktur `PatientenanforderungsAuswahl`, die die Menge der vom `Patienten` erreichbaren `Anforderung Patient` beschreibt.

Navigationelle Kontexte:

- *Patientenkontext*
- *Patientenanforderungskontext*

Kommentar zu Navigationelle Kontexte: Hier werden die navigationellen Kontexte aufgezählt, von denen die Verknüpfung Teil ist, oder zwischen denen sie einen Übergang darstellt. Die hier beschriebene Verknüpfung `AnforderungPatient` stellt einen Übergang zwischen den Kontexten `Patientenkontext` und `Patientenanforderungskontext` dar.

Kommentar: Die Verknüpfung `AnforderungPatient` stellt die Navigation zwischen dem `Patienten` und einer `Patientenanforderung` dar. Die Bestandteile und Methoden wurden von der Konzeptbeziehung `AnforderungPatient` unverändert übernommen.

Die Verknüpfung wird durch die Zugangsstruktur `PatientenanforderungsAuswahl` realisiert.

Die Verknüpfung realisiert den Übergang vom `Patientenkontext` zum `Patientenanforderungskontext`.

Trace bck:

- `AnforderungPatient`

Kommentar zu Trace bck: Hier wird die Beziehung des Konzeptentwurfs erwähnt, von der diese Verknüpfung abgeleitet wurde. Hier ist es die gleichnamige Beziehung.

4.3.1.4 Zugangsstrukturen

Die Zugangsstrukturen `StationsanforderungsAuswahl` und `PatientenanforderungsAuswahl` sind Indexe aller möglichen `Anforderungen Station` oder `Anforderungen Patient` und ermöglichen die Auswahl einer dieser `Anforderungen` aus einer der Listen `Stationsanforderungsdaten` oder `Patientenanforderungsdaten`. Die Auswahl wird in den Objekten `AktuelleStationsanforderung` oder `AktuellePatientenanforderung` festgehalten. Diese Zugangsstrukturen realisieren die Beziehungen `AnforderungStation` und `AnforderungPatient`.

Die Zugangsstrukturen `StationsanforderungsergebnisAuswahl` und `PatientenanforderungsergebnisAuswahl` sind ebenfalls Indexe und realisieren eine Auswahl aus vorliegenden `Anforderungsergebnissen`. Sie beinhalten alle `A` mit `A` Instanz von `Anforderungsergebnis`, `A.Besitzer` = `AktuelleStation.ID` oder `A.Besitzer` = `AktuellerPatient.ID`. Die Auswahl verläuft analog zu den Zugangsstrukturen `StationsanforderungsAuswahl` und `PatientenanforderungsAuswahl`. Diese Zugangsstrukturen realisieren die Beziehungen `AnforderungsergebnisStation` und `AnforderungsergebnisPatient`.

Die Zugangsstruktur `PatientenAuswahl` stellt eine Auswahl aus einer Liste aller `Patienten` einer `Station` dar. Sie beinhaltet alle `P` mit `P` Instanz von `Patient` und `P.Station` = `Station.ID`. Die Auswahl verläuft analog zu den obengenannten Zugangsstrukturen. Diese Zugangsstruktur realisiert die Verknüpfungen `Belegung` und `Stationsnachbar`.

Die Zugangsstruktur `ExterneDokumentenAuswahl` ermöglicht die Auswahl eines Dokuments, welches mit einer externen Anwendung erstellt wurde. Das Verhalten ist analog zur Zugangsstruktur `PatientenAuswahl`.

Die Zugangsstruktur `PatientenanforderungsAuswahl` wird im folgenden genauer vorgestellt.

4.3.1.5 Zugangsstruktur PatientenanforderungsAuswahl

Klassenname: PatientenanforderungsAuswahl

Typ: Index

Kommentar zu Typ: An dieser Stelle wird der Typ der Zugangsstruktur definiert. Die in dieser Arbeit verwendeten Typen sind *Index*, welcher eine gleichzeitige Präsentation aller navigierbaren Knoten bedingt, und *Guided Tour*, welcher ein sequentielles Abschreiten der erreichbaren Elemente impliziert. Der Typ der Zugangsstruktur `PatientenanforderungsAuswahl` ist ein Index.

Erbt von:

Verknüpfungen:

- `AnforderungPatient`
- `TerminplanPatientUndAnforderungPatient`
- `AnforderungsergebnisPatientUndAnforderungPatient`
- `PflegeUndAnforderungPatient`
- `TherapieUndAnforderungPatient`
- `DiagnostikUndAnforderungPatient`

Kommentar zu Verknüpfungen: Eine Zugangsstruktur spezifiziert immer eine oder mehr Verknüpfungen. Diese werden hier aufgezählt. Im Falle der hier präsentierten Zugangsstruktur sind dies die oben genannten Verknüpfungen.

Selektoren: eine neue Instanz jeder Zielklasse, alphabetisch geordnet

Kommentar zu Selektoren: Hier werden Selektoren aufgeführt, die die Instanzen der Zielklasse der Verknüpfung, die diese Zugangsstruktur spezifiziert, auswählen. Ferner wird eine Reihenfolge festgelegt. In diesem Fall ist es jeweils eine Instanz jeder Zielklasse der Verknüpfung.

Zielklasse:

- `Essensanforderung Patient`
- `Laboranforderung Patient`
- `Apothekenanforderung Patient`
- `EKG-Anforderung`
- `Nuklearmedizinanforderung`
- `Sonografieanforderung`
- `Lungenfunktionsprüfung`
- `Röntgenanforderung`
- `OP/Anästhesieanforderung`
- `Anforderung Physikalische Therapie`
- `Konsiliarschein`

Kommentar zu Zielklasse: Hier werden die Zielklassen aufgeführt, die über die Zugangsstruktur erreicht werden können. In diesem Beispiel sind die obengenannten Zielklassen alle Unterklassen der Navigationsklasse `Anforderung Patient`.

Navigationelle Kontexte:

- `Patientenanforderungskontext`

Kommentar zu Navigationelle Kontexte: Hier werden navigationelle Kontexte, zu denen die Zugangsstruktur gehört oder die sie definiert, vorgestellt. Die hier präsentierte Zugangsstruktur `PatientenanforderungsAuswahl` definiert den Kontext `Patientenanforderungskontext`.

Kommentar:

Die Zugangsstruktur `PatientenanforderungsAuswahl` besteht aus jeweils einer neuen Instanz der elf, unter Zielklassen aufgelisteten Unterklassen der `Anforderung Patient`.

Die Zugangsstruktur ist den Verknüpfungen `AnforderungPatient`, `TerminplanPatientUndAnforderungPatient`, `AnforderungsergebnisPatientUndAnforderungPatient`, `PflegeUndAnforderungPatient`, `TherapieUndAnforderungPatient` und `DiagnostikUndAnforderung-Patient` zugeordnet.

Die Zugangsstruktur realisiert den Kontext `Patientenanforderungskontext`.

4.3.1.6 Navigationelle Kontexte

Für die `Schwesternsicht` wurden die Kontexte `Stationskontext`, `Patientenkontext`, `Neuaufnahmekontext`, `Stationsanforderungskontext`, `Patientenanforderungskontext` und `Externer Dokumentenkontext` festgelegt. Abbildung 4.10 stellt diese Kontexte dar. Die Pfeile zwischen den Kontexten stellen Verküpfungen dar, die zwischen Elementen verschiedener Kontexte navigierbar sind. Ein Rechteck mit abgerundeten Ecken gibt den Namen einer Zugangsstruktur an, die eine Verknüpfung spezifiziert. Einzelheiten werden in diesem Abschnitt beschrieben.

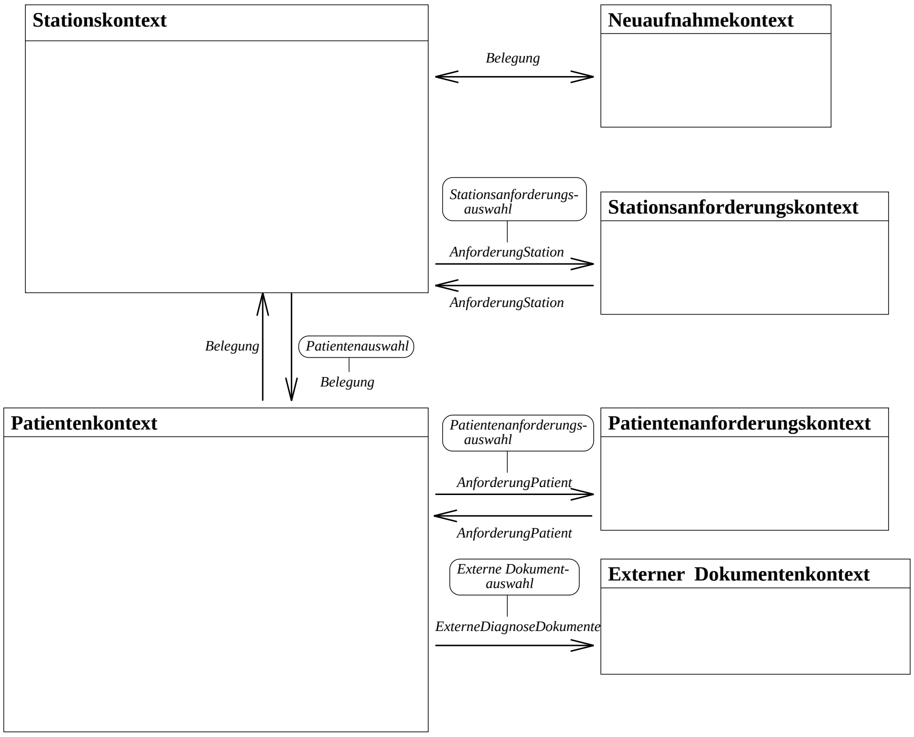


Abbildung 4.10: Die Kontexte der Schwesternsicht

Anmerkung: Alle Elemente der Schwesternsicht sind Teil eines Kontexts. Das bedeutet, daß für die meisten Knotenklassen ein oder mehrere Kontextklassen existieren, die zusätzliche Anker für nur in den entsprechenden Kontexten existierenden Verknüpfungen definieren. Dieses Vorgehen ist hier sinnvoll, da in den beiden wichtigsten Kontexten `Stationskontext` und `Patientenkontext` alle Elemente mit allen anderen Elementen und Ausgängen verknüpft sind.

Der `Stationskontext` umfaßt die Klassen `Station, Schwester, Terminplan Station, Terminplan Schwester` und `Anforderungsergebnis Station`. Der Eingang dieses Kontexts ist die Klasse `Station`. Der Kontext wird im Falle einer Neuaufnahme über die Verknüpfung `Belegung`, im Falle einer Patientenauswahl über die diese Verknüpfung realisierende Zugangsstruktur `PatientenAuswahl` verlassen. Ein weiterer Ausgang wird durch die Zugangsstruktur `StationsanforderungsAuswahl` dargestellt. Die Zugangsstruktur `StationsanforderungsergebnisAuswahl` realisiert eine 1-zu-n-Verknüpfung innerhalb des Kontexts. Dieser Kontext realisiert eine Gruppierung aller die `Station` direkt betreffenden Elemente der Sicht. Innerhalb des Kontexts sind alle Elemente und Ausgänge von allen Elementen aus erreichbar. Dies wird im nächsten Schritt (dem Abstrakten Schnittstellenentwurf) durch eine Navigationsleiste realisiert, die bei allen Knoten des Kontexts präsent ist. Diese Navigationsleiste dient als ständig präsente Orientierungshilfe und Übersicht für die das System benutzende Schwester.

Der `Patientenkontext` beinhaltet die Klassen `Patient, Terminplan Patient, Diagnostik, Pflege, Therapie` und `Anforderungsergebnis Patient`. Eingang dieses Kontexts ist die Klasse `Patient`. Ausgänge werden durch die Verknüpfung `Belegung` und die Zugangsstrukturen `PatientenanforderungsAuswahl` und `ExterneDokumentAuswahl` festgelegt. Die Zugangsstrukturen `PatientenAuswahl` und `PatientenanforderungsergebnisAuswahl` stellen 1-zu-n-Verknüpfungen innerhalb des Kontexts dar. Innerhalb des Kontexts sind alle Elemente und Ausgänge von allen Elementen aus erreichbar, mit Ausnahme des `Externen Dokumentenkontexts`, der nur von der `Diagnostik` aus erreichbar ist. Dieser Kontext realisiert eine Gruppierung aller den `Patienten` direkt betreffenden Elemente der Sicht. Die Verknüpfungen dieses Kontexts werden ebenfalls durch eine im nächsten Schritt definierte Navigationsleiste dargestellt.

Der `Stationsanforderungskontext` beinhaltet alle existierenden `Anforderungen Station` (`Materialanforderung, Laboranforderung, Apothekenanforderung` und `Essensanforderung`). Die Elemente des Kontexts können über die Zugangsstruktur `StationsanforderungsAuswahl` erreicht werden. Innerhalb dieses Kontexts ist nur der direkte Rücksprung zum `Stationskontext` möglich. Dieser Kontext besitzt nur die direkte Rücksprungmöglichkeit zur `Station`, weil der Benutzer während der Ausfertigung einer `Anforderung` diese erst fertigstellen oder den Bearbeitungsprozeß abbrechen soll, bevor er zu einem anderen Knoten navigiert.

Der `Patientenanforderungskontext` beinhaltet alle existierenden `AnforderungenPatient` (`EKG-Anforderung, Konsiliarschein, Röntgenanforderung, Nuklearmedizinanforderung, Sonografieanforderung, Lungenfunktionsprüfung, Anforderung physikalische Therapie, OP/Anästhesieanforderung, Apothekenanforderung Patient, Essensanforderung Patient` und `Laboranforderung Patient`). Die Elemente des Kontexts können über die Zugangsstruktur `PatientenanforderungsAuswahl` erreicht werden. Innerhalb dieses Kontexts ist nur der direkte Rücksprung zum `Patientenkontext` möglich. Dieser Kontext besitzt nur die direkte Rücksprungmöglichkeit zum `Patienten`, weil der Benutzer während der Ausfertigung einer `Anforderung` diese erst fertigstellen oder den Bearbeitungsprozeß abbrechen soll, bevor er zu einem anderen Knoten navigiert.

Der `Neuaufnahmekontext` umfaßt eine neue Instanz der Klasse `Patient`. Er wird im Falle einer Neuaufnahme über die Beziehung `Belegung` erreicht. Von hier ist nur der direkte Rücksprung zum `Stationskontext` möglich. Ähnlich wie im `Stationsanforderungskontext` und im `Patientenanforderungskontext` soll der Benutzer den Bearbeitungsprozeß beenden, bevor er zu einem anderen Knoten navigiert.

Der `Externe Dokumentenkontext` beinhaltet alle `Externen Diagnosedokumente`, die zum aktuellen `Patienten` gehören, d.h. alle `D` mit `D` Instanz von `Externes Dokument`, `D.Besitzer` = `AktuellerPatient.ID`. Von hier ist kein Rücksprung zum `Patientenkontext` möglich. Dies bedeutet konkret, daß ein externes Dokument in einem neuen Fenster angezeigt werden soll, welches vom laufenden System der hypermedialen Patientenakte unabhängig ist.

Hier wird nun der Kontext `Patientenanforderungskontext` näher vorgestellt.

4.3.1.7 Navigationeller Kontext Patientenanforderungskontext

Klassenname: Patientenanforderungskontext

Erbt von:

Oberklasse von:

Beinhaltet:

- Klasse `Anforderung Patient` mit den Unterklassen
  - `Essensanforderung Patient`
  - `Apothekenanforderung Patient`
  - `Laboranforderung Patient`
  - `EKG-Anforderung`
  - `Nuklearmedizinanforderung`
  - `Sonografieanforderung`
  - `Lungenfunktionsprüfung`
  - `Röntgenanforderung`
  - `OP/Anästhesieanforderung`

- `Anforderung Physikalische Therapie`
- `Konsiliarschein`

Kommentar zu Beinhaltet: An dieser Stelle werden Elemente des Kontexts, die Navigationsklassen, Zugangsstrukturen und Verknüpfungen sein können, vorgestellt. Verknüpfungen werden aus Platzgründen nicht genannt, sondern es wird festgelegt, daß alle Verknüpfungen zwischen Navigationsklassen, die Element des Kontexts sind, ebenfalls diesem Kontext angehören. Im Falle des hier vorgestellten Kontexts `Patientenanforderungskontext` sind alle Unterklassen der Navigationsklasse `Anforderung Patient` Element dieses Kontexts.

Realisierte Zugangsstruktur: `PatientenanforderungsAuswahl`

Kommentar zu Realisierte Zugangsstruktur: Hier wird die dem Kontext zugrundeliegende Zugangsstruktur erwähnt. Bei dem hier vorgestellten Beispiel ist es die Zugangsstruktur `PatientenanforderungsAuswahl`.

Eingangsklasse:

- `Anforderung Patient`

Kommentar zu Eingangsklasse: Die Eingangsklasse eines Kontexts ist die Klasse, zu der navigiert wird, wenn der Kontext betreten wird. Im Fall des hier vorgestellten Beispielkontexts ist dies die Klasse `Anforderung Patient` oder eine ihrer Unterklassen.

Ausgang:

- `AnforderungPatient`

Kommentar zu Ausgang: An dieser Stelle werden Verknüpfungen aufgezählt, bei deren Navigation der Kontext verlassen wird. In diesem Beispiel ist es die Verknüpfung `AnforderungPatient`, die aus dem Kontext hinausführt.

Navigation: Direkte Anwahl eines Elements, nur direkte Navigation zurück möglich.

Kommentar zu Navigation: Hier werden die Navigationsregeln für den beschriebenen Kontext formuliert. In diesem Fall sind alle Elemente des Kontexts direkt anwählbar und es ist von einem Element des Kontexts nur der direkte Rücksprung zum Element möglich, von welchem aus zu ihm navigiert wurde.

Kommentar: Der Navigationelle Kontext `Patientenanforderungskontext` realisiert die Zugangsstruktur `PatientenanforderungsAuswahl` und umfaßt die Klasse `Anforderung Patient` mit ihren Unterklassen.

Alle Elemente des Kontexts sind direkt anwählbar, es ist aber von dort nur der direkte Rücksprung zum `Patienten` möglich.

Der Eingang in den Kontext erfolgt über die Klasse `Anforderung Patient`.

Der Kontext wird über die Verknüpfung `AnforderungPatient` in Richtung `Patient` verlassen.

Kommentar zu Kommentar: Hier werden die Eigenschaften des Kontexts näher beschrieben.

4.3.1.8 Kontextklassen

Es existieren für alle Knotenklassen Kontextklassen, die die kontextspezifischen zusätzlichen Anker der hinzugekommenen Verknüpfungen dokumentieren. Im Falle der `Station` ist dies beispielsweise ein Anker, der eine Verknüpfung zum `Terminplan Schwester` definiert. Bei der `Anforderung Patient` ist es ein zweiter Anker, der zurück zur `Station` führt. Dies drückt die Tatsache aus, daß eine `Anforderung` entweder erstellt oder der Prozeß abgebrochen wurde, wobei ein Anker einen Abbruch des Prozesses vor der Fertigstellung auslöst und der andere eine neue Instanz der

`Anforderung Patient` anlegt. Dieses Verhalten wird während des Abstrakten Schnittstellenentwurfs näher spezifiziert. Ein weiteres Beispiel ist die Klasse `Terminplan Patient`. Diese Klasse hat im Navigationsentwurf der Schwesternsicht nur eine Beziehung zum `Patienten`. Da im `Patientenkontext` alle Elemente mit allen anderen Elementen und Ausgängen verknüpft sind, müssen zusätzliche Anker und Verknüpfungen zu den Klassen `Anforderungsergebnis Patient, Pflege, Therapie` und `Diagnostik` (Elemente des Kontexts) sowie den außerhalb des Kontexts liegenden Klassen `Station` und `Anforderung Patient` definiert werden.

Beispielhaft wird hier die Kontextklasse `Patient im Neuaufnahmekontext` vorgestellt.

4.3.1.9 Kontextklasse Patient im Neuaufnahmekontext

Klassenname: Patient im Neuaufnahmekontext

Attribute:

- AbbruchStation: Anker (`Belegung`)

Kommentar zu Attribute: An dieser Stelle werden in dem entsprechenden Kontext hinzugefügte Attribute und Anker präsentiert. In dem hier vorgestellten Beispiel existiert ein zusätzlicher Anker.

Gehört zu Klasse: `Patient`

Kommentar zu Gehört zu Klasse: Hier wird die Klasse, die die Kontextklasse erweitert, genannt. Hier ist es die Navigationsklasse `Patient`.

Kontext: `Neuaufnahmekontext`

Kommentar zu Kontext: An dieser Stelle wird der Kontext, für den diese Klasse relevant ist, aufgeführt. Hier ist es der `Neuaufnahmekontext`.

Kommentar: Diese Kontextklasse definiert für die Klasse `Patient` im `Neuaufnahmekontext` einen zusätzlichen Anker. Der Anker `AbbruchStation` wurde eingeführt, um wie über den Anker `Belegung` zur Station navigieren zu können. Er existiert, weil eine Neuaufnahme eines `Patienten` abgebrochen oder vollständig durchgeführt werden kann. Es werden zwei Anker benötigt, um hier zu differenzieren. Die genaue Funktionalität wird im abstrakten Schnittstellenentwurf festgelegt.

4.3.1.10 Navigationelle Transformationen

Im allgemeinen sind die navigationellen Transformationen der hypermedialen Patientenakte zwischen Knoten trivial, d.h. jeder Aggregationsknoten kann nur exklusiv angezeigt werden. Eine Ausnahme hiervon bildet der Knoten `Externes Dokument`. Da ein `Externes Dokument` kein integrierter Teil des Gesamtssystems ist, soll das System dieses in einem eigenen externen Fenster anzeigen, welches zusätzlich zu allen anderen Fenstern existieren kann.

Festgelegt werden an dieser Stelle die Transformationen innerhalb einer Aggregation. Die Transformationen des Knotens `Patient` sind in Abbildung 4.11 wiedergegeben. Die Klasse `Signalleiste` ist ständig zu sehen. Die Klassen `Patient` und `Fieberkurve` werden exklusiv angezeigt. Die Klasse `Hilfe` kann von `Patient` oder `Fieberkurve` aus angesteuert werden und wird zusätzlich zu einer der beiden angezeigt. Bei Verlassen des Aggregationsknotens werden alle Teile des Knotens ausgeblendet.

4.3.2 Arztsicht

Die *Arztsicht* unterscheidet sich nur in geringen Einzelheiten von der *Schwesternsicht*. Aus diesem Grund werden hier nur die wichtigsten Unterschiede erläutert.

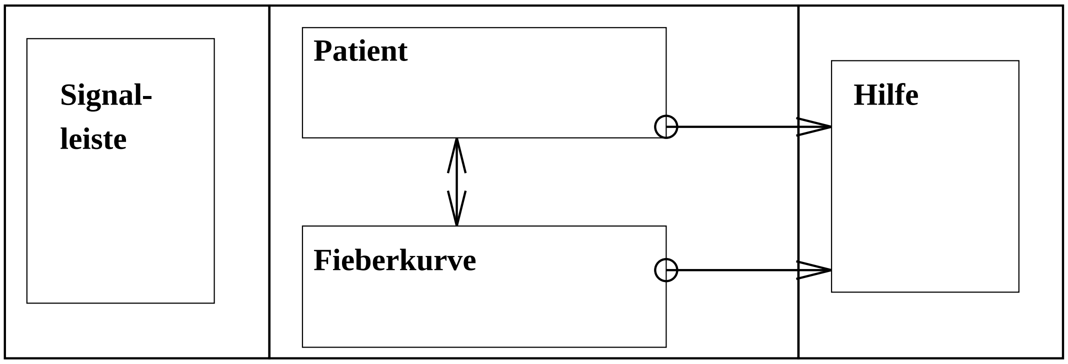


Abbildung 4.11: Die navigationellen Transformationen des Aggregationsknotens Patient

Die Verknüpfungen `AnforderungStation` und `AnforderungPatient` werden hier durch jeweils zwei Zugangsstrukturen realisiert: `PatientenanforderungsAuswahl` bzw. `StationsanforderungsAuswahl` und `ZuBearbeitendeAnforderungsAuswahl`. Letztere beinhaltet alle `Anforderungen`, die zwar schon von einer Schwester erstellt wurden, aber noch nicht vom Arzt begutachtet und verschickt wurden, geordnet nach Datum. Alle die `Schwester` betreffenden Klassen sind durch die entsprechenden Klassen des `Arztes` ersetzt worden. Weiterhin kann der Arzt im Knoten `Diagnostik` eine `Entlassung` anordnen. Als letztes kann der Arzt die Teile der Konzeptklasse `Medizinische Datenbank`, die den Knoten `Diagnostik, Pflege` und `Therapie` zugeordnet sind, editieren.

Die Arztsicht beinhaltet einen Kontext, der in der Schwesternsicht nicht vorkommt. Dies ist der `Zu Bearbeitende Anforderungskontext`. Dieser Kontext beinhaltet die durch die Zugangsstruktur `ZuBearbeitendeAnforderungsauswahl` festgelegten `Anforderungen`. Navigation ist entweder zurück zum `Patienten`, zur `Station` oder zum nächsten/letzten Element des Kontexts möglich.

4.3.3 Verwaltungssicht

Die hier dargestellte Verwaltungssicht soll keine vollständige Spezifikation der Funktionalität einer Krankenhausverwaltung widerspiegeln. Stattdessen dient sie zur Darstellung eines Aspekts der Patientenakte, nämlich der Abrechnung. Die Verwaltungssicht umfaßt das Subsystem `Verwaltungs-/Abrechnungssystem` des Konzeptentwurfs (vergleiche Abbildungen 4.2 und 4.7).

Die Knotenklasse `Verwaltung` besitzt zwei Anker, die jeweils eine 1-n-Beziehung zu den Knotenklassen `Abrechnung Patient` und `Abrechnung Station` festlegen. Die Knotenklasse `Abrechnung Patient` besteht aus mehreren Attributen und Ankern zur `Verwaltung` und zur `Abrechnung Station`. Die `Abrechnung Station` besitzt einen Anker zur `Verwaltung` und einen zu den zugeordneten 1-n `Abrechnungen Patient`, die in der Konzeptsicht der `Abrechnung Station` direkt als Aggregationselemente zugeordnet waren. Die 1-zu-n-Verknüpfung `BerechnungAbrechnungPatient` zwischen `Verwaltung` und `Abrechnung Patient` wird durch die Zugangsstruktur `AlleStationen-Patientenauswahl` konkretisiert. Diese Zugangsstruktur ist ein verschachtelter Index, bei dem erst die `Station` und dann ein darauf liegender `Patient` ausgewählt wird. Die Auswahl der Objekte `AktuelleStation` und `AktuellerPatient` erfolgt durch Auswahl aus den Listen `Stationsdaten` und `Patientendaten`, die der Verknüpfung `BerechnungAbrechnungPatient` zugeordnet sind. Die Verknüpfungen `BerechnungAbrechnungStation` werden durch ähnliche einstufige Indexe realisiert, die auf dieselben Datenobjekte zugreifen.

Der Kontext `Allgemeiner Abrechnungskontext` beinhaltet die Knotenklassen `Verwaltung, Abrechnung Patient` und `Abrechnung Station`. Das Navigationsverhalten in diesem Kontext ist frei, das heißt, daß alle Elemente des Kontexts von jedem anderen Element aus erreicht werden können, mit Ausnahme des Ausgangsindexes `AlleElementeEinerStationAuswahl`, welcher nur von der `Abrechnung Station` aus erreicht werden kann.

Der Kontext `Patientenabrechnung als Teilabrechnungskontext` legt das Navigationsverhalten für die `Abrechnung Patient` fest. Dies bedeutet konkret, daß es dem Benutzer möglich ist, sich die einzelnen `Abrechnungen Patient`, die Teil einer `Abrechnung Station` sind, von dieser aus anzuzusehen, und von dort aus entweder zur zugehörigen `Abrechnung Station` zurückzugelangen oder die nächste oder vorhergehende `Abrechnung Patient` dieses Kontexts zu betrachten. Dies setzt voraus, daß die Elemente dieses Kontextes, alle Instanzen von `Abrechnung Patient` mit `AP` = Instanz von `Abrechnung Patient` und `AP.PatientenID` = `AktuellerPatient.ID` geordnet sind, welches in alphabetischer Reihenfolge der Fall ist. Diese Knotenklasse ist somit Teil zweier verschiedener Kontexte, wobei die angezeigten Navigationsmöglichkeiten vom Kontext abhängen.

4.3.4 Leistungsstellensicht

Die Sicht der Leistungsstelle, wie sie in dieser Diplomarbeit behandelt wird, reflektiert nicht den vollständigen Aufgabenbereich einer Leistungsstelle. Es wird der Bereich wiedergegeben, der sich mit dem Empfang einer Anforderung und dem Versenden eines Anforderungsergebnisses sowie der Terminplanung beschäftigt. Es existiert für jede der `Leistungsstellen Innere Abteilung, Nuklearmedizin, Abteilung OP/Anästhesie, Abteilung Physikalische Therapie, Radiologie, Apotheke, Labor, Küche, Materialstelle` und `Externe Stelle` eine eigene Sicht. Aus Übersichtlichkeitsgründen wird hier nur eine allgemeine Sicht für die Oberklasse `Leistungsstelle` vorgestellt. Die Leistungsstellensicht umfaßt die Subsysteme `Leistungsstellensystem` und `Anforderungssystem` (siehe Abbildung 4.2).

Die Knotenklasse `Leistungsstelle` besitzt Anker zu den Klassen `Terminplan Leistungsstelle, Anforderung` und `Anforderungsergebnis`. Die Verknüpfungen `Bearbeitung` und `Ergebnis` zu den beiden letztgenannten Klassen sind 1-zu-n-Verknüpfungen, die durch Zugangsstrukturen konkretisiert werden. Der `Terminplan Leistungsstelle` besitzt Anker zur `Leistungsstelle` und zur `Anforderung`, wobei die 1-zu-n-Verknüpfung zur letztgenannten Klasse ebenfalls durch eine Zugangsstruktur konkretisiert wird. Ferner sind die Klassen `Anforderung` und `Anforderungsergebnis` durch eine Verknüpfung `GehörtZu` verbunden.

Die Verknüpfung `Bearbeitung` zwischen der `Leistungsstelle` und 1-zu-n `Anforderungen` wird durch zwei Zugangsstrukturen realisiert, nämlich die Indexe `TerminierteAnforderungsAuswahl` und `UnterminierteAnforderungsAuswahl`. Diese unterscheiden sich durch ihre Elemente, die im ersten Fall aus allen Instanzen der Klasse `Anforderung` bestehen, für die gilt: `A` sei Instanz von `Anforderung`; `A.Status` = 'vereinbart'. Im zweiten Fall muß `A.Status` = 'vorgeschlagen' sein. Die Auswahl der Elemente erfolgt aus der der Verknüpfung `Bearbeitung` zugeordnete Liste `ZuBearbeitendeAnforderungsdaten`. Die Auswahl wird im Objekt `AktuelleZuBearbeitendeAnforderung` gespeichert.

Im Falle der der Verknüpfung `Ergebnis` zugeordneten Zugangsstruktur `ErgebnisAuswahl` sieht es ähnlich aus, mit dem Unterschied, daß nur `Anforderungsergebnisse`, für die gilt: `AE` sei Instanz von `Anforderungsergebnis`, `AE.GehörtZuID` = `A.ID` mit `A` Instanz von `Anforderung` und `A.Status` = 'vereinbart'. Das heißt, daß ein `Ergebnis` nur für eine bereits terminierte `Anforderung` erstellt werden darf.

Die die Verknüpfung `Terminierung` realisierende Zugangsstruktur `TerminplanAnforderungsauswahl` beinhaltet alle existierenden Instanzen von `Anforderung`, die über den `Terminplan` direkt angewählt werden können.

Die beiden Kontexte der Sicht sind der `Leistungsstellenkontext` und der `Ergebnisbearbeitungskontext`. Der `Leistungsstellenkontext` umfaßt die Knotenklassen `Leistungsstelle, Terminplan Leistungsstelle` und `Anforderung`. Innerhalb dieses Kontexts ist freie Navigation zwischen allen Elementen möglich. Ausgänge sind die Zugangsstruktur `Ergebnisauswahl` und die Verknüpfung `GehörtZu`, die in den `Ergebnisbearbeitungskontext` führen. Letztere ist nur von der `Anforderung` aus erreichbar.

Der `Ergebnisbearbeitungskontext` beinhaltet alle durch die `ErgebnisAuswahl` festgelegten `Anforderungsergebnisse`. Hier ist nur die Navigation zur `Leistungsstelle` oder zur zugehörigen

`Anforderung` möglich.

4.3.5 Besuchersicht

Wie oben schon erwähnt, ist die Besuchersicht als Informationssystem über das Krankenhaus konzipiert. Es werden das Krankenhaus selbst, die einzelnen Leistungsstellen und Stationen präsentiert, wobei jede Station zusätzlich die darauf arbeitenden Ärzte und Schwestern vorstellt.

Alle Klassen der Besuchersicht beinhalten als Attribute hauptsächlich Präsentationsinformationen. Die Klasse `Krankenhaus` besitzt Anker zu den Klassen `Leistungsstelle` und `Station`. Die 1-zu-n-Verknüpfungen zu diesen beiden Klassen werden durch Zugangsstrukturen realisiert. Die Knotenklassen `Leistungsstelle` und `Station` sind mit der Klasse `Krankenhaus` verknüpft. Die Klasse `Station` hat weiterhin Anker zu den Knotenklassen `Arzt` und `Schwester`, die durch Zugangsstrukturen spezifiziert werden. Dies stellt die Beziehung zu dem auf einer `Station` arbeitenden Personal dar. Diese Beziehungen wurden aus dem Grund festgelegt, um vor allem dem Patienten, der diese Sicht als Informationssicht benutzen soll, die Möglichkeit zu geben, sich über das Personal auf seiner Station zu informieren und um dem Personal die Möglichkeit zu geben, sich auf eine persönliche Weise darzustellen.

Es existieren die drei Kontexte `Krankenhausinformationenkontext, Stationsinformationenkontext` und `Krankenhaustourkontext`.

Der `Krankenhauskontext` beinhaltet die Knotenklassen `Krankenhaus, Station` und `Leistungsstelle` und verknüpft alle Elemente mit allen anderen. Er kann über die Zugangsstruktur `Krankenhaustour` verlassen werden. Bei der Navigation zur `Station` wird der Unterkontext `Stationsinformationenkontext` betreten, der alle Elemente und Eigenschaften des `Krankenhausinformationenkontexts` beinhaltet. Dazu kommen die Knotenklassen `Arzt` und `Schwester` als Elemente. Die Auswahl der einzelnen `Leistungsstellen, Stationen, Ärzte` und `Schwestern` geschieht über Zugangsstrukturen in Form von Indexen.

Der `Krankenhaustourkontext` ist eine geführte Tour durch alle `Leistungsstellen` und `Stationen` des `Krankenhauses`. Er besteht aus einer sortierten Liste aller `Leistungsstellen` und `Stationen`, die erst alle alphabetisch sortierten `Leistungsstellen` und anschließend alle alphabetisch sortierten `Stationen` beinhaltet. Es kann zum jeweils nächsten oder letzten Element oder zum Eingangsknoten `Krankenhaus` navigiert werden.

4.4 Schnittstellenentwurf

In diesem Abschnitt wird beschrieben, wie das abstrakte Schnittstellenmodell der hypermedialen Patientenakte erstellt wurde. Es erfolgt zunächst eine Vorstellung der Notation. Anschließend werden beispielhaft die ADVs `Patient` und `Navigationsleiste Patient` der Schwesternsicht präsentiert.

Die Dokumentation jeder ADV erfolgt nach folgendem Schema:

Einleitung

Hier wird in kurzen Worten eine Übersicht der beschriebenen ADVs gegeben.

Bestandteile

In diesem Abschnitt werden die Bestandteile einer ADV vorgestellt. Eine ADV kann aus weiteren ADVs und/oder Schnittstellenprimitiven bestehen. Die in dieser Arbeit verwendeten Schnittstellenprimitive werden im folgenden kurz vorgestellt.

Textfeld: Ein Textfeld stellt ein Attribut dar, dessen Wert aus alphanumerischen Zeichen zusammengesetzt ist. Dies sind im allgemeinen die Attribute der Typen `String, Date, Integer`

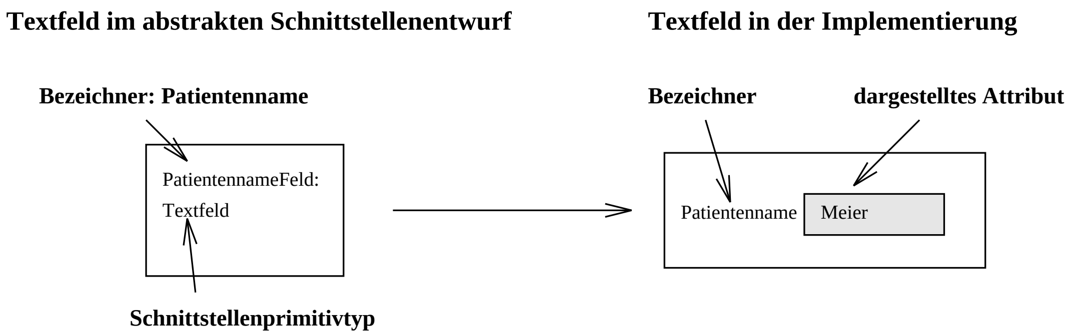


Abbildung 4.12: Textfeld im abstrakten Schnittstellenentwurf und in der Implementierung

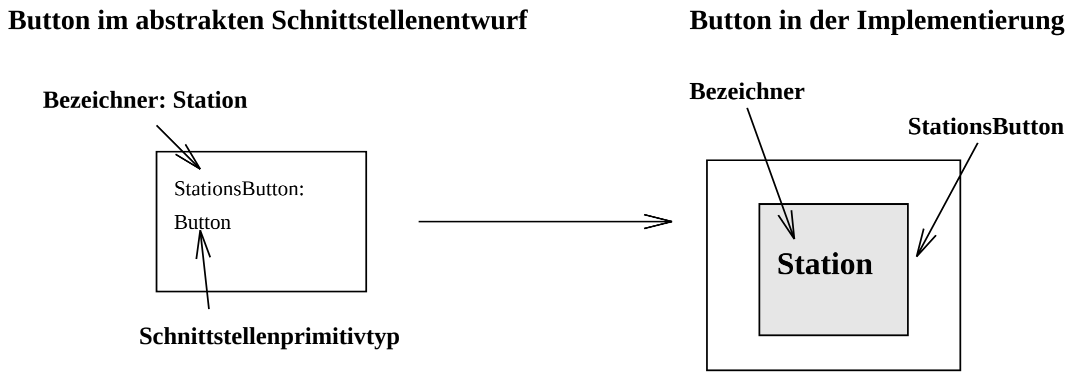


Abbildung 4.13: Button im abstrakten Schnittstellenentwurf und in der Implementierung

oder `Real`. Der Name eines Textfelds besteht im allgemeinen aus einem Bezeichner und der Endung 'Feld'. Jedem Textfeld ist ein Label zugeordnet, welches den Bezeichner anzeigt. Der Inhalt eines Textfelds ist editierbar, wenn die Methode `ermöglicheEingabe(Textfeld)` für dieses Textfeld durchführbar ist. Abbildung 4.12 stellt ein Textfeld im abstrakten Schnittstellenentwurf und in der Implementierung dar.

Button: Ein Button ist ein grafisches Element, welches auf das Ereignis Mausklick eine bestimmte Reaktion auslöst. Buttons spezifizieren Anker oder lösen durch Mausklick-Ereignisse Transformationen der Schnittstelle aus. Der Name eines Buttons besteht im allgemeinen aus einem Bezeichner und der Endung 'Button'. Jeder Button besitzt ein Attribut 'Farbe' und ein Label, welches den Namen des Buttons anzeigt. Dieses Label enstpricht im allgemeinen dem Bezeichner. Abbildung 4.13 stellt einen Button im abstrakten Schnittstellenentwurf und in der Implementierung dar.

Image: Ein Image ist ein Bild, welches in einem durch das System darstellbaren Format vorliegt. Ein Image kann ein Label besitzen, welches den Bezeichner beinhaltet.

ToggleButton: ToggleButtons werden im allgemeinen zu Gruppen zusammengefaßt. Wenn ein Ereignis `Mausklick` auf einem dieser Buttons stattfindet, wird dieser ToggleButton auf den Wert 'true' und alle anderen der Gruppe auf den Wert 'false' gesetzt. Der Wert 'true' wird durch einen schwarzen Punkt gekennzeichnet. Der Name eines ToggleButtons besteht aus einem Bezeichner, im Falle von genau zwei zusammenhängenden ToggleButtons einem weiteren Bezeichner 'Ja' oder 'Nein', und der Endung 'Button'. Jeder Gruppe von ToggleButtons und jedem ToggleButton selbst sind Labels zugeordnet, die im allgemeinen die Bezeichner darstellen.

ChoiceMenu: Ein ChoiceMenu ist ein auf- und zuklappbares Menü, welches eine Liste mit verschiedenen Attributen beinhaltet, die vom selben Typ wie die in einem Textfeld darstellbaren Attribute sein können. Ein ChoiceMenu reagiert im allgemeinen auf das Ereignis `Mausklick`. Wenn das ChoiceMenu im Zustand 'zu' ist, wird es durch einen Mausklick geöffnet, d.h. aufgeklappt und die Liste mit den dargestellten Attributen wird sichtbar. Ein Mausklick auf ein Element der Liste wählt dieses Element aus und klappt das ChoiceMenu zu. Der Name eines ChoiceMenus besteht im allgemeinen aus einem Bezeichner und der Endung 'Liste'. Jedes ChoiceMenu besitzt ein Label, welches im allgemeinen den Bezeichner darstellt.

Checkbox: Eine Checkbox ist ein Feld, welches zwei Zustände besitzt. Der Zustand 'true' wird in der Implementierung grafisch durch ein Kreuz oder einen Haken dargestellt. Dieser Schnittstellenprimitive wird benutzt, um die Eingabe von Attributen vom Typ Boolean zu ermöglichen. Der Name einer Checkbox besteht aus einem Bezeichnerstring und der Endung 'Box'. Jede Checkbox besitzt ein Label, welches im allgemeinen den Bezeichner darstellt. Eine Checkbox reagiert auf das Ereignis `Mausklick`, welches die Checkbox vom aktuellen auf den anderen Zustand setzt.

Fenster: Ein Fenster ist ein Feld, welches zusätzlich zu der aktuell angezeigen ADV angezeigt werden kann. Fenster sind als ADVs definiert, die weitere ADVs und/oder Schnittstellenprimitive beinhalten können. Wenn ein Fenster geöffnet wird, wird es über der aktuellen ADV angezeigt und kann mit der Maus verschoben werden. Im allgemeinen wird ein Fenster aus sich selbst heraus geschlossen, d.h. es beinhaltet einen Button, der die Methode `verbergeFenster()` aufruft und damit ausblendet. Der Name eines Fensters besteht im allgemeinen aus einem Bezeichner und der Endung 'Fenster'. Ein Fenster besitzt ein Label, welches in seinem Rahmen angezeigt wird. Dieses Label entspricht im allgemeinen dem Bezeichner.

Konfigurationsdiagramm

Konfigurationsdiagramme beschreiben die Beziehungen zwischen Ereignissen, die auf eine ADV einwirken, den Elementen der ADV und den Methoden des unterliegenden ADOs. Sie sind nach dem im folgenden erklärten Schema aufgebaut, welches in Abbildung 4.14 anschaulich gezeigt wird.

Auf der linken Seite befindet sich eine abstrakte Darstellung aller Elemente der zugehörigen ADV. Im Fall des Beispiels sind dies das Textfeld `AnmerkungenFeld` und der Button `ÜbernahmeButton`. Links davon werden die externen Ereignisse aufgeführt, die Methoden des unterliegenden ADO aufrufen. Durchgezogene Pfeile führen von den Ereignissen zu den Schnittstellenelementen, die auf sie reagieren. Hier sind dies die Ereignisse `Tastatureingabe`, auf welches das Textfeld reagiert, `Mausklick`, worauf sowohl der Button als auch das Textfeld reagieren, und `Anzeige`, auf das die ADV `AnmerkungenPflege` reagiert. Von den Schnittstellenelementen führen weitere, diesmal gestrichelte Pfeile zu einer auf der rechten Seite des Diagramms befindlichen abstrakten Darstellung des unterliegenden ADOs. Diese Pfeile verweisen auf die Methoden, die von den Schnittstellenelementen, auf denen ein Ereignis stattgefunden hat, aufgerufen werden. Hier sind dies die Methoden `schreibeAnmerkungenPflege()`, die durch einen `Mausklick` auf den Button aufgerufen wird, `zeigeAnmerkungenPflege()`, die durch das Ereignis `Anzeige` aufgerufen wird, `ermöglicheEingabe(Textfeld)`, die durch das Ereignis `Mausklick` auf das Textfeld aufgerufen wird, und die Methode `setzeInhalt(TextFeld)`, die durch das Ereignis `Tastatureingabe` auf dem Textfeld aufgerufen wird. Wenn eine aufgerufene Methode weitere Methoden aufruft oder ein oder mehrere Schnittstellenelemente verändert, führen ein oder mehrere gepunktete Pfeile zu diesen Schnittstellenelementen. In dem gezeigten Beispiel wirken sich die aufgerufenen Methoden `setzeInhalt(TextFeld)` und `ermöglicheEingabe(Textfeld)` auf das Textfeld aus, und die Methode `schreibeAnmerkungenPflege()` auf den ADV `AnmerkungenPflege`.

Methoden

In diesem Abschnitt werden die durch die ADVs aufgerufenen Methoden der ADOs vorgestellt und erläutert. Die im Konzeptentwurf definierten Methoden werden hier um schnittstellenspezifische

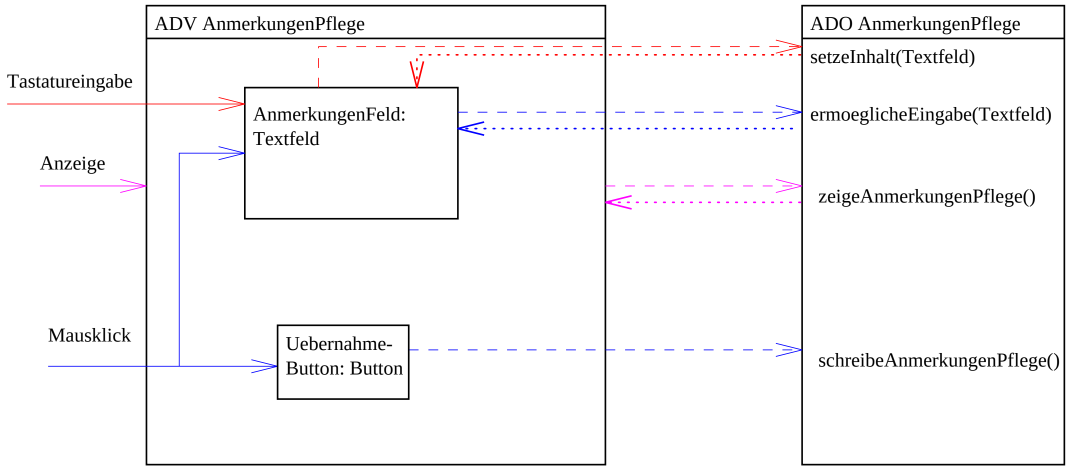


Abbildung 4.14: Beispiel eines Konfigurationsdiagramms

Methoden ergänzt. Zu jeder Methode wird angemerkt, wodurch sie aufgerufen wird.

Präsentationsattribute

In diesem Abschnitt werden die Präsentationsattribute der ADV definiert. Diese umfassen Schriftgrad und Farbe sowie eine relative Positionierung der Elemente der ADV.

ADVChart

Eine ADVChart beschreibt die Zustände einer ADV und die möglichen Transitionen zwischen diesen Zuständen. Hier wird angegeben, welche Elemente wann sichtbar sind und ob sie exklusiv darstellbar sind. Sichtbarkeit wird beschrieben, indem für Elemente festgelegt wird, wann sie sich im Zustand 'Anzeige aus' oder 'Anzeige an' befinden. Elemente, die nebeneinander existieren können, werden durch gestrichelte Linien getrennt, um anzuzeigen, daß sie nicht denselben Raum einnehmen. Elemente, die sich zusammen in einem gestrichelten Kästchen befinden, nehmen denselben virtuellen Raum ein und können daher nur exklusiv angezeigt werden. Transitionen der ADV werden durch Pfeile mit einer Nummer auf dem entsprechenden Element gekennzeichnet. Ein Beispiel für eine ADVChart wird in Abbildung 4.15 dargestellt. Die Elemente `ADV Basisdaten`, `ADV Legendenfenster`, `ADV Hilfefenster`, `ADV Signalleiste`, `Button LegendenButton`, `Button HilfeButton`, `WechselButton` und `ADV Navigationsleiste Patient` werden zusammen angezeigt, wobei die Elemente `Hilfefenster` und `Legendenfenster` im Zustand 'an' oder 'aus' sein können. Die Elemente `ADV Stammdaten` und `ADV Vitalwerte` werden exklusiv und zusätzlich zu den anderen Elementen angezeigt.

Transitionen

Die Transitionen der ADV sind nach folgendem Schema aufgebaut:

Nummer: Die Nummer der Transition. Diese entspricht der Nummer im ADVChart.

Ereignis: Hier wird das Ereignis, welches die Transition auslöst, oder die ausgelöste Methode festgelegt.

Vorbedingungen: Hier werden optionale Vorbedingungen, die erfüllt sein müssen, damit die Transition ausgelöst wird, festgelegt.

Nachbedingungen: Hier werden die Nachbedingungen einer Transition definiert. OOHDM gestattet es, wie in [RSLC95] in Abschnitt 4.2 beschrieben, Aktionen als Nachbedingungen zu

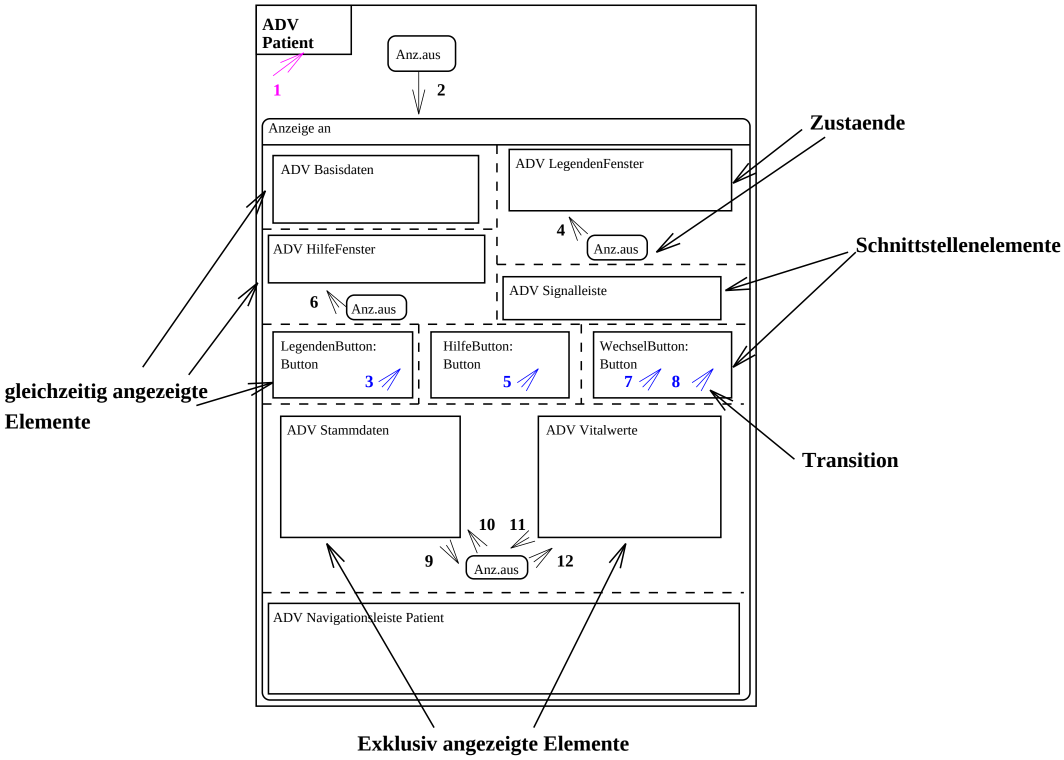


Abbildung 4.15: Beispiel eines ADVCharts

spezifizieren. Dies können aufgerufene Methoden oder Veränderungen des Wahrnehmungskontexts WK sein. Aufgerufene Methoden, die selbst Transitionen bilden, werden zwecks Übersichtlichkeit mit ihrer Transitionsnummer in eckigen Klammern aufgeführt.

In Abbildung 4.16 werden zwei Beispieltransitionen des in Abbildung 4.15 vorgestellten ADVCharts präsentiert. Die Transition **1** beschreibt das Ereignis `Anzeige`. Es müssen keine zusätzlichen Vorbedingungen erfüllt sein, damit die Nachbedingungen, der Aufruf der Methoden `leseDaten()` und `zeigePatient()` erfüllt werden. Die zweite Transition beschreibt das Ereignis `zeigePatient()`. Dies ist die von Transition **1** aufgerufene Methode. Dieser Aufruf hat zur Folge, daß die ADV `Patient` dem Wahrnehmungskontext WK hinzugefügt, d.h. sichtbar wird.

4.4.1 ADV Patient

Die ADV `Patient` stellt die abstrakte Schnittstelle der Navigationsklasse `Patient`, die eine Aggregation aus den Klassen `Fieberkurve, Signalleiste` und `Hilfe` ist, dar. Die Bestandteil-ADVs, die die ADV `Patient` bilden, sind die ADVs `Stammdaten, Vitalwerte, Signalleiste, Navigationsleiste Patient, LegendenFenster` und `HilfeFenster`. Diese ADVs werden in ihren eigenen Abschnitten beschrieben (siehe [Pro97]). Abbildung 4.17 stellt das Konfigurationsdiagramm der ADV `Patient` dar. Die Buttons der ADV reagieren auf das Ereignis `Mausklick`, wobei die Buttons `LegendenButton` und `HilfeButton` eine Anzeige des jeweils zugehörigen Fensters bewirken, und der Button `WechselButton` die Anzeige der ADVs `Stammdaten` und `Vitalwerte` wechselt. Die ADV `Signalleiste` reagiert auf das Ereignis `Nachricht`, indem sie sich selbst aktualisiert, und die ADV `Patient` reagiert auf das Ereignis `Anzeige`, indem sie die Methoden `leseDaten()` und `zeigePatient()` aufruft.

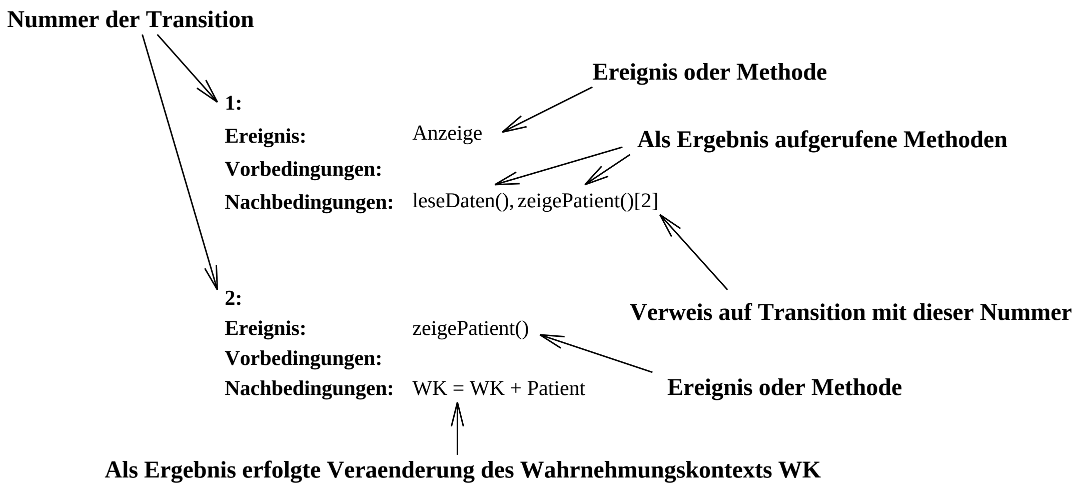


Abbildung 4.16: Beispiele für Transitionen eines ADVCharts

Bestandteile:

- ADV `Basisdaten`, bestehend aus den Textfeldern `StationsNameFeld, NameFeld, VornameFeld, ZimmerFeld, GeburtsdatumFeld` und `KrankenkassenFeld`, die die Attribute `Station, Name, Vorname, Zimmer, Geburtsdatum` und `Krankenkasse` darstellen
- ADV `LegendenFenster`
- Button `LegendenButton`
- Button `WechselButton`
- Button `HilfeButton`
- ADV `Stammdaten`
- ADV `Vitalwerte`
- ADV `Navigationsleiste Patient` (siehe 4.4.2)
- ADV `Signalleiste`
- ADV `HilfeFenster`

Methoden:

zeigePatient(): Diese Methode bewirkt die Anzeige der ADV `Patient`. Hier wird das anfängliche Layout der ADV festgelegt. Sie wird durch das Ereignis `Anzeige` aufgerufen.

leseDaten(): Diese Methode bewirkt das Einlesen aller Daten der ADV `Patient` in den Arbeitsspeicher. Sie wird durch das Ereignis `Anzeige` aufgerufen.

zeigeLegendenFenster(): Diese Methode bewirkt die Anzeige der ADV `LegendenFenster`. Sie wird durch das Ereignis `Mausklick` auf den Button `LegendenButton` aufgerufen.

zeigeHilfeFenster(): Diese Methode bewirkt die Anzeige der ADV `HilfeFenster`. Sie wird durch das Ereignis `Mausklick` auf den Button `HilfeButton` aufgerufen.

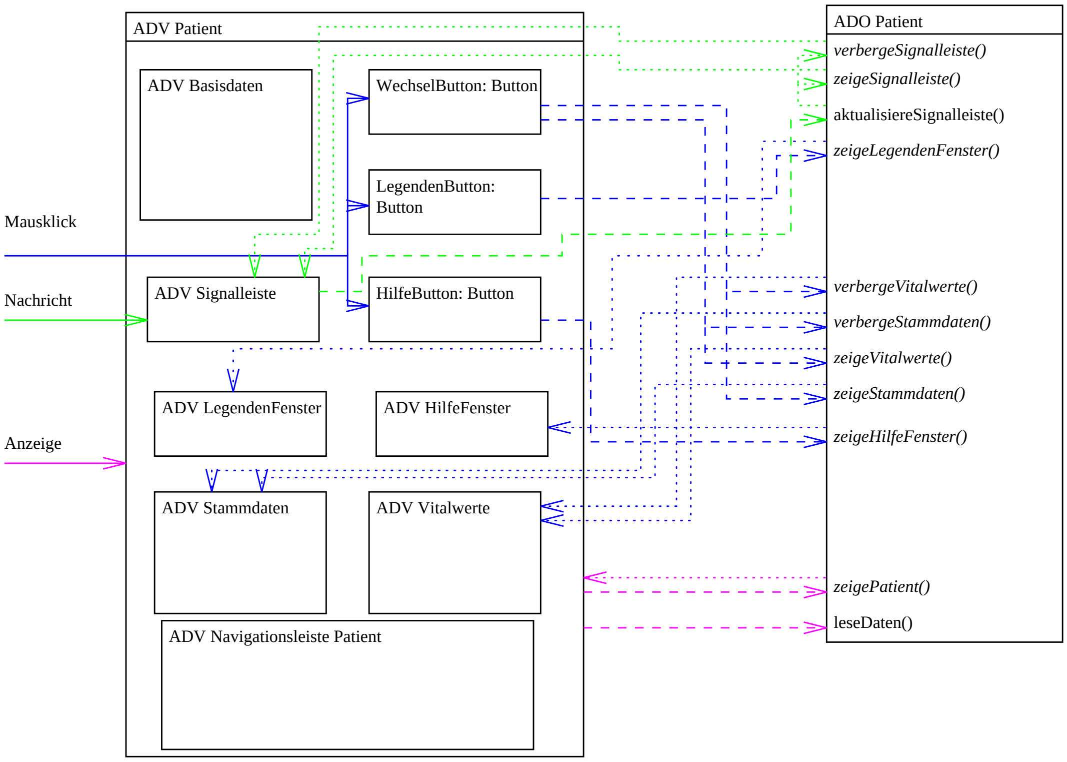


Abbildung 4.17: Konfigurationsdiagramm der ADV Patient

zeigeStammdaten(): Diese Methode bewirkt die Anzeige der ADV `Stammdaten`. Sie wird durch das Ereignis `Mausklick` auf den Button `WechselButton` aufgerufen, wenn die ADV `Stammdaten` nicht im WK ist.

zeigeVitalwerte(): Diese Methode bewirkt die Anzeige der ADV `Vitalwerte`. Sie wird durch das Ereignis `Mausklick` auf den Button `WechselButton` aufgerufen, wenn die ADV `Vitalwerte` nicht im WK ist.

verbergeStammdaten(): Diese Methode bewirkt das Ausblenden der ADV `Stammdaten`. Sie wird durch das Ereignis `Mausklick` auf den Button `WechselButton` aufgerufen, wenn die ADV `Vitalwerte` nicht im WK ist.

verbergeVitalwerte(): Diese Methode bewirkt das Ausblenden der ADV `Vitalwerte`. Sie wird durch das Ereignis `Mausklick` auf den Button `WechselButton` aufgerufen, wenn die ADV `Stammdaten` nicht im WK ist.

ermöglicheEingabe(Textfeld): Diese Methode bewirkt, daß die Eingabe von Text über die Tastatur in dem entsprechenden Textfeld möglich ist. Sie wird durch das Ereignis `Mausklick` auf das angegebene Textfeld aufgerufen.

verbieteEingabe(Textfeld): Diese Methode bewirkt, daß die Eingabe von Text über die Tastatur in dem entsprechenden Textfeld nicht möglich ist. Sie wird durch das Ereignis `Mausklick` auf ein anderes als das angegebene Textfeld aufgerufen.

Präsentationsattribute:

Schrift: Helvetica, 14 Punkte, für Überschriften 16 Punkte und fett

Farbe: Hintergrund Hellgrau; Textfelder weiß wenn Eingabe möglich, sonst grau

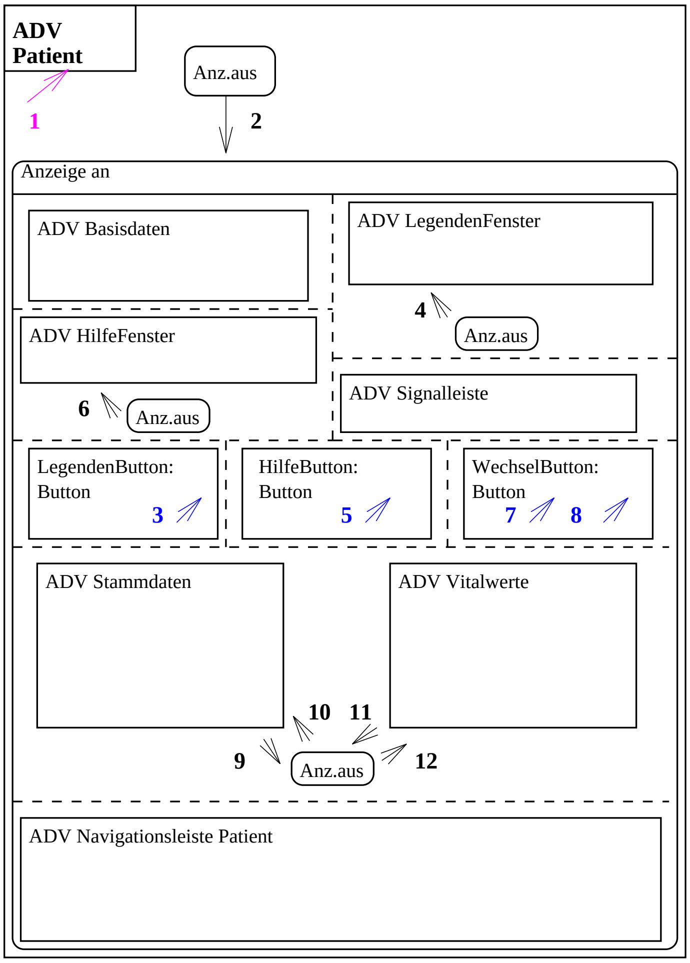


Abbildung 4.18: ADVChart Patient

Positionierung der Elemente von Patient: Obere Reihe von links nach rechts: ADV `Basisdaten`, ADV `Signalleiste`, übereinander die Buttons `WechselButton`, `LegendenButton`, `HilfeButton`; Mittlere Reihe: ADV `Stammdaten` oder `ADV Vitalwerte`; Untere Reihe: `ADV Navigationsleiste Patient`

Transitionen:

Transitionen der ADV Patient:

1. **Ereignis** : Anzeige
   Vorbedingungen :
   Nachbedingungen : leseDaten(); zeigePatient()[2]
2. **Ereignis** : zeigePatient()
   Vorbedingungen :
   Nachbedingungen : WK = WK + `Patient`
3. **Ereignis** : Mausklick
   Vorbedingungen : Fokus(`LegendenButton`)
   Nachbedingungen : zeigeLegendenFenster()[4]
4. **Ereignis** : zeigeLegendenFenster()
   Vorbedingungen :
   Nachbedingungen : WK = WK + `Legendenfenster`
5. **Ereignis** : Mausklick

Vorbedingungen : Fokus(`HilfeButton`)

Nachbedingungen : zeigeHilfeFenster()[6]

6. **Ereignis** : zeigeHilfeFenster()

   Vorbedingungen :

   Nachbedingungen : WK = WK + `Hilfefenster`

7. **Ereignis** : Mausklick

   Vorbedingungen : Fokus(`WechselButton`); `Stammdaten` in WK

   Nachbedingungen : verbergeStammdaten()[9]; zeigeVitalwerte()[12]

8. **Ereignis** : Mausklick

   Vorbedingungen : Fokus(`WechselButton`); `Vitalwerte` in WK

   Nachbedingungen : verbergeVitalwerte()[11]; zeigeStammdaten()[10]

9. **Ereignis** : verbergeStammdaten()

   Vorbedingungen :

   Nachbedingungen : WK = WK - `Stammdaten`

10. **Ereignis** : zeigeStammdaten()

    Vorbedingungen :

    Nachbedingungen : WK = WK + `Stammdaten`

11. **Ereignis** : verbergeVitalwerte()

    Vorbedingungen :

    Nachbedingungen : WK = WK - `Vitalwerte`

12. **Ereignis** : zeigeVitalwerte()

    Vorbedingungen :

    Nachbedingungen : WK = WK + `Vitalwerte`

4.4.2 Navigationsleiste Patient

Diese ADV faßt die Anker aller Navigationsklassen des `Patientenkontexts` zusammen, der Teil der `Schwesternsicht` des Navigationsentwurfs ist und realisiert die Zugangsstrukturen dieses Kontexts. Sie ist Teil der ADV `Patient` (in Abschnitt 4.4.1 beschrieben), `Diagnostik`, `Therapie`, `Pflege`, `Terminplan` und `Anforderungsergebnis` (siehe [Pro97]). Abbildung 4.19 stellt das Konfigurationsdiagramm dieser ADV dar.

Bestandteile:

- ADV `Patientenliste`, bestehend aus einer Liste mit 1-n Textfeldern `PatientNameFeld`, die das Attribut `Name` aller Elemente von `Patientendaten` beinhalten, dem Button `OKButton` und dem Button `AbbruchButton`
- ADV `Patientenanforderungsliste`, bestehend aus einer Liste mit 1-n Textfeldern `PatientenanforderungsNameFeld`, die das Attribut `Name` aller Elemente von `Patientenanforderungsdaten` beinhalten, dem Button `OKButton` und dem Button `AbbruchButton`
- ADV `Patientenanforderungsergebnisliste`, bestehend aus einer Liste mit 1-n Textfeldern `PatientenanforderungsergebnisNameFeld`, die das Attribut `Name` aller Elemente von `Patientenanforderungsergebnisdaten` beinhalten, dem Button `OKButton` und dem Button `AbbruchButton`
- Button `PatientenlistenButton`
- Button `PatientenanforderungslistenButton`
- Button `PatientenanforderungsergebnislistenButton`
- Button `DiagnostikButton`
- Button `PflegeButton`
- Button `TherapieButton`

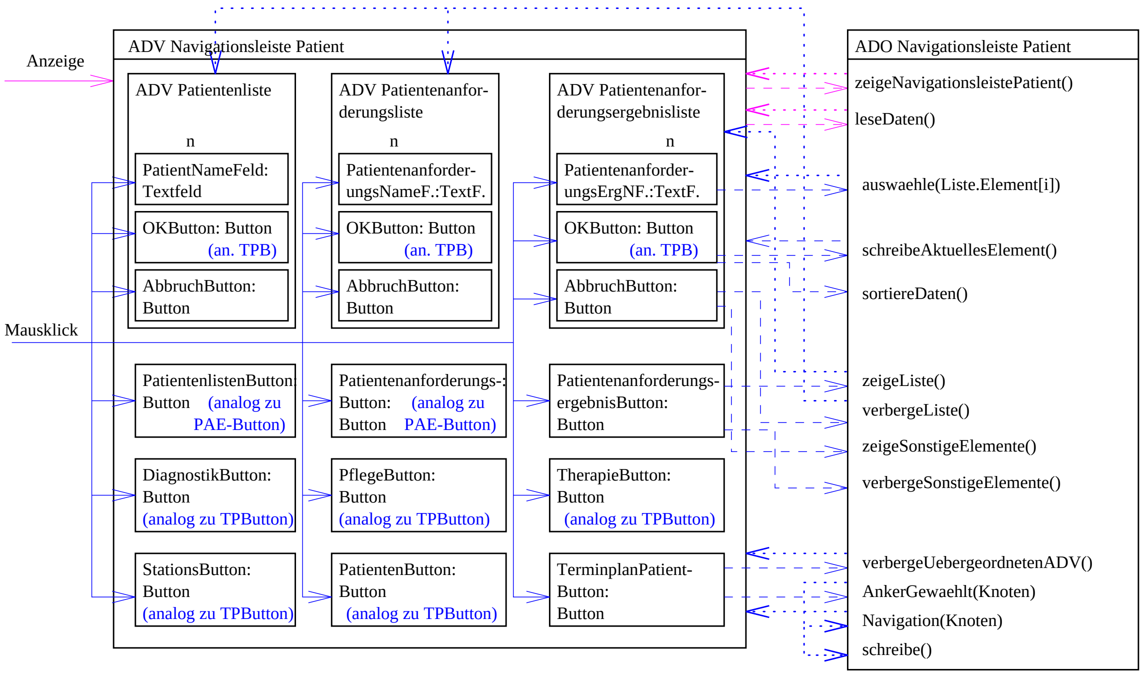


Abbildung 4.19: Konfigurationsdiagramm der ADV Navigationsleiste Patient

- Button `StationsButton`
- Button `PatientenButton`
- Button `TerminplanPatientButton`

Methoden:

zeigeNavigationsleistePatient(): Diese Methode verursacht die Anzeige der ADV `Navigationsleiste Patient`. Dies geschieht im Rahmen der `zeige()`-Methode der übergeordneten ADV. Sie wird durch das Ereignis `Anzeige` aufgerufen.

leseDaten(): Diese Methode liest die Daten der Objektlisten `Patientendaten`, `Patientenanforderungsdaten` und `Patientenanforderungsergebnisdaten` in den Arbeitsspeicher ein. Sie wird durch das Ereignis `Anzeige` aufgerufen.

auswähle(Liste.Element(i)): Diese Methode wählt das durch einen Mausklick markierte Element i einer Liste aus. Bei der ADV `Patientenliste` ist es ein `Patient`, bei der ADV `Patientenanforderungsliste` eine `Anforderung Patient` und bei der ADV `Patientenanforderungsergebnisliste` ein `Anforderungsergebnis Patient`. Sie wird durch das Ereignis `Mausklick` auf das Element i der Liste ausgelöst.

schreibeAktuellesElement(): Diese Methode faßt die Methoden `schreibeAktuellenPatienten()`, `schreibeAktuellePatientenanforderung()` und `schreibeAktuellesPatientenanforderungsergebnis()` zusammen. Sie bewirkt, daß die jeweils ausgewählten Elemente gespeichert werden. Sie wird durch das Ereignis `Mausklick` auf den Button `OKButton` einer Listen-ADV ausgelöst.

zeigeListe(): Diese Methode verursacht die Anzeige einer ADV, die hier die `Patientenliste`, die `Patientenanforderungsliste` oder die `Patientenanforderungsergebnisliste` sein kann. Sie wird durch das Ereignis `Mausklick` auf einen der Buttons `PatientenlistenButton`, `PatientenanforderungsButton` oder `PatientenanforderungsergebnisButton` aufgerufen.

verbergeListe(): Diese Methode verursacht das Ausblenden einer ADV, die hier die `Patientenliste`, die `Patientenanforderungsliste` oder die `Patientenanforderungsergebnisliste` sein kann. Sie wird durch das Ereignis `Mausklick` auf einen der Buttons `OKButton` der Listen-ADVs aufgerufen.

zeigeSonstigeElemente(): Diese Methode verursacht die Anzeige von Elementen der übergeordneten ADV und wird dort realisiert. Sie wird durch das Ereignis Mausklick auf den Button `AbbruchButton` einer der Listen-ADVs aufgerufen.

verbergeSonstigeElemente(): Diese Methode verursacht das Ausblenden von Elementen der übergeordneten ADV und wird dort realisiert. Sie wird durch das Ereignis `Mausklick` auf einen der Buttons `PatientenlistenButton, PatientenanforderungsButton` oder `PatientenanforderungsergebnisButton` aufgerufen.

verbergeÜbergeordnetenADV(): Diese Methode bewirkt das Ausblenden des der ADV `Navigationsleiste Patient` übergeordneten ADV. Diese Methode wird durch das Ereignis `Mausklick` auf einen der Buttons `OKButton` der Listen-ADVs oder einen der Buttons `StationsButton, PatientenButton, DiagnostikButton, PflegeButton, TherapieButton` oder `TerminplanPatientButton` aufgerufen.

auswähleAnker(Knoten): Diese Methode ruft die Methoden `navigiere(Knoten)` und `schreibe()` auf. Diese Methode wird durch das Ereignis `Mausklick` auf einen der Buttons `OKButton` der Listen-ADVs oder einen der Buttons `StationsButton, PatientenButton, DiagnostikButton, PflegeButton, TherapieButton` oder `TerminplanPatientButton` aufgerufen.

navigiere(Knoten): Diese Methode verursacht Navigation zu der den Knoten realisierenden ADV. Sie wird durch die Methode `auswähleAnker(Knoten)` aufgerufen.

schreibe(): Diese Methode bewirkt das Abspeichern der Daten des übergeordneten ADVs. Sie wird durch die Methode `auswähleAnker(Knoten)` aufgerufen.

sortiereDaten(): Diese Methode bewirkt das Sortieren der Daten der ADOs der Listen-ADVs. Sie wird durch das Ereignis `Mausklick` auf einen der Buttons `OKButton` der Listen-ADVs aufgerufen.

Präsentationsattribute:

Schrift: Helvetica, 14 Punkte, für Überschriften 16 Punkte und fett

Farbe: Hintergrund Hellgrau; Textfelder weiß wenn Eingabe möglich, sonst grau

Positionierung der Elemente von Navigationsleiste Patient: Drei Reihen mit je drei Buttons; obere Reihe von links nach rechts `StationsButton, PatientenlistenButton, PflegeButton`; mittlere Reihe von links nach rechts `TherapieButton, DiagnostikButton, TerminplanButton`; untere Reihe von links nach rechts `PatientenButton, PatientenanforderungsButton, PatientenanforderungsergebnisButton`.

Anmerkung: Die Listen-ADVs werden exklusiv zueinander und zu den ADVs der übergeordneten ADV, die dort in der mitteren Reihe dargestellt sind, angezeigt.

Transitionen:

1. **Ereignis** : Anzeige
   Vorbedingungen :
   Nachbedingungen : leseDaten(); zeigeNavigationsleistePatient()[2]
2. **Ereignis** : zeigeNavigationsleistePatient()
   Vorbedingungen :

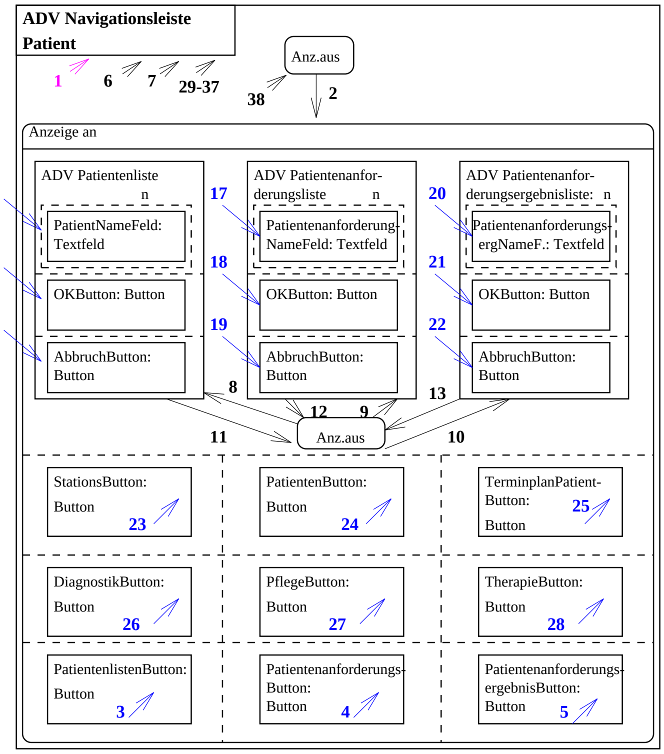


Abbildung 4.20: ADVChart Navigationsleiste Patient

Nachbedingungen : WK = WK + `Navigationsleiste Patient`

3. **Ereignis** : Mausklick

   Vorbedingungen : Fokus(`PatientenlistenButton`)

   Nachbedingungen : verbergeSonstigeElemente()[7]; zeigePatientenliste()[8]

4. **Ereignis** : Mausklick

   Vorbedingungen : Fokus(`PatientenanforderungslistenButton`)

   Nachbedingungen : verbergeSonstigeElemente()[7]; zeigePatientenanforderungsliste()[9]

5. **Ereignis** : Mausklick

   Vorbedingungen : Fokus(`PatientenanforderungsergebnislistenButton`)

   Nachbedingungen : verbergeSonstigeElemente()[7]; zeigePatientenanforderungsergebnisliste()[10]

6. **Ereignis** : zeigeSonstigeElemente()

   Vorbedingungen :

   Nachbedingungen : WK = WK + `sonstigeElemente`

7. **Ereignis** : verbergeSonstigeElemente()

   Vorbedingungen :

   Nachbedingungen : wenn `Patientenliste` in WK dann verbergePatientenliste()[11]; wenn `Patientenanforderungsliste` in WK dann verbergePatientenanforderungsliste()[12]; wenn `Patientenanforderungsergebnisliste` in WK dann verbergePatientenanforderungsergebnisliste()[13]; WK = WK - sonstigeElemente

8. **Ereignis** : zeigePatientenliste()

   Vorbedingungen :

   Nachbedingungen : WK = WK + `Patientenliste`

9. **Ereignis** : zeigePatientenanforderungsliste()

   Vorbedingungen :

Nachbedingungen : WK = WK + `Patientenanforderungsliste`

10. **Ereignis** : zeigePatientenanforderungsergebnisliste()

    Vorbedingungen :

    Nachbedingungen : WK = WK + `Patientenanforderungsergebnisliste`

11. **Ereignis** : verbergePatientenliste()

    Vorbedingungen :

    Nachbedingungen : WK = WK - `Patientenliste`

12. **Ereignis** : verbergePatientenanforderungsliste()

    Vorbedingungen :

    Nachbedingungen : WK = WK - `Patientenanforderungsliste`

13. **Ereignis** : verbergePatientenanforderungsergebnisliste()

    Vorbedingungen :

    Nachbedingungen : WK = WK - `Patientenanforderungsergebnisliste`

14. **Ereignis** : Mausklick

    Vorbedingungen : Fokus(`PatientNameFeld[i]`)

    Nachbedingungen : auswähle(`PatientNameFeld[i]`)

15. **Ereignis** : Mausklick

    Vorbedingungen : Fokus(`Patientenliste.OKButton`)

    Nachbedingungen : schreibeAktuellerPatient(); auswähleAnker(`Patient`)[35]

16. **Ereignis** : Mausklick

    Vorbedingungen : Fokus(`Patientenliste.AbbruchButton`)

    Nachbedingungen : verbergePatientenliste()[11]; zeigeSonstigeElemente()[6]

17. **Ereignis** : Mausklick

    Vorbedingungen : Fokus(`PatientenanforderungNameFeld[i]`)

    Nachbedingungen : auswähle(`PatientenanforderungNameFeld[i]`)

18. **Ereignis** : Mausklick

    Vorbedingungen : Fokus(`Patientenanforderungsliste.OKButton`)

    Nachbedingungen : schreibeAktuellePatientenanforderung(); auswähleAnker(`Anforderung Patient`)[36]

19. **Ereignis** : Mausklick

    Vorbedingungen : Fokus(`Patientenanforderungsliste.AbbruchButton`)

    Nachbedingungen : verbergePatientenanforderungsliste()[12]; zeigeSonstigeElemente()[6]

20. **Ereignis** : Mausklick

    Vorbedingungen : Fokus(`PatientenanforderungsergebnisNameFeld[i]`)

    Nachbedingungen : auswähle(`PatientenanforderungsergebnisNameFeld[i]`)

21. **Ereignis** : Mausklick

    Vorbedingungen : Fokus(`Patientenanforderungsergebnisliste.OKButton`)

    Nachbedingungen : schreibeAktuellesPatientenanforderungsergebnis(); auswähleAnker(`Anforderungsergebnis Patient`)[37]

22. **Ereignis** : Mausklick

    Vorbedingungen : Fokus(`Patientenanforderungsergebnisliste.AbbruchButton`)

    Nachbedingungen : verbergePatientenanforderungsergebnisliste()[13]; zeigeSonstigeElemente()[6]

23. **Ereignis** : Mausklick

    Vorbedingungen : Fokus(`StationsButton`)

    Nachbedingungen : auswähleAnker(`Station`)[29]

24. **Ereignis** : Mausklick

    Vorbedingungen : Fokus(`PatientenButton`)

Nachbedingungen : auswähleAnker(`Patient`)[30]

25. **Ereignis** : Mausklick

    Vorbedingungen : Fokus(`TerminplanPatientButton`)

    Nachbedingungen : auswähleAnker(`Terminplan Patient`)[31]

26. **Ereignis** : Mausklick

    Vorbedingungen : Fokus(`DiagnostikButton`)

    Nachbedingungen : auswähleAnker(`Diagnostik`)[32]

27. **Ereignis** : Mausklick

    Vorbedingungen : Fokus(`PflegeButton`)

    Nachbedingungen : auswähleAnker(`Pflege`)[33]

28. **Ereignis** : Mausklick

    Vorbedingungen : Fokus(`TherapieButton`)

    Nachbedingungen : auswähleAnker(`Therapie`)[34]

29. **Ereignis** : auswähleAnker(`Station`)

    Vorbedingungen :

    Nachbedingungen : verbergeÜbergeordnetenADV()[38]; navigiere(`Station`)

30. **Ereignis** : auswähleAnker(`Patient`)

    Vorbedingungen :

    Nachbedingungen : verbergeÜbergeordnetenADV()[38]; navigiere(`Patient`)

31. **Ereignis** : auswähleAnker(`Terminplan Patient`)

    Vorbedingungen :

    Nachbedingungen : verbergeÜbergeordnetenADV()[38]; navigiere(`Terminplan Patient`)

32. **Ereignis** : auswähleAnker(`Diagnostik`)

    Vorbedingungen :

    Nachbedingungen : verbergeÜbergeordnetenADV()[38]; navigiere(`Diagnostik`)

33. **Ereignis** : auswähleAnker(`Pflege`)

    Vorbedingungen :

    Nachbedingungen : verbergeÜbergeordnetenADV()[38]; navigiere(`Pflege`)

34. **Ereignis** : auswähleAnker(`Therapie`)

    Vorbedingungen :

    Nachbedingungen : verbergeÜbergeordnetenADV()[38]; navigiere(`Therapie`)

35. **Ereignis** : auswähleAnker(`Patient`)

    Vorbedingungen :

    Nachbedingungen : verbergeÜbergeordnetenADV()[38]; navigiere(`Patient`)

36. **Ereignis** : auswähleAnker(`Anforderung Patient`)

    Vorbedingungen :

    Nachbedingungen : verbergeÜbergeordnetenADV()[38]; navigiere(`Anforderung Patient`)

37. **Ereignis** : auswähleAnker(`Anforderungsergebnis Patient`)

    Vorbedingungen :

    Nachbedingungen : verbergeÜbergeordnetenADV()[38]; navigiere(`Anforderungsergebnis Patient`); schreibe()

38. **Ereignis** : verbergeÜbergeordnetenADV()

    Vorbedingungen :

    Nachbedingungen : WK = WK - übergeordneter ADV

4.5 Implementation

In diesem Kapitel wird die Umsetzung der in den vorherigen Entwicklungsschritten spezifizierten Modelle in einen Prototypen beschrieben.

Es existieren heute viele verschiedene Plattformen, auf denen Hypermediasysteme implementiert werden können. Die bekanntesten Multimediaentwicklungsumgebungen wie ToolBook [Bud96] oder Director [Mac97] unterstützen viele speziell auf Multimedia zugeschnittene Funktionen, haben aber den Nachteil eingeschränkter Flexibilität und Mächtigkeit von Entwicklungsumgebungen gegenüber Programmiersprachen.

Eine andere Möglichkeit ist die Implementierung auf HTML-Basis [JN95]. Ein Problem bei der Benutzung von HTML-Dokumenten ist allerdings die fehlende Dynamik. HTML selbst liefert keine direkten Möglichkeiten, Web-Seiten dynamisch zu gestalten. Dafür gibt es verschiedene Methoden:

- Kommunikation mit externen Datenquellen über CGI-Schnittstellen [Gun96]
- dynamische Generierung von HTML-Seiten durch CGI-Skripte und Skriptsprachen wie JavaScript [Fla97]
- Einbettung von Funktionalität in HTML-Seiten durch Java-Applets [MSS96]

4.5.1 Allgemeines zum Prototypen

Die Implementierung des Prototypen der hypermedialen Patientenakte erfolgt auf HTML/Java-Basis im Rahmen eines WWW-Intranets [RM97].

Für den Prototypen wird die gesamte Funktionalität eines Knotens, welcher durch eine ADV und ihre unterliegende Navigationsklasse gebildet wird, in einem Applet realisiert, welches in eine HTML-Seite eingebettet ist. Jede ADV wird durch eine HTML/Java-Applet-Seite implementiert, welche durch einen WWW-Browser dargestellt wird. Die Umsetzung des abstrakten Schnittstellenmodells ist durch die Java-AWT-Klassen relativ einfach und direkt möglich, da das *Abstract Windowing Toolkit* von Java viele vordefinierte Benutzerschnittstellenelemente wie Buttons, Textfelder, Listen und andere zur Verfügung stellt. Ferner stellt Java verschiedene Layout-Manager bereit, die die Positionierung von Bildschirmelementen sehr einfach machen.

4.5.2 Realisierung durch ein Intranet

Ein Intranet ist die Realisierung eines Netzwerks auf Basis der Internet-Technologie, mit dem Unterschied, daß dieses Netzwerk durch sogenannte Firewalls vom Internet abgeschottet ist. Der Hauptvorteil diser Art der Implementierung liegt in der Hardwareunabhängigkeit und Verteiltheit eines mit dieser Technologie arbeitenden Netzwerks. Es ist möglich, verschiedene Plattformen nahtlos miteinander zu vernetzen. So kann gewährleistet werden, daß die aufgabenangemessene Hardware auf jeder Station und jeder Leistungsstelle reibungslos miteinander kommuniziert.

4.5.3 Implementierung in Java

Die Realisierung der Modelle durch Java-Applets hat den Vorteil, daß der gesamte Informationsgehalt einer Seite durch eine einzige Technik dargestellt wird. Es ist hier nicht nötig, verschiedene Techniken wie Scripting, CGI-Programmierung und dynamische Seitengenerierung zu vereinen, sondern man hat die Funktionalität und Mächtigkeit einer objektorientierten Programmiersprache, die speziell für verteilte Anwendungen in einer Netzwerkumgebung entwickelt wurde, zur Verfügung. Die Verwendung der Programmiersprache Java unterstützt die Verwirklichung einiger Anforderungen durch ihre inhärenten Eigenschaften *Portabilität, Objektorientierung, Multithreading, Verteiltheit, Robustheit* und *Sicherheit* (siehe [MSS96]).

Für den Prototypen wurden die folgenden ADVs mit den darunterliegenden ADOs der Schwesternsicht implementiert:

- `Station` als `StationsApplet.java`
- `Terminplan Station` als `TerminplanStationApplet.java`

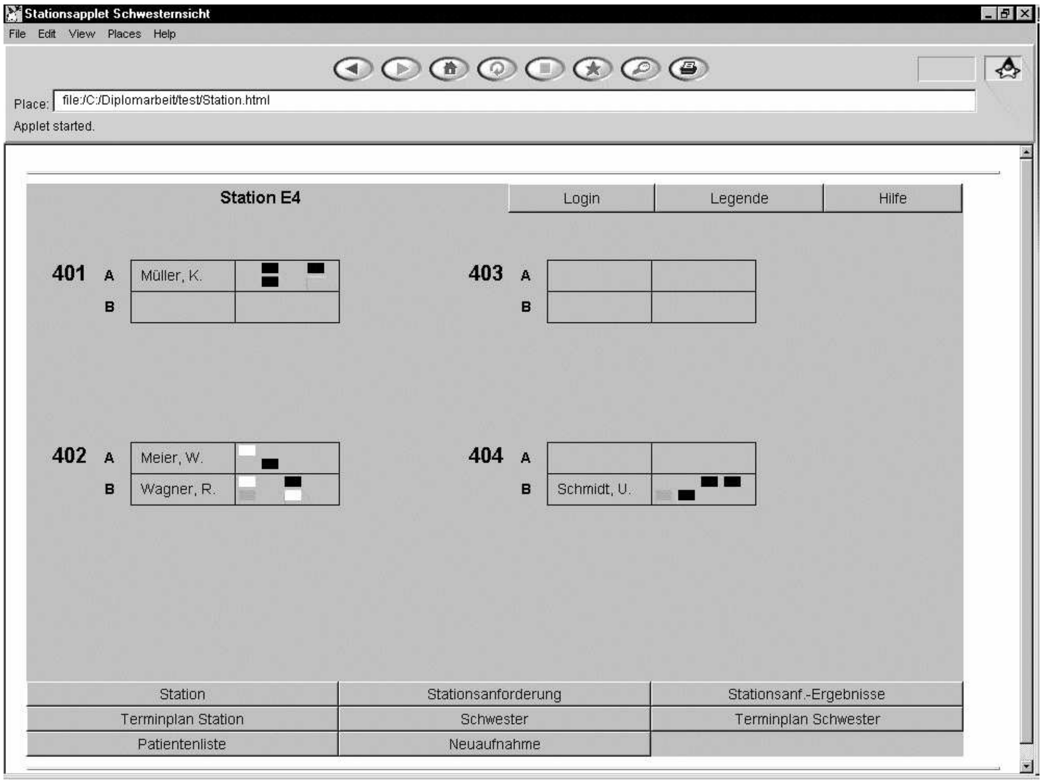


Abbildung 4.21: Screenshot des Stationsbildschirms

- `Materialanforderung` als Teil des Applets `StationsanforderungsApplet.java`
- `Materialanforderungsergebnis` als Teil des Applets `Stationsanforderungsergebnis-Applet.java`
- `Patient` als `PatientenApplet.java`
- `Neuaufnahme` als `NeuaufnahmeApplet.java`
- `Fieberkurve` als Teil des Applets `PatientenApplet.java`
- `Terminplan Patient` als `TerminplanPatientApplet.java`
- `Diagnostik` als `DiagnostikApplet.java`
- `Pflege` als `PflegeApplet.java`
- `Therapie` als `TherapieApplet.java`
- `Röntgenanforderung` als Teil des Applets `PatientenanforderungsApplet.java`
- `Röntgenanforderungsergebnis` als Teil des Applets `Patientenanforderungsergebnis-Applet.java`

Die obengenannten Applets setzen sich aus verschiedenen GUI-Elementen von Java, die die verschiedenen, hier nicht genannten weiteren ADVs wie `HilfeFenster`, `VitalwertTabelle` und andere realisieren. Eine vollständige Auflistung aller Java-Klassen findet sich im Anhang [Pro97].

Die Abbildungen 4.21 und 4.22 stellen Screenshots der Applets *StationsApplet* und *PatientenApplet*, die die ADVs `Station` und `Patient` implementieren, dar.

Jedes Applet der hypermedialen Patientenakte besteht aus zwei Hauptkomponenten:

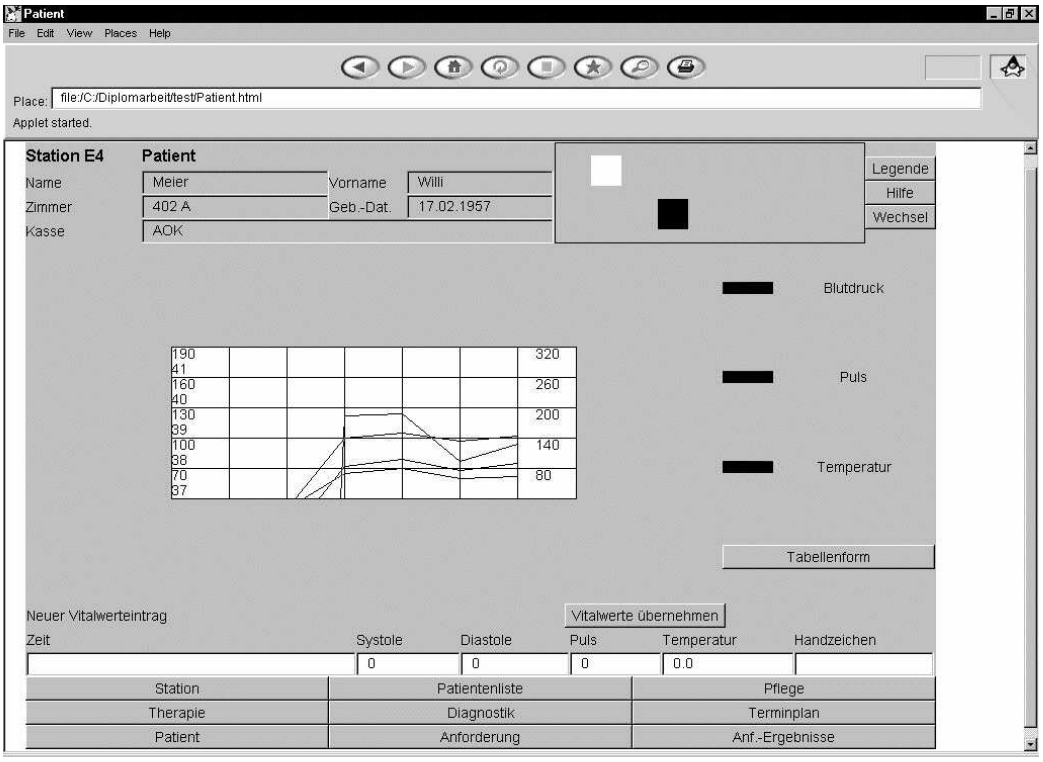


Abbildung 4.22: Screenshot des Patientenbildschirms

- Einer Initialisierung, die im Rahmen der *init()*-Methode des Applets vorgenommen wird: Es wird als erstes die bei allen Knoten vorhandene Methode *leseDaten()* aufgerufen, die den entsprechenden Datensatz vom unterliegenden Speichermedium lädt. Anschließend wird das Bildschirmlayout festgelegt. Dies geschieht durch Benutzung der Klassen des *Abstract Windowing Toolkit*. Im Rahmen der Festlegung des Layouts wird das Applet mit den aktuell geladenen Daten initialisiert.

- Dem Event-Handling: Der größte Teil der Funktionalität der hypermedialen Patientenakte wird durch benutzerinitiierte Ereignisse ausgelöst. Diese Ereignisse werden in Java im Rahmen der vordefinierten Methode *handleEvent()* behandelt, die sich wiederum verschiedener Hilfsmethoden wie der Methode *action()* bedient. Die meisten im Konzeptentwurf definierten funktionalen und im abstrakten Schnittstellenentwurf definierten oberflächenverändernden Methoden werden im Rahmen dieser Methode *action()* oder einer anderen Hilfsmethode aufgerufen. Auch hier ermöglicht es diese Methode, eine relativ einfache und direkte Umsetzung des abstrakten Schnittstellenmodells vorzunehmen.

Die Speicherung der Patientendaten erfolgt bei dem Prototypen in einer Access-Datenbank. Das Java-Applet fungiert als Front-End für diese Datenbank, die dem Benutzer grafisch aufbereitet im Rahmen eines Web-Browsers präsentiert wird. Dies wird durch die Benutzung der JDBC-Klassen (JDBC-Java DataBase Connectivity) gewährleistet. Diese Klassen ermöglichen es, Java-Anwendungen und Applets mit SQL-fähigen Datenbanken über eine Schnittstelle kommunizieren zu lassen [PM96].

Abbildung 4.23 stellt die Architektur von JDBC sowie die Zusammenarbeit von JDBC und ODBC dar. Der Treiber-Manager wird benutzt, um eine Verbindung zu einer Datenbank über einen JDBC-Treiber herzustellen. Dieser Treiber muß vom Manager zunächst registriert werden. Im Falle einer gewünschten Verbindung wählt der Manager einen passenden Treiber aus. Der gewählte Treiber nimmt dann konkret SQL-Anfragen seitens der Java-Anwendung entgegen und liefert Anfrage-

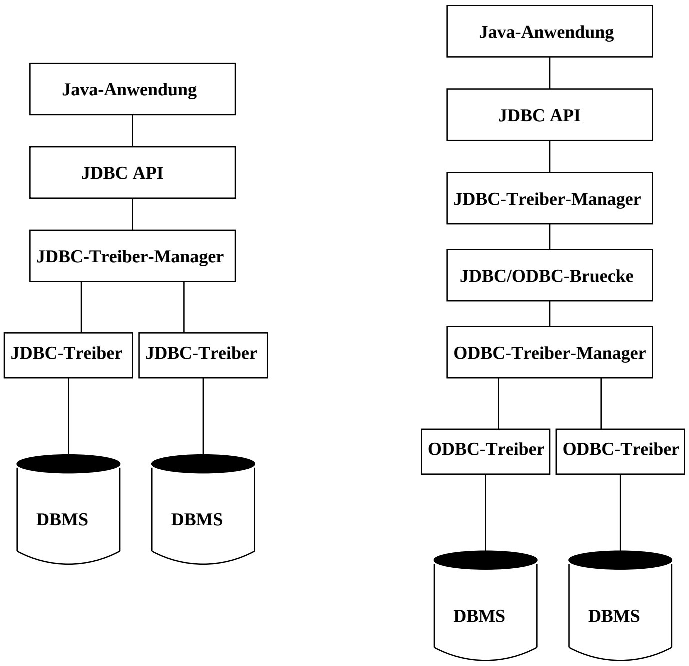


Abbildung 4.23: JDBC: Architektur und Zusammenarbeit mit ODBC

ergebnisse an die Java-Anwendung. Verwendet wird eine URL-Syntax für die Adressierung der Zieldatenbank.

Die Java-Version 1.1 beinhaltet eine JDBC-ODBC-Brücke, die es ermöglicht, aus Java-Anwendungen heraus mit ODBC-fähigen Anwendungen und Datenbanken zu kommunizieren. Diese Brücke wurde für die prototypische Implementierung der hypermedialen Patientenakte genutzt, um eine Access-Datenbank als Speichermedium verwenden zu können. In der Realisierung werden Datensätze geladen, indem SQL-Strings konstruiert und an die Datenbank über den Treiber weitergeleitet und die erhaltenen Ergebnisse direkt in Objekte und Attribute geschrieben werden.

4.5.4 Browser und externe Anwendungen

Da der Prototyp auf Applet-Basis funktioniert, wird als Werkzeug zur Benutzung der hypermedialen Patientenakte ein Web-Browser benötigt. Die gängigsten erhältlichen Browser, der Netscape Navigator und der Microsoft Internet Explorer, sind in den zur Zeit der Implementierung erhältlichen Versionen noch nicht in der Lage, die Java-Version 1.1, welche die JDBC-Klassen beinhaltet, zu unterstützen. Ferner ist mit diesen Browsern auch keine differenzierte Applet-Sicherheitsbehandlung möglich. Da der Prototyp mit Auslagerungsdateien auf lokalen Speichermedien arbeitet, die Browser aber prinzipiell Applets aus Sicherheitsgründen unter keinen Umständen lokalen Zugriffe ermöglichen, sind diese beiden Produkte als Werkzeug für den Prototypen eher schlecht geeignet.

Verwendet wird der HotJava-Browser von Sun. Dieser hat den Vorteil, daß es dem Benutzer möglich ist, verschiedene Sicherheitsstufen für Applets einzustellen. Diese Eigenschaft macht den HotJava-Browser für Intranet-Anwendungen interessanter als die beiden vorhergehenden. Er hat allerdings den Nachteil, eine schlechtere Performanz als der Navigator und der Explorer zu liefern. Dies liegt an der vollständigen Implementierung des Browsers in Java, welches im allgemeinen wesentlich langsamer ausgeführt wird als C- oder C++-Programme. Der Prototyp wurde auf PCs getestet und erreichte auf einem Pentium mit 32 MB RAM eine zufriedenstellende Leistung. Ein 486-PC zeigte deutlich merkbare Verzögerungen.

4.5.5 Integration von Multimediadaten und externen Werkzeugen

Die Verwendung eines Browsers macht sich dessen Eigenschaften als universelles Werkzeug zunutze. Browser sind im allgemeinen durch Plug-Ins erweiterbar, daß heißt, durch die Anbindung dieser Zusatzmodule kann die Funktionalität des Browsers wesentlich erweitert werden. Dieser wird so in die Lage versetzt, verschiedenste Datentypen zu behandeln. Die Integration von Multimediadaten wie Animationen, Filmen und komplexen Bildformaten wird somit leicht. Weiterhin ist ein Browser in der Lage, externe Anwendungen aufzurufen, um Datentypen anzuzeigen, für deren Darstellung es kein Plug-In gibt.

Konkrete Möglichkeiten für die hypermediale Patientenakte finden sich vor allem im Bereich von Film- und Bilddateien. Radiologie und Nuklearmedizin liefern im Rahmen von Anforderungsergebnissen komplexe Bilder und Filme, die durch die interne Funktionalität eines Browsers nicht verarbeitet werden können. Hier ist es vonnöten, entweder Plug-Ins für den Browser zu verwenden, die in der Lage sind, dem Benutzer der hypermedialen Patientenakte auf einer Station diese Bilder und Filme anzuzeigen, oder Anwendungen auf der Hardware einer Station zu installieren, die diese Formate verarbeiten können und vom Browser als externe Anwendung aufgerufen werden.

Viele aktuelle Anwendungsprogramme, die vor allem im Verwaltungsbereich eingesetzt werden, sind heute in der Lage, ihre Datensätze in das HTML-Format zu konvertieren. Beispiele hierfür sind die Textverarbeitung Word und das Tabellenkalkulationsprogramm Excel, die auf Windows 95/NT-Basis laufen. Die im Rahmen eines Intranets funktionierende hypermediale Patientenakte ist in der Lage, solche externen Dokumente direkt innerhalb des Browsers anzuzeigen und somit eine nahtlose Integration zu erhalten. Somit wird es beispielsweise einem Arzt ermöglicht, einen Arztbrief für einen Patienten mit einem Textverarbeitungsprogramm zu schreiben, wobei er die volle Funktionalitat dieser Anwendung nutzt, anschließend diesen Brief in das HTML-Format zu konvertieren und in die hypermediale Patientenakte zu integrieren, und zuletzt dem Hausarzt des Patienten über E-Mail eine Kopie des Arztbriefs zukommen zu lassen.

Eine andere Möglichkeit ist die Nutzung von Datenbanken. Anstelle oder ergänzend zu einer Konvertierung der Datensätze in das HTML-Format besteht die Möglichkeit einer direkten Kommunikation der hypermedialen Patientenakte mit SQL-fähigen Datenbanken. Die Verwendung der JDBC-Klassen ermöglicht eine einfache Kommunikation mit allen Datenbanken, die in der Lage sind, über SQL zu kommunizieren und für die ein JDBC-Treiber existiert. Die JDBC-Klassen beinhalten einen JBDC-ODBC-Treiber, der direkt über OBDC mit der Datenbankanwendung Access für Windows 95/NT zusammenarbeitet. Konkret bedeutet das, daß die hypermediale Patientenakte in der Lage ist, als dynamisches Front-End für eine oder mehrere darunterliegende Datenbanken zu fungieren, die verteilt innerhalb des Intranets liegen.

Eine Kommunikation mit externen Stellen wird ebenfalls durch die Implementierung im Rahmen eines Intranets unterstützt. Diese Kommunikation kann auf zwei komplementäre Arten geschehen:

Anbindung an das Internet. Da ein Intranet mit derselben Technologie wie das Internet arbeitet, ist es direkt möglich, einen Anschluß an diesen weltweiten Datenverbund zu realisieren. Dies ermöglicht es den Benutzern, ausgedehnte globale Recherchen durchzuführen und mit anderen ebenfalls angeschlossenen Krankenhäusern und Ärzten zu kommunizieren. Ein nicht zu unterschätzendes Problem hierbei ist allerdings die Gewährleistung einer Systemsicherheit. Aus Platz- und Zeitgründen kann hier nicht auf diese Aspekte eingegangen werden. Interessenten seien auf [PS96] verwiesen.

Nutzung von HL7. Eine andere Möglichkeit der externen Kommunikation ist die Integration eines Servers, der in der Lage ist, über das standardisierte Anwendungsprotokoll HL7 (siehe [Pac94]) mit anderen medizinischen Institutionen zu kommunizieren, in das Intranet.

Kapitel 5

Bewertung

In diesem Kapitel wird zunächst eine Bewertung des spezifizierten Systems und des Prototypen vorgenommen. Dies geschieht durch Überprüfung der Anforderungen und Vergleich derselben mit den erzielten Ergebnissen. Dann wird OOHDM im praktischen Einsatz bewertet.

5.1 Bewertung des Systems

Die Reihenfolge der Bewertungen entspricht der in Kapitel 3. Diese sind wiederum den dort vorgestellten Qualitätsanforderungen zugeordnet.

Akzeptanz

Die Akzeptanz der hypermedialen Patientenakte wird durch die Umsetzung der folgenden Aspekte erreicht.

Anwenderfreundliche Benutzeroberflächengestaltung. Diese Anforderung besteht aus den folgenden Unterpunkten.

- *Wiedererkennung.* Der Wiedererkennungswert der hypermedialen Patientenakte ist durch die Tatsache gegeben, daß existierende Formulare des Mathias-Spitals in Rheine für die Gestaltung der Benutzerschnittstellen als Vorbild dienten und nach Möglichkeit direkt umgesetzt wurden. Beispiele hierfür sind das Patientenstammblatt, realisiert durch die ADVs `Patient` und `Neuaufnahme`, sowie die `Röntgenanforderung`. Die Reiterleiste, die bei der papiernen Patientenakte als Signalgeber dient, wurde ebenfalls grafisch in die Benutzeroberfläche integriert.
- *Ergonomie und Präsentation der Benutzeroberfläche.* Die Benutzeroberfläche der hypermedialen Patientenakte sollte nicht mit Informationen überladen werden. Dies wurde in gewissem Maße verhindert, indem Formulare, die mehr als eine Bildschirmseite in Anspruch nehmen, aufgeteilt wurden. Es werden allerdings immer noch relativ viele Informationen auf einer Bildschirmseite präsentiert, was an der Informationsdichte der verwendeten Formulare liegt. Eine weitere Aufsplitterung der zusammengehörigen Daten wurde aus Kohärenzgründen nicht vorgenommen. Die Gruppierung von Informationen erfolgte anhand von logisch zusammengehörigen Informationseinheiten. Beispiele hierfür sind die ADVs `Diagnose`, `Pflege` und `Therapie`. Der Tatsache, daß unterschiedliche Benutzergruppen Informationen auf verschiedene Weise präsentiert haben wollen, wird durch die Spezifikation von mehreren navigationellen Sichten Rechnung getragen.

- *Dateneingabe und Datenausgabe.* Die Haupteingabemöglichkeit ist bei der hypermedialen Patientenakte die Tastatur. Da die Eingabe von Text vor allem bei Standardmaßnahmen im Bereich von Pflegedokumentation, Therapiemaßnahmen und Medikation häufige Redundanz erfordert, ist hier eine Vorgabe von Standardmaßnahmen und wichtigen Medikamenten über Auswahlmenüs möglich. Die Direkteingabe von Vitalwerten durch Überwachungsgeräte konnte im Prototypen aus Zeitgründen nicht implementiert werden, könnte aber über eine TCP/IP-Netzwerkverbindung realisiert werden. Die Darstellung von Multimediadaten wird durch interne Fähigkeiten oder Plug-Ins des Browsers ermöglicht. Die Aufrufe der entsprechenden Funktionen werden aus dem Java-Applet heraus vorgenommen, welches die Funktionalität des Browsers nutzen kann. Eine gleichzeitige Darstellung von mehreren Bildern wird direkt durch die Möglichkeit, mehrere Fenster zu öffnen, unterstützt. Es können auch mehrere Datensätze gleichzeitig angezeigt werden, da der Browser mehrmals gleichzeitig aufgerufen werden kann.
- *Hilfesystem und Nachschlagewerk.* Zu jedem Knoten existiert ein Hilfefenster, welches dem Benutzer die Besonderheiten und Anwendungsmöglichkeiten des jeweiligen Knotens erklärt. Die Realisierung der hypermedialen Patientenakte als WWW-Intranetsystem ermöglicht eine nahtlose Integration von HTML-Online-Bibliotheken und -Nachschlagewerken des medizinischen Anwendungsbereiches. Somit können Pflege, Diagnostik und Therapie des Patienten durch umfangreiches und leicht zugängliches Referenzmaterial effizient unterstützt werden. Es ist weiterhin auf einfache Art und Weise möglich, elektronische Enzyklopädien und Dienstvorschriften in Hypertextform in das System zu integrieren.

Terminologie. Bei der Spezifikation und Implementation der hypermedialen Patientenakte wurde darauf geachtet, existierende Begriffe und geläufige Phrasen der Krankenhausarbeitsumgebung zu verwenden. Dies geschah dadurch, daß existierende Formulare direkt übernommen wurden und Auswahllisten mit vorgegebenen Pflege- und Therapiemaßnahmen für das Personal bereitgestellt werden.

Anwenderorientierte Umsetzung des Ist-Zustands. Eine anwendernahe Entwicklung der hypermedialen Patientenakte ist durch die Verwendung der Ergebnisse der PG 263 *PROMETHEUS* gegeben, die in enger Zusammenarbeit mit dem Mathias-Spital in Rheine erstellt wurden. Dabei wurden die existierenden Arbeitabläufe auf den Stationen vor Ort analysiert und systematisch in einer Spezifikation umgesetzt. Zusätzlich erfolgte die Entwicklung in zwei Iterationen, wobei der Prototyp der ersten Iteration in Rheine vorgestellt wurde und auf der Basis dieser Rücksprache eine Überarbeitung des Systems stattfand.

Zugänglichkeit

Die Anforderung nach Zugänglichkeit wird durch die Umsetzung der Anforderung nach Transparenz erreicht.

Transparenz. Die Implementation auf Basis von Java-Applets im Rahmen eines Intranets macht es für den Benutzer überflüssig, sich um die Lokalisierung von Daten zu kümmern. Ein mögliches Szenario ist in Abbildung 5.1 dargestellt. Die hypermediale Patientenakte kann eine verteilte Datenbasis besitzen, die verschiedene Institutionen wie andere Krankenhäuser, Ärzte u.a. umfaßt. Die wichtigsten, in diesen Basen enthaltenen Daten werden lokal im Krankenhaus in einer zentralen Basis als Notfalldaten gespeichert, um Ausfälle im Netzwerk zu kompensieren. Die Schnittstelle der hypermedialen Patientenakte im Krankenhaus präsentiert den Benutzern alle Informationen in zusammengefaßter Form.

Verwaltbarkeit

Die hypermediale Patientenakte wurde verwaltbar, indem die folgenden Anforderungen umgesetzt wurden.

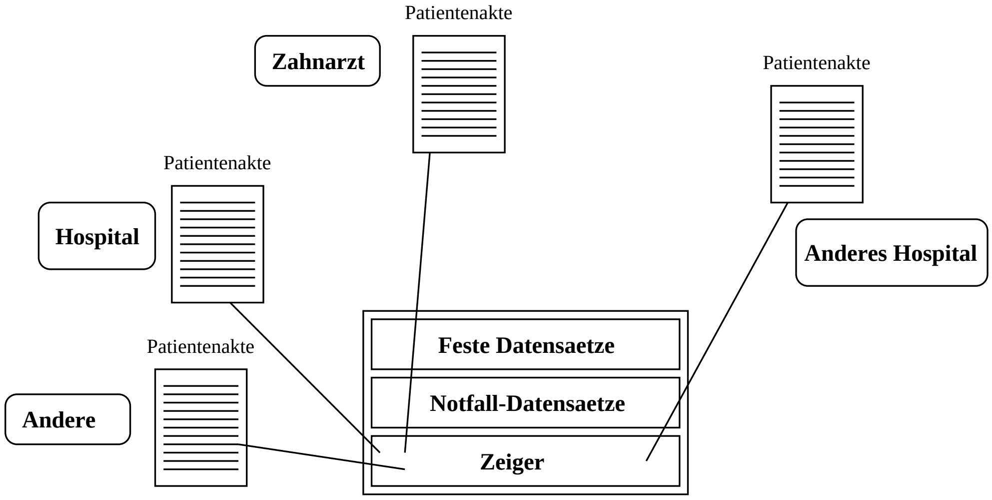


Abbildung 5.1: Eine hypermediale Patientenakte mit einer verteilten Datenbasis

Terminplanung. Eine Terminplanung wurde im Rahmen der hypermedialen Patientenakte spezifiziert. Mit dieser ist es möglich, Anforderungen zu terminieren und beliebig Termine zwischen Patienten, Stationen, Leistungsstellen und Pflegepersonal zu vereinbaren.

Administrative Unterstützung. Die Verringerung des Aufwands für administrative Fähigkeiten und eine dadurch bedingte Erhöhung der Wirtschaftlichkeit wird durch daß System erreicht, indem Dokumentation und Anforderungserstellung erleichtert werden. Dies geschieht durch die oben schon genannte Auswahl von Standardmaßnahmen und durch die sofortige elektronische Versendung von Anforderungen. Ein wichtiges Problem kann allerdings nicht direkt gelöst werden: Vor allem älteres Pflegepersonal hat keine oder wenig Erfahrung mit schneller Tastatureingabe. Falls viele Daten manuell eingegeben werden müssen, kann hier ein nicht unbeträchtlicher Nettozeitverlust eintreten. Weitere Möglichkeiten, dieses zu vermeiden, sind neben der implementierten Vorgabe von Standardmaßnahmen Schulungen und Kurse durch das Krankenhaus.

Direkte Berechnung der finanziellen Aspekte des Krankenhausaufenthalts. Durch eine Spei cherung der Patientendaten in einer SQL-fähigen Datenbank (Access im Falle des Prototypen) erfolgt, wird eine einfache Kommunikation mit externen Anwendungen, die für Berechnungen und Kalkulationen geeignet sind, ermöglicht. Im Falle des Prototypen liegt eine direkte Einbettung in die Microsoft-Office-Umgebung vor. Alternativ kann die Funktionalität von Java genutzt und ein eigenes Abrechnungs-Applet mit Grafikunterstützung implementiert werden.

Unterstützung von Forschung und Statistik. Auch hier leistet der Systementwurf, mit einer unterliegenden Datenbank zu arbeiten, wertvolle Dienste. Es kann entweder mit anderen Werkzeugen auf die Datenbanken zugegriffen werden, oder es können Plug-Ins für den Browser oder Java-Applets implementiert werden, die die gewünschte Funktionalität realisieren. Die Grafikfähigkeiten von Java sind hier eine große Hilfe.

Integration von externen Anwendungen. Diese Anforderung wird durch die Benutzung eines Browsers umgesetzt, für den verschiedene Plug-Ins verwendet werden können, um die unterschiedlichsten Datentypen zu integrieren. Auf diesen Aspekt wurde ausführlich in Abschnitt 4.5.4 eingegangen.

Umfassende Patienteninformation. Die Darstellung aller Aspekte der stationären Patientenverwaltung im Rahmen eines Browsers bewirkt eine umfassende und einheitliche Patienteninformation für alle Benutzer der hypermedialen Patientenakte.

Adäquate Datenspeicherung. Die Benutzung SQL-fähiger Datenbanken, die über eine JDBC-Schnittstelle angesprochen werden, sorgt für eine angemessene Speicherung der relevanten Daten.

Vertraulichkeit

Die Gewährleistung von Vertraulichkeit geschieht durch die Berücksichtigung folgender Anforderungen.

Datenschutz. Die erforderlichen Datenschutzmaßnahmen werden im Rahmen der Intranet-Realisierung des Systems getroffen. Hierfür gibt es verschiedene Möglichkeiten, die an dieser Stelle nicht näher erläutert werden. Es sei hier auf die entsprechende Fachliteratur verwiesen [PS96].

Sichten. Um der Tatsache Rechnung zu tragen, daß die verschiedenen mit der hypermedialen Patientenakte arbeitenden Benutzergruppen unterschiedliche Datenpräsentationen wünschen, wurden die fünf Sichten Schwesternsicht, Arztsicht, Leistungsstellensicht, Verwaltungssicht und Besuchersicht spezifiziert.

Legale Dokumentation. Die Speicherung der relevanten Daten in einer Datenbank macht es einfach, jede Maßnahme mit einer dazugehörigen Benutzeridentifikation zu versehen. Durch entsprechende Definition von Tabellen in einer relationalen Datenbank werden Benutzer-ID und Maßnahme miteinander verknüpft.

Sicherheit. Die Realisierung der hypermedialen Patientenakte als verteilte Anwendung sorgt für eine Sicherheit gegenüber dem Ausfall von Teilkomponenten.

Effektivität

Effektivität der hypermedialen Patientenakte wird durch die folgenden Anforderungen erreicht.

Vereinfachung und Automatisierung von Routinetätigkeiten. Diese Anforderung umfaßt die folgenden Unterpunkte.

- *Pflegeplanung und -dokumentation.* Die Pflegeplanung und -dokumentation wird durch die direkte Anbindung an eine SQL-fähige Datenbank mittels der JDBC-Klassen effizient unterstützt. Die Auswahl von Standardmaßnahmen aus einer Auswahlliste, die direkt in die Pflegeplanung übernommen werden können, leistet hier ebenfalls gute Dienste.
- *Diagnoseunterstützung und Therapieplanungssystem.* Die Implementation des Systems auf WWW-Basis ermöglicht eine einfache Integration von externen Ressourcen in die Therapieplanung. Weltweit verteilte medizinische Datenbanken mit Informationen zu Diagnose und Therapie können vom Arzt für Behandlungszwecke genutzt werden.
- *Minimierung der Patientenverweildauer.* Eine Minimierung der Patientenverweildauer wird durch eine effiziente Terminplanung, Verwendung von Standardtherapien und integrierte Verwaltung des gesamten Patientenaufenthalts erreicht.
- *Verbesserte Patientenpflege.* Durch die Tatsache, daß durch eine computerisierte Patientenakte die Organisation und Pflegeplanung durch Zeitersparnis bei administrativen Aufgaben verbessert wird, profitiert der Patient von der Tatsache, daß dadurch mehr Zeit für die eigentliche Pflege zur Verfügung steht.

Mehrbenutzersystem. Die Implementierung im Rahmen eines Intranets auf WWW-Basis bettet das System in eine funktionierende und erprobte Mehrbenutzerumgebung ein.

Kommunikationsförderung. Die Realisierung des Systems im Rahmen eines Intranets stellt den Benutzern die Funktionalität des verwendeten Browsers zur Verfügung. Diese umfaßt auch die Nutzung eines E-Mail-Systems, mit dem verschiedene Benutzer und Leistungsstellen miteinander kommunizieren können.

Erhöhung der Wirtschaftlichkeit. Diese Anforderung wird durch dieselben Aspekte wie bei der Anforderung nach administrativer Unterstützung erreicht (siehe unter *Verwaltbarkeit*).

Zeitliche Effektivität. Es muß gewährleistet sein, daß das System eine angemessene Reaktionszeit besitzt. Dies ist in der gegenwärtigen Implementation bedingt der Fall. Die verwendete Programmiersprache Java ist im Vergleich mit anderen Sprachen wie C++ recht langsam. Zufriedenstellende Ergebnisse wurden auf einem Pentium-PC erzielt.

Direkte Überwachungsdatenspeicherung und -auswertung. Diese Anforderung konnte aus zeitlichen Gründen nicht mehr umgesetzt werden.

Erweiterbarkeit und Flexibilität

Diese Qualitätsanforderung wird durch die beiden untengenannten Anforderungen abgedeckt.

Verteiltheit und Dezentralisation. Die Implementation als Intranet erfüllt diese Anforderung.

Modularität und Offenheit. Leichte Erweiterbarkeit, Verbesserung und Weiterentwicklung des Systems werden durch die Verwendung von Java, welches durch seine objektorientierte Struktur die drei Punkte bei effizienter Programmierung direkt unterstützt, erfüllt.

Integrität

Die technische Anforderung nach Konsistenz realisiert diese Qualitätsanforderung.

Konsistenz. Die Implementation als Intranet trägt zur Erfüllung dieser Anforderung bei. Das unter der Anforderung nach Transparenz unter *Zugänglichkeit* vorgestellte Szenario einer verteilten Datenbasis mit einem lokal gespeicherten Notfall-Datensatz ist hier ebenfalls hilfreich.

Portierbarkeit

Die Anforderung nach Plattformunabhängigkeit trägt hierzu ihren Teil bei.

Plattformunabhängigkeit. Die implementierungsunabhängige Systementwicklung nach OOHDM sowie die Implementierung auf der Basis eines WWW-Intranets stellen sicher, daß das System nicht nur auf eine Plattform zugeschnitten ist, sondern in der Lage ist, verschiedene Hardware zu verbinden.

Nützlichkeit

Diese Qualitätsanforderung wird erreicht, indem eine Effizienzsteigerung erreicht wird, die unter dem Punkt *Effektivität* behandelt wurde.

Lesbarkeit

Diese Qualitätsanforderung wird erreicht, indem die unter *Akzeptanz* angesprochenen Punkte berücksichtigt werden.

5.2 Bewertung von OOHDM im Einsatz

Die Bewertung von OOHDM im praktischen Einsatz wird anhand der vier behandelten Phasen und eines Gesamteindrucks geschildert.

Konzeptentwurf. Der Konzeptentwurf von OOHDM bedient sich etablierter und bewährter Prinzipien der Objektorientierung (siehe Kapitel 2.4.3), um den Anwendungsbereich zu modellieren. Diese sind vollauf geeignet, um die Modellierung umfassend und vollständig durchzuführen.

Navigationsentwurf. Die Art der Dokumentation des Navigationsentwurfs von OOHDM, bei der eine Reihe unterschiedlicher Beschreibungsformen eingesetzt werden, ist eine der Schwächen dieser Methode. Es existieren mehrere technische Papiere aus verschiedenen Jahren, die den Navigationsentwurf uneinheitlich dokumentieren. Vor allem der Unterschied zwischen Zugangsstrukturen und navigationellen Kontexten ist nicht immer klar herausgestellt worden. Ein weiterer Schwachpunkt ist das Fehlen einer übersichtlichen grafischen Notation, die Kontexte, Zugangsstrukturen, Verknüpfungen und Knotenklassen zusammenfassend darstellen kann. Ansonsten ist der Navigationsentwurf eine für Hypermediasysteme eminent wichtige Phase, die ihr Ziel, die Modellierung der Navigation in einem solchen System, erreicht.

Abstrakter Schnittstellenentwurf. Der abstrakte Schnittstellenentwurf deckt einen interessanten Bereich, die abstrakte Modellierung der grafischen Benutzeroberfläche eines Hypermediasystems, ab. Die verwendeten Konstrukte (siehe Kapitel 2.4.3) sind vollends geeignet, um diesen Aspekt zu modellieren. Ein schwerwiegender Nachteil dieser Phase ist allerdings das Fehlen eines Werkzeugs, welches hier Unterstützung liefert, da es zwar theoretisch sinnvoll ist, alle möglichen Zustände einer Benutzeroberfläche 'auf dem Papier' zu spezifizieren, dies unter dem Gesichtspunkt einer praktischen Umsetzung aber eine sehr mühselige Arbeit ist, die sich ohne GUI-Builder und einem Werkzeug zur Überprüfung der Konsistenz der Modelle nur bedingt auszahlt.

Implementation. Die unter OOHDM erzeugten Modelle bilden eine gute Grundlage für die Implementierung eines Hypermediasystems. Die Umsetzung der Modelle in eine objektorientierte Programmiersprache ist relativ einfach und direkt möglich.

Gesamteindruck. OOHDM ist eine durchaus zu empfehlende Methode, die allerdings noch unter einigen Schwächen leidet. Sie ist besonders für Informationssysteme geeignet, aber es sollten sich auch Anwendungen mit mehr Funktionalität damit spezifizieren lassen. Eine einheitliche Dokumentation sowie eine den Entwicklungsprozeß durchgängig unterstützende Entwicklungsumgebung, welche Dienste wie Datenverwaltung, Entwicklungstransaktionen, Konfigurationsmanagement und andere beinhaltet, sind wünschenswerte Ergänzungen dieser Methode.

Kapitel 6

Zusammenfassung und Ausblick

Dieses Kapitel gibt zunächst in Abschnitt 6.1 eine Zusammenfassung dieser Diplomarbeit wieder. Anschließend wird in Abschnitt 6.2 ein Ausblick auf mögliche Fortsetzungen dieser Arbeit vorgenommen.

6.1 Zusammenfassung

Im Verlauf dieser Diplomarbeit wurden die folgenden Schritte durchgeführt:

Betrachtung der Problematik der hypermedialen Patientenakte. Es wurden als erstes Grundlagen ermittelt, die für Aspekte der Bereiche Hypermediaentwicklung und computerunterstützter Patientenakte relevant waren. Dabei wurden für den Bereich Hypermediaentwicklung Entwurfsmethoden für Hypermediasysteme betrachtet, während für den Bereich der computerunterstützten Patientenakte auf die Ergebnisse der Projektgruppe 263 *PROMETHEUS* zurückgegriffen werden konnte. Es wurde ein Anforderungskatalog entworfen, der die fachlichen und technischen Anforderungen an eine hypermediale Patientenakte spezifiziert.

Vergleich von Methoden der systematischen Hypermediaentwicklung. Im Verlauf dieser Phase wurden Entwicklungsmethoden zum systematischen Entwurf von Hypermediaanwendungen betrachtet und verglichen. Es wurde die Methode OOHDM zur Spezifikation der hypermedialen Patientenakte ausgewählt.

Spezifikation der hypermedialen Patientenakte nach OOHDM. Hier wurde die hypermediale Patientenakte in den ersten drei OOHDM-Entwurfsschritten Konzeptentwurf, Navigationsentwurf und abstrakter Schnittstellenentwurf modelliert.

Prototypische Implementierung. Als vierter Entwicklungsschritt wurde die hypermediale Patientenakte auf HTML/ Java-Applet-Basis für ein WWW-Intranet prototypisch implementiert.

Bewertung. Zuletzt wurde eine Bewertung der Ergebnisse anhand der spezifizierten Anforderungen und der Methode OOHDM im praktischen Einsatz vorgenommen.

6.2 Ausblick

Die Entwicklung der hypermedialen Patientenakte nach OOHDM und anschließende Implementierung in Java hat die meisten Anforderungen erfüllt, liefert aber auch Ansatzpunkte für Weiterentwicklungen und Verbesserungen.

Die zwei wichtigsten Punkte sind:

Eine Entwicklungsumgebung für den Entwurfsprozeß nach OOHDM. OOHDM ist nach Meinung des Autors dieser Diplomarbeit eine gute Methode zum Entwurf von Hypermediasystemen, leidet aber unter dem Fehlen einer Entwicklungsumgebung, welche den Entwicklungsprozeß durch die drei Entwurfsphasen durchgängig unterstützt. Da die Modelle aufeinander aufbauen, müssen im Navigations- oder abstrakten Schnittstellenentwurf vorgenommene Änderungen rückwirkend von Hand in den vorhergehenden Modellen vorgenommen werden. Eine Unterstützung des abstrakten Schnittstellenentwurfs durch einen GUI-Builder, der in der Lage ist, Prozedurköpfe und Schnittstellenobjekte für verschiedene Implementierungsplattformen zu generieren, wäre ebenfalls wünschenswert.

Entwicklung der hypermedialen Patientenakte als Java-Applikation. Die aktuelle Realisierung des Prototypen auf Applet-Basis besitzt Vor- und Nachteile. Der Vorteil dieser Realisierung liegt in der Nutzung der Funktionalität des Browser-Werkzeugs im Rahmen eines WWW-Intranets, wodurch einerseits die Einbindung von externen Anwendungen und Multimediadaten leichtgemacht wird, andererseits die Nutzung einer sich als weltweiten Standard etablierenden Plattform vorgenommen wird.

Ein Nachteil liegt darin, daß appletspezifische Besonderheiten berücksichtigt werden müssen, die eine direkte objektorientierte Umsetzung der Spezifikationsmodelle erschweren. Dies ließe sich durch eine reine Java-Applikation direkter realisieren. Diese hat aber wiederum den Nachteil, daß kein existierender Browser genutzt werden kann, sondern dessen Funktionalität in die Applikation integriert werden müßte, was einen erheblichen Mehraufwand an Enwicklungszeit und -kosten mit sich bringt.

Literaturverzeichnis

[AW92] M.K. Amatayakul und M.J. Wogan. Record Administrator's Needs for Computer-based Patient Records. In *[BC92]*, Kapitel 1/8, Seiten 57–66. 1992.

[Bal94] V. Balasubramanian. State of the Art Review on Hypermedia Issues And Applications. Technischer Bericht, Graduate School of Management, Rutgers University, Newark, New Jersey, 1994.

[BBI96] V. Balasubramanian, M. Bieber und T. Isakowitz. Systematic Hypermedia Design. Technischer Bericht, 1996.

[BC92] M. Ball und M. Collen, Hrsg. *Aspects of the Computer-based Patient Record.* Springer-Verlag, 1992.

[BCF+96a] I. Basusta, S.A. Cengiz, T. Frenzel, M. Gronek, H. Heinecke, J. Hilker, M. Lange, M. Patoka, R. Privighitorita, T. Proske, C. Stroetmann und A. Witte. Spezifikation eines Systems zum Behandlungsmanagement im Krankenhaus. Endbericht, PG 263 $\mathcal{PROMETHEUS}$, Lehrstuhl für Softwaretechnologie, Universität Dortmund, 1996.

[BCF+96b] I. Basusta, S.A. Cengiz, T. Frenzel, M. Gronek, H. Heinecke, J. Hilker, M. Lange, M. Patoka, R. Privighitorita, T. Proske, C. Stroetmann und A. Witte. Spezifikation eines Systems zum Behandlungsmanagement im Krankenhaus. Spezifikation, PG 263 $\mathcal{PROMETHEUS}$, Lehrstuhl für Software-Technologie, Fachbereich Informatik, Universität Dortmund, 1996.

[Bie93] M. Bieber. Automating Hypermedia for Decision Support. *Hypermedia*, 4(2), 1993.

[BL92] T. Berners-Lee. An Architecture for Wide Area Hypertext. *Hypertext '91 Poster Abstract, SIGLINK Newsletter*, Dezember 1992.

[BMG+89] R. Balzer, M.Begeman, P.K. Garg, M. Schwartz und B. Shneiderman. Panel Discussion on Hypertext and Software Engineering. In *Proceedings of Hypertext '89.* ACM Press, 1989.

[BR87] J. Bigelow und V. Riley. Manipulating Source Code in Dynamic Design. In *Proceedings fo Hypertext '87.* ACM Press, 1987.

[BS94] S. Barbosa und D. Schwabe. Navigation Modeling in Hypermedia Applications. Technischer Bericht, PUC Rio, Departamento de Informatica, 1994.

[Buc92] W.H. Buckley. Patient's Needs for Computer-based Patient Records. In *[BC92]*, Kapitel 1/5, Seiten 36–39. 1992.

[Bud96] G. Budenz. *Asymetrix ToolBook Version 4.* Tewi Buch, 1996.

[Bus45] V. Bush. As We May Think. *The Atlantic Monthly*, Juli 1945.

[CCCL94] L.M.F. Carneiro, M.H. Coffin, D.D. Cowan und C.J.P. Lucena. ADVcharts: a Visual Formalism for Highly Interactive Systems. In M.D. Harrison und C. Johnson, Hrsg., *Software Engineering in Human-Computer Interaction*. Cambridge University Press, 1994.

[CHB92] D. Coleman, F. Hayes und S. Bear. Introducing Objectcharts or how to use Statecharts in Object-Oriented Design. *IEEE Transactions on Software Engineering*, 18(1), Januar 1992.

[CL95] D.D. Cowan und C.J.P. Lucena. Abstract Data Views: An Interface Specification Concept to Enhance Design for Reuse. *IEEE Transactions on Software Engineering*, 21(3), Marz 1995.

[Dav92] A.R. Davies. Health Care Researcher's Needs for Computer-based Patient Records. In *[BC92]*, Kapitel 1/7, Seiten 46–56. 1992.

[Eng63] D.C. Engelbart. A Conceptual Framework for the Augmentation of Man's Intellect. *Vistas in Information Handling, Spartan Books*, 1, 1963.

[Fla97] D. Flanagan. *JavaScript: The Definitive Guide, 2nd Edition*. O'Reilly and Associates Inc., 1997.

[Fri88a] L. Friedlander. *The Shakespeare Project Interactive Multimedia*. Microsoft Press, 1988.

[Fri88b] M.E. Frisse. Searching for Information in a Hypertext Medical Handbook. *Communications of the ACM*, Juli 1988.

[FSLS80] P.J. Fischer, W.C. Stratmann, H.P. Lundsgaarde und D.J. Steele. User Reaction to PROMIS: Issues Related to Acceptability of Medical Innovations. In J. O'Neil, Hrsg., *Proceedings of the Fourth Annual Symposium on Computer Applications in Medical Care*. IEEE Computer Society Press, 1980.

[GS97] B. Gesell und T. Schmal. Das pro7-System als Basis fuer den Aufbau eines verteilten krankenhausweiten Informationssystems. In *SoftKis '97*, 1997.

[GSP93] F. Garzotto, D. Schwabe und P. Paolini. HDM - A Model Based Approach to Hypermedia Application Design. In *ACM Transaction on Information Systems*, Band 11, Seiten 1–26, Januar 1993.

[Gun96] S. Gundavaram. *CGI Programming on the World Wide Web*. O'Reilly and Associates Inc., 1996.

[Hal88] F.G. Halasz. *NoteCards: A Multimedia Idea Processing Environment, Interactive Multimedia*. Microsoft Press, 1988.

[HBvR94] L. Hardman, D.C.A. Bulterman und G. van Rossum. The Amsterdam Hypermedia Model: Adding Time, Structure and Context to Hypertext. Communications of the ACM, 1994.

[HPSS87] D. Harel, A. Pnueli, J.P. Schmidt und R. Sherman. On the Formal Semantics of Statecharts. In *Proceedings of the 2nd IEEE Symposium on Logic in Computer Science, Ithaca, N.Y.*, Juni 1987.

[HS94] F.G. Halasz und M. Schwartz. The Dexter Hypertext Reference Model. In *Communications of the ACM*, Band 37, Seiten 50–63, Februar 1994.

[IKK96] T. Isakowitz, A. Kamis und M. Koufaris. Extending the Capabilities of RMM: Russian Dolls and Hypertext. Technischer Bericht, Stern School of Business, New York University, 1996.

[ISB95] T. Isakowitz, E. Stohr und P. Balasubramanian. RMM: A Methodology for Structured Hyermedia Design, 1995.

[IT90] P.M. Irish und R.H. Trigg. *Supporting Collaboration in Hypermedia: Issues and Experiences, The Society of Text.* MIT Press, 1990.

[Jac92] I. Jacobsson. *OOSE, A Use-case Driven Approach.* Addison-Wesley, 1992.

[JG66] R.A. Jydstrup und M.J. Gross. Cost of Information Handling in Hospitals. In *Health Services Research*, Kapitel 1, Seiten 235–261. 1966.

[JN95] R. Jones und A. Nye. *HTML und das World Wide Web.* O'Reilly International Thomson Verlag, 1995.

[Mac97] Macromedia. Director Software. WWW-Seite, 1997. http://www.macromedia.com/software/director.

[Mar92] C.Z. Margolis. Clinician's Needs for Computer-based Patient Records. In *[BC92]*, Kapitel 1/2, Seiten 12–15. 1992.

[McD92] C.J. McDonald. Physician's Needs for Computer-based Patient Records. In *[BC92]*, Kapitel 1/1, Seiten 3–11. 1992.

[McH92] M.L. McHugh. Nurse's Needs for Computer-based Patient Records. In *[BC92]*, Kapitel 1/3, Seiten 16–29. 1992.

[MSS96] S. Middendorf, R. Singer und S. Strobel. *Java Programmierhandbuch und Referenz.* dpunkt, Verlag für digitale Technologie, 1996.

[MT90] H. Maurer und I. Tomek. Broadening the Scope of Hypermedia. *Hypermedia*, 2(3), 1990.

[MX95] D. Christodoulakis M. Xenos. Towards a High Quality Computer-Based Patient Record. Technischer Bericht, AMICE'95 Strategic Alliances between Patient Documentation and Medical Informatics, 1995.

[Nel65] T. Nelson. A File Structure for the Complex, the Changing and the Indeterminate. In *ACM 20th National Conference*, 1965.

[Nel80] T. Nelson. Replacing the Printed Word: A Complete Literary System. In *Information Processing '80*, 1980.

[Pac94] J. Paczkowski. Offene Kommunikation mit dem standardisierten Anwendungsprotokoll HL7. *Das Krankenhaus*, 3, 1994.

[PM96] P. Patel und K. Moss. *Java Database Programming with JDBC.* Coriolis Group Books, 1996.

[Pro97] T. Proske. Entwicklung einer Hypermedialen Patientenakte mit einem modellbasierten Ansatz. Technischer Anhang. Technischer Anhang zur Diplomarbeit, 1997.

[Pry92] A.T. Pryor. Current State of Computer-based Patient Record Systems. In *[BC92]*, Kapitel 2/1, Seiten 67–82. 1992.

[PS96] K. Pommerening und E. Scheidt. Sicherheitsempfehlungen zum Internet-Anschluß von Krankenhäusern. WWW-Seite, August 1996. http://www.uni-mainz.de/FB/Medizin/IMSD/AGDatenschutz/Internet.html.

[RBP+91] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy und W. Lorensen. *Object Oriented Modeling and Design.* Prentice Hall Inc., 1991.

[Rei82] W. Reisig. *Petrinetze.* Springer-Verlag, 1982.

[Ric70] R. Richart. Evaluation of a Medical Data System. In *Computers in Biomedical Research*, Kapitel 3, Seiten 415–425. 1970.

[RM97] L. Rosenfeld und P. Morville. *Information Architecture for the World Wide Web.* O'Reilly and Associates Inc., 1997. erscheint voraussichtlich im November 1997.

[ROK95] ROKD. Open Hospital System. Broschüre, Februar 1995.

[RS94] G. Rossi und D. Schwabe. From Domain Models to Hypermedia Applications: an Object-Oriented Approach. Technischer Bericht, Dep. de Informatica, PUC, Rio de Janeiro, Brazil, 1994.

[RSLC95] G. Rossi, D. Schwabe, C.J.P. Lucena und D.D. Cowan. An Object-Oriented Model for Designing the Human-Computer Interface Of Hypermedia Applications. In *Proceedings of the International Workshop on Hypermedia Design (IWHD'95), Montpellier*, 1995.

[RT90] U. Rao und M. Turoff. Hypertext Functionality: A Theoretical Framework. *International Journal of Human-Computer Interaction*, 1990.

[Sch96] R. Schulmeister. *Grundlagen hypermedialer Lernsysteme: Theorie - Didaktik - Design.* Addison-Wesley, 1996.

[Sen92] C. Sennett. The Computer-based Patient Record: The Third Party Payer's Perspective. In *[BC92]*, Kapitel 1/6, Seiten 40–45. 1992.

[SHT89] N.A. Streitz, J. Hannemann und M. Thuring. From Ideas and Arguments to Hyperdocuments: Travelling through Activity Spaces. In *Proceedings of Hypertext '89.* ACM Press, 1989.

[SL89] J.L. Schnase und J.J. Leggett. Computational Hypertext in Biological Modelling. In *Proceedings of Hypertext '89.* ACM Press, 1989.

[SRB95a] D. Schwabe, G. Rossi und D.J. Barbosa. Abstraction, Composition and Lay-Out Definition Mechanisms in OOHDM. In *Electronic Procedings of the ACM Workshop on Effective Abstractions in Multimedia*, November 1995.

[SRB95b] D. Schwabe, G. Rossi und S.D.J. Barbosa. Systematic Hypermedia Application Design with OOHDM. Technischer Bericht, Dep. de Informatica, PUC, Rio de Janeiro, Brazil, 1995.

[Ull92] E. Ullian. Hospital Administrator's Needs for Computer-Based Patient Records. In *[BC92]*, Kapitel 1/4, Seiten 30–35. 1992.

[vRJMB93] G. van Rossum, J. Jansen, K.S. Mullender und D.C.A. Bulterman. CMIFed: A Presentation Environment for Portable Hypermedia Documents. In *Proceedings of the First International Conference on Multimedia, Anaheim, California*, Seiten 183–188, August 1993.

[WB90] R. Wirfs-Brock. *Designing OO Software.* Prentice-Hall, 1990.

[Wee69] L.L. Weed. *Medical Records, Medical Education and Patient Care: The Problem-Oriented Record as a Basic Tool.* Year Book Medical Publishers, 1969.

[YSGM88] N. Yankelovich, K. Smith, N.L. Garrett und N. Meyrowitz. *Issues in Designing a Hypermedia Document System, Interactive Multimedia.* Microsoft Press, 1988.

[YW89] E. Yoder und T.C. Wettach. Using Hypertext in a Law Firm. In *Proceedings of Hypertext '89.* ACM Press, 1989.