# Nachvollziehbarkeit von den Anforderungen zum entwickelten System durch den Einsatz von Piktogrammen

Wilhelm Hasselbring
Lehrstuhl für Software-Technologie, Universität Dortmund, D-44221 Dortmund
E-mail: hasselbring@ls10.informatik.uni-dortmund.de

**Zusammenfassung**

Ein Piktogramm ist eine stilisierte bildliche Darstellung zur Informationsvermittlung. Solche graphischen Darstellungen haben i.a. eine spezifische Bedeutung für verschiedene Anwendungsbereiche (z.B. ein Halteverbotszeichen für den Straßenverkehr).

Im vorliegenden Beitrag berichten wir über unsere Erfahrungen mit dem Einsatz von Techniken zur interdisziplinären Anforderungsanalyse für Krankenhausinformationssysteme und über die dabei erreichte Nachvollziehbarkeit von den Anforderungen zum entwickelten System mit Hilfe des durchgängigen Einsatzes von anwendungsorientierten Piktogrammen in den verschiedenen Phasen der Modellierung bis hin zur Implementierung, wobei auch das frühzeitige Prototyping eine wichtige Rolle spielt. Die Unterstützung der Nachvollziehbarkeit durch den Einsatz von Piktogrammen in Modellierungswerkzeugen wird kurz diskutiert.

## Inhaltsverzeichnis

# 1 Einleitung

In graphischen Benutzungsschnittstellen für Software-Systeme werden häufig *Ikons* [Git86, Bra97] verwendet, die Piktogramme darstellen, um mit den Ikons eine bestimmte Metapher verbinden zu können. Softwaretechnisch können Piktogramme als Vektor- oder Rastergraphiken repräsentiert werden. Wir diskutieren den Einsatz von Piktogrammen in den verschiedenen Phasen der Entwicklung eines Krankenhausinformationssystems, wobei insbesondere für die Anwender ein nachvollziehbarer Entwicklungsprozeß von den Anforderungen zum Anwendungssystem erreicht werden konnte.

Bei der Entwicklung von Software für den Krankenhausbereich nimmt die Anforderungsanalyse eine zentrale Stellung ein, um die *tatsächlichen* Anforderungen der Anwender zu erfüllen. Der Anwendungsbereich Krankenhaus zeichnet sich besonders durch ein hohes Maß an unterschiedlichen Aufgaben und teilweise gegensätzlichen Anforderungen unterschiedlicher Anwendergruppen, die sich grob den verschiedenen Berufsgruppen im Krankenhaus zuordnen lassen, aber auch durch deren Zusammenarbeit aus.

Ein Krankenhausinformationssystem ist als das gesamte informationsverarbeitende und informationsspeichernde Teilsystem eines Krankenhauses zu betrachten [WZB+96]. Der rechnerunterstützte Teil eines Krankenhausinformationssystems ist der Teil, der Rechner und Rechnernetze einsetzt. Bei der Entwicklung eines rechnerunterstützten Krankenhausinformationssystems ist eine Berücksichtigung aller Informationsflüsse wichtig. Eine isolierte Betrachtung des rechnerunterstützten Teils ist nicht ausreichend.

Die Erfahrungen aus einem interdisziplinären Kooperationsprojekt zwischen der Universität Dortmund (Fachbereich Informatik und Fachbereich Statistik) und dem Herzzentrum im Klinikum Wuppertal mit dem Einsatz von Piktogrammen in der Modellierung bei der Neuentwicklung eines Informationssystems für die Kardiologie werden insbesondere in Bezug auf die Nachvollziehbarkeit diskutiert. Dieses Informationssystem dient der Unterstützung der medizinischen Forschung durch die statistische Auswertung von Diagnose- und Therapiedaten.

Als zentrale Technik für die Anforderungsanalyse wurden im vorgestellten Projekt Kooperationsdiagramme eingesetzt, die es erlauben, zusammen mit den Anwendern die Anforderungen an ein zu entwickelndes Informationssystem auf der Ebene von Informationsflüssen in der Klinik zu analysieren, wobei auch der nicht durch Rechner unterstützte Informationsaustausch berücksichtigt wird. Die resultierenden Kooperationsdiagramme bieten dann die Grundlage für die Strukturierung der Daten in der Datenbank, die Spezifikation des Verhaltens der Applikation und die Gestaltung der Benutzungsschnittstellen.

In Abschnitt 2 diskutieren wir zunächst einige Aspekte der Nachvollziehbarkeit in der Software-Entwicklung, bevor in Abschnitt 3 die Fallstudie präsentiert wird. Einige Fragen der Werkzeugunterstützung für den Einsatz von Piktogrammen in der Modellierung werden in Abschnitt 4 diskutiert. Abschnitt 5 liefert eine Zusammenfassung.

# 2 Nachvollziehbarkeit

Nachvollziehbarkeit (traceability) in der Software-Entwicklung kann als die Möglichkeit zur Beschreibung und zum Verfolgen des Lebenszyklus eines Artefakts sowohl in Vorwärts- als auch in Rückwärtsrichtung definiert werden [Dav90, GF94]. Ein (Modellierungs-) Element aus einer Anforderungsbeschreibung sollte dann z.B. von seinem 'Ursprung' (wo kommt diese Anforderung her) bis hin zur Implementierung (wie wurde diese Anforderung erfüllt) verfolgt werden können; und umgekehrt. Nachvollziehbarkeit in beide Richtungen ist notwendig. Hypertext-Techniken stellen eine Möglichkeit dar, Nachvollziehbarkeit in Modellierungswerkzeugen zu unterstützen (siehe auch Abschnitt 4).

Der Ursprung einer Anforderung (requirements pre-traceability) [GF94, Poh96] bezieht sich z.B. auf solche Aspekte, die zur Beschreibung dieser Anforderung geführt haben. Diese *Prä-Nachvollziehbarkeit* ist wichtig und hilfreich, um Änderungen der Anforderungen, die aus dem Anwendungsbereich kommen, vernünftig handhaben zu können. Ein wichtiges Ziel ist, im nachhinein Anforderungen verstehen und die Entstehung von Anforderungen nachvollziehen zu können. Die Prä-Nachvollziehbarkeit kann z.B. die Verantwortlichkeiten für bestimmte Aspekte einer Anforderung liefern [GF95]. Die Grundlagen und Ursachen, die zur Ermittlung der Anforderungen geführt haben, sollten damit erkennbar sein.

Andererseits ist die *Post-Nachvollziehbarkeit* von den Anforderungen hin zur implementierten Anwendung eine wichtige Voraussetzung für eine gute Akzeptanz von Software-Systemen durch die Anwender. Dieser Beitrag konzentriert sich auf Aspekte der Post-Nachvollziehbarkeit.

Nachvollziehbarkeit soll insbesondere unterstützen, Änderungen in einzelnen Modellen — z.B. in den Anforderungen, im Entwurf, im Programm etc. — in anderen Modellen nachvollziehen zu können. Das ist auch dann besonders wichtig, wenn große Softwaresysteme im Team entwickelt werden [PJ94]. Zwischen den verschiedenen Sichten bzw. Sichtweisen [NJJ+96] der unterschiedlichen *Stakeholder* (Personen, die bestimmte Rollen im Entwicklungsprozeß einnehmen; insbesondere auch die Anwender) gibt es i.a. Abhängigkeiten, die möglichst konsistent gehalten werden sollten, auch wenn temporäre Inkonsistenzen zugelassen werden müssen [FGH+93]. Die in einem bestimmten Kontext zu unterstützende Nachvollziehbarkeit hängt auch von den jeweiligen Stakeholdern und der aktuellen Phase im Entwicklungsprozeß ab.

# 3 Eine Fallstudie

Die Grundlage und Motivation für das im folgenden vorgestellte Projekt war ein bestehendes Informationssystem zur Erfassung und Auswertung von Herzkatheter-Untersuchungen. Die Hauptkritikpunkte am alten System waren:

- Die graphische Nutzungsoberfläche war sehr schlecht strukturiert.
- Das Datenmodell war für wissenschaftliche Auswertungen kaum geeignet.

Um die Anforderungen der Anwender besser zu erfüllen, wurde ein besonderer Schwerpunkt auf einen systematischen und nachvollziehbaren Entwicklungsprozeß gelegt, wobei auch die Mitwirkung der Anwender eine wichtige Rolle spielt (partizipative Software-Entwicklung). Der Einsatz von Piktogrammen in der Analyse (Abschnitt 3.1), im Entwurf (Abschnitt 3.2) und in der Implementierung (Abschnitt 3.3) wird im Kontext des vorgestellten Projekts in den folgenden Abschnitten diskutiert.

## 3.1 Analyse des Ist-Zustands und der Anforderungen

Um eine genaue Einschätzung der Eigenschaften des in der Klinik existierenden, alten Systems zu gewinnen, wurde zunächst gemeinsam mit den Anwendern der Ist-Zustand durch Kooperationsdiagramme untersucht (Abschnitt 3.1.2), bevor die Anforderungen an das neu zu entwickelnde System mit Hilfe von weiteren Kooperationsdiagrammen ermittelt wurden (Abschnitt 3.1.3). In Abschnitt 3.1.1 wird zunächst die eingesetzte Technik der Kooperationsdiagramme eingeführt.

### 3.1.1 Kooperationsdiagramme

Kooperationsdiagramme modellieren den Informationsfluß zur Kooperation von Organisationsbereichen und Personen in funktionellen Rollen. Die Technik der Kooperationsdiagramme wurde z.B. in [KWR97] für die Definition von Kernsystem und Ausbaustufen zur Auswahl eines integrierten

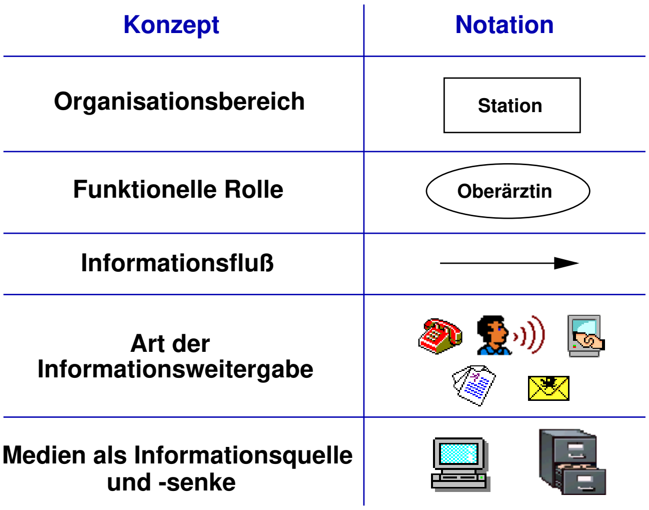


**Abbildung 1:** Vorgegebene Elemente in unseren Kooperationsdiagrammen für Krankenhausinformationssysteme.

Krankenhausinformationssystems eingesetzt (dort *Kooperationsbilder* genannt). Im Rahmen des hier vorgestellten Projekts sind die Kooperationsdiagramme in modifizierter und erweiterter Form für die *Neuentwicklung* eines Informationssystems zum Einsatz gekommen. Ein wichtiges Ziel für die Erweiterungen war, einen auch für die Anwender nachvollziehbaren schrittweisen Übergang von den Ergebnissen der Analyse hin zum entwickelten System zu erreichen. Dazu wurden folgende Änderungen bzw. Erweiterungen gegenüber dem Ansatz aus [KWR97] durchgeführt:

- Unsere vordefinierten Elemente in Kooperationsdiagrammen für Krankenhausinformationssysteme sind in Abbildung 1 dargestellt. In Kooperationsdiagrammen wird der Informationsaustausch zwischen Organisationsbereichen und funktionellen Rollen, die Personen einnehmen können, dargestellt. Die Art der Informationsweitergabe wird an den gerichteten Kanten, die den Informationsfluß darstellen, durch vorgegebene Piktogramme für Telefon, direktes Gespräch, rechnergestütze Kommunikation, Hauspost oder Briefpost dargestellt. So können alle Informationsflüsse im Krankenhaus explizit in der Analyse berücksichtigt werden, um eine isolierte Betrachtung der rechnergestützen Kommunikation zu vermeiden.

  Diese vorgegebenen Piktogramme zur Kennzeichnung der unterschiedlichen Kommunikationsarten werden schrittweise um spezielle Piktogramme aus dem jeweiligen Anwendungsbereich ergänzt, die für die Anwender einen Rückschluß auf die Bedeutung bzw. auf die Art der übermittelten Information zulassen (z.B. als Symbol für Herzkatheterbefunde). So entsteht eine anwendungsspezifische Modellierungsnotation.

- Rollen, die direkt Organisationsbereichen zugeordnet werden können, werden *in* die entsprechenden Rechtecke gezeichnet. In [KWR97] werden solche Rollen *neben* den zugehörigen Organisationsbereichen angeordnet.

- Der Rechner kann in den Kooperationsdiagrammen nicht nur als Medium zur Informationsweitergabe eingesetzt werden (), sondern auch als Informationsquelle und -senke (). Das gleiche gilt für traditionell archivierte Informationen ( ). So kann auch Information, die nicht direkt zwischen Personen oder Organisationseinheiten ausgetauscht wird, explizit berücksichtigt werden. Das gilt insbesondere für Informationen, die inkrementell erhoben und nur bei Bedarf benötigt werden.

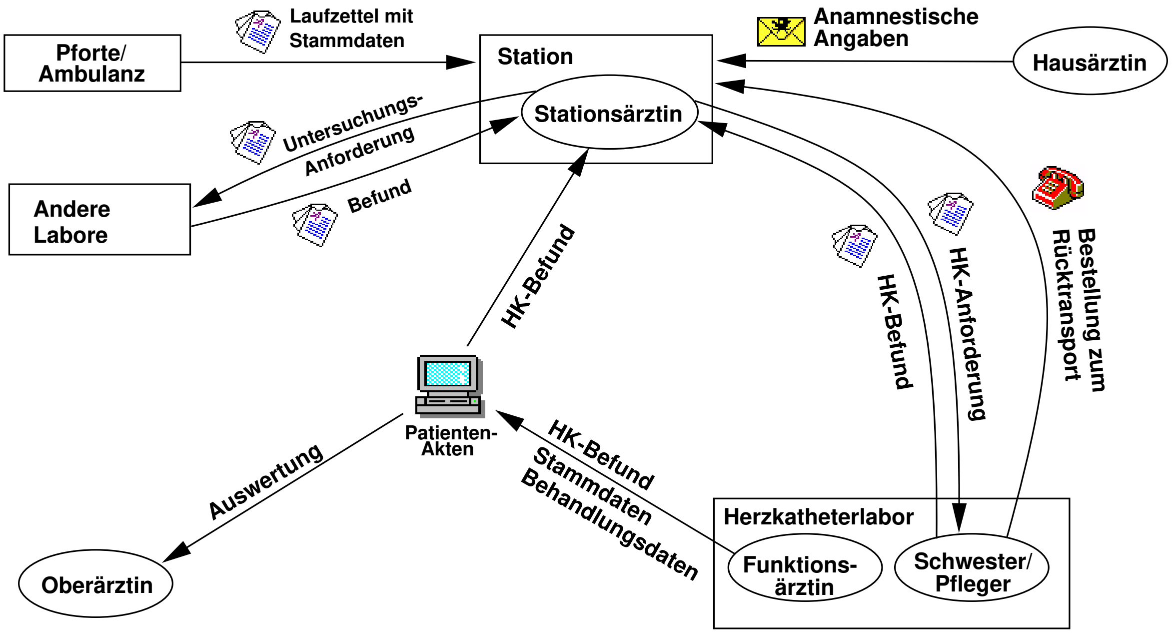


**Abbildung 2:** Ein Ausschnitt aus dem Kooperationsdiagramm zur Ist-Analyse. Rechtecke stellen Organisationsbereiche dar, Ovale stehen für funktionelle Rollen und gerichtete Kanten stehen für die Informationsweitergabe. HK steht für Herzkatheter.

> Um bei komplexeren Modellen die Übersicht zu wahren, ist es möglich, den Rechner als Informationsquelle und -senke logisch auf mehrere Modellierungs-Komponenten aufzuteilen, die dann jeweils für eine Datenmenge zuständig sind. Technisch können mehrere solcher logischen Komponenten in einem System realisiert werden. Umgekehrt kann eine logische Komponente natürlich auch durch mehrere physische Systeme realisiert werden.

Es hat sich bewährt, die Kooperationsdiagramme zunächst an einer großflächigen Tafel (bzw. Flip-Chart) gemeinsam mit den Anwendern zu konstruieren und dann später mit einem Text- und Graphikprogramm zu dokumentieren. Die Entwickler übernehmen bei der Modellierung anfangs nur die Moderation. Die durch die Entwickler dokumentierten Ergebnisse werden später durch die Anwender begutachtet.

Die Modellierung mit Piktogrammen sollte natürlich auch durch geeignete Werkzeuge unterstützt werden. Es ist auch naheliegend, die Nachvollziehbarkeit durch den Einsatz von Hypertext-Techniken zu unterstützen, um eine explizite Sichtbarkeit von Abhängigkeiten erreichen und pflegen zu können. Spezielle Modellierungswerkzeuge, die insbesondere auch die Nachvollziehbarkeit mit Piktogrammen unterstützen sollen, werden in Abschnitt 4 diskutiert.

### 3.1.2 Analyse des Ist-Zustands mit dem alten System

Ein Ausschnitt des Kooperationsdiagramms als Ergebnis der Ist-Analyse ist in Abbildung 2 dargestellt. Die Annotationen an den Kanten geben die Art der Informationsweitergabe durch Piktogramme und den übertragenen Inhalt textuell an. Die Interaktion mit dem Rechner als Informationsquelle und -senke erfolgt implizit rechnerunterstützt, so daß hier die Art der Informationsweitergabe nicht angegeben werden muß. Die direkte rechnerunterstützte Informationsweitergabe wird durch das Piktogramm modelliert. Wie in Abbildung 2 zu erkennen ist, fand bisher keine direkte Kommunikation über den Rechner statt; er diente nur dazu, Behandlungsdaten im Herzkatheterlabor einzugeben und bei Bedarf auf der Station einzusehen oder statistisch auszuwerten.

Zur Erläuterung des Modells: Patienten kommen i.a. mit einem Laufzettel von der Pforte oder aus der Ambulanz auf die Station und werden evtl. im Herzkatheterlabor operiert. Die Herzkatheteranforderungen und -befunde werden durch die Hauspost transportiert, nur der Laufzettel mit der Einweisung wird durch die Patienten selbst mitgebracht.

Der Transport von Patienten aus der Station in die Labore wurde hier nicht explizit modelliert, weil hierbei keine ‘Information’ ausgetauscht wird. Die Anwesenheit eines Patienten ist für die Durchführung von Behandlungen natürlich notwendig. Um die konkreten zeitlichen Abläufe in der Klinik angemessen zu modellieren, müßten diese Aspekte auch berücksichtigt werden. Für eine Unterstützung des Arbeitsflusses durch ein Workflow-Management-System wäre eine Modellierung der Reihenfolge, in der die einzelnen Arbeitschritte ausgeführt werden (sollen), sinnvoll. Für diesen Zweck sind z.B. FUNSOFT-Netze [DG94], die eine abstrakte Petri-Netz-Variante darstellen, besser geeignet als Kooperationsdiagramme. Im vorgestellten Projekt waren die zeitlichen Abläufe für die Behandlungen selbst jedoch fest vorgegeben, so daß auf deren explizite Modellierung verzichtet wurde. Die Daten werden immer dann eingegeben, wenn sie anfallen.

Mit dem existierenden System mußten alle Behandlungsdaten durch den behandelnden Funktionsarzt im Herzkatheterlabor eingegeben werden. Die Daten wurden durch den Oberarzt für Auswertungen und gelegentlich durch den Stationsarzt für Herzkatheterbefunde verwendet. Die Qualität der eingegebenen Daten war nicht sehr hoch, weil die Eingabe mit den schlecht strukturierten Masken für den behandelnden Arzt eine erhebliche Mehrbelastung darstellte. Auch waren die in der Datenbank verwendeten Datenstrukturen kaum für aussagekräftige Auswertungen geeignet, so daß eine Neuentwicklung notwendig wurde.

### 3.1.3 Analyse der Anforderungen an das neu zu entwickelnde System

Auf der Basis der Analyse des Ist-Zustands wurden dann die Anforderungen an das neu zu entwickelnde System gemeinsam mit den Anwendern mit Hilfe von weiteren Kooperationsdiagrammen untersucht. Eine zentrale Frage der Anforderungsanalyse war: **Welche** Daten werden **wo** durch **wen** erhoben und von **wem** benötigt? Ein Ausschnitt des Ergebnisses ist in Abbildung 3 dargestellt, wobei zunächst nur die vordefinierten Elemente für Krankenhausinformationssysteme verwendet werden (siehe Abbildung 1).

Ein wesentlicher Unterschied zum alten System ist die Tatsache, daß die Eingabe der Daten jetzt durch verschiedene Personen bzw. Rollen durchgeführt wird. Die Daten werden im neuen System eingegeben, wenn sie anfallen. Z.B. gibt die Sekretärin die Stammdaten ein, sobald sich die Patienten in der Abteilung melden. Insbesondere die Eingabe, Speicherung und Bearbeitung der medizinischen Daten durch die Ärzte wurde in Zusammenarbeit mit Statistikern der Universität Dortmund optimiert, um die Qualität der geplanten Auswertungen zu erhöhen. Die Strukturierung der Eingabemasken in der späteren Implementierung orientiert sich auch an der Zuordnung der Information zu den einzelnen Piktogrammen (siehe auch Abschnitt 3.3).

In Abbildung 4 ist ein modifiziertes Kooperationsdiagramm zur Anforderungsanalyse dargestellt, in dem an den Kanten die übermittelten, einzugebenden bzw. gelesenen Informationen anschaulich durch weitere Piktogramme dargestellt sind. Bei den neuen Piktogrammen handelt es sich um spezifische Erweiterungen für die Kardiologie des Klinikums Wuppertal, die von den Anwendern selbst mit ausgewählt wurden. Selbstverständlich müssen nicht für alle Modellierungselemente Piktogramme eingeführt werden.

Der Arbeitsfluß wird durch das neue System unterstützt, indem der Stationsarzt Herzkatheteranforderungen () für den Oberarzt zusammenstellt. Der Oberarzt stellt dann aus den Anforderungen den Labor-Plan () zusammen. Aus organisatorischen Gründen werden die Herzkatheteranforderungen und -befunde wie zuvor auch auf Papier zwischen Station und Herzkatheterlabor ausgetauscht.

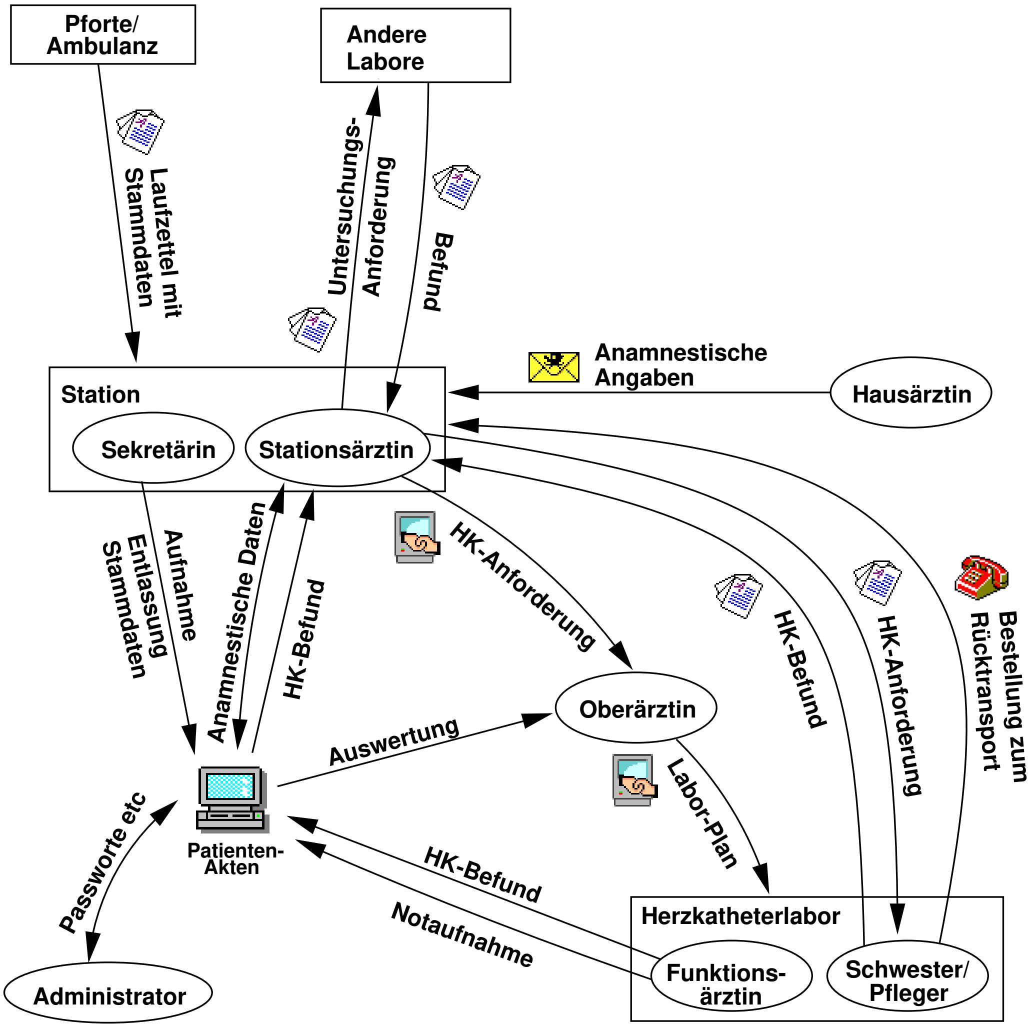


**Abbildung 3:** Ein Ausschnitt des ersten Kooperationsdiagramms zur Anforderungsanalyse, in dem nur die vordefinierten Elemente in Kooperationsdiagrammen für Krankenhausinformationssysteme verwendet werden (siehe Abbildung 1).

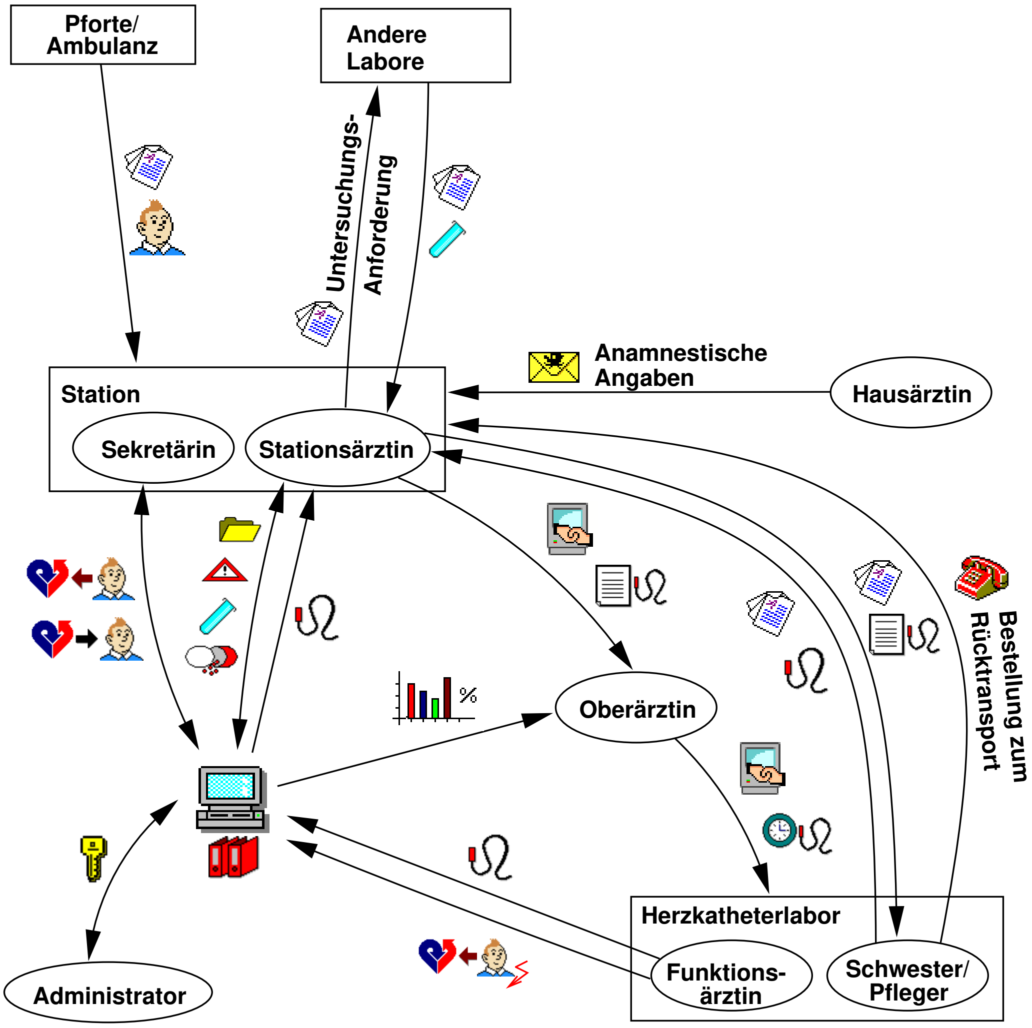


**Abbildung 4:** Das modifizierte Kooperationsdiagramm zur Anforderungsanalyse. An den Kanten sind die übermittelten, einzugebenden bzw. gelesenen Informationen anschaulich durch spezifische Piktogramme dargestellt: steht für die Patientenstammdaten, steht für Herzkatheterbefund, steht für die allgemeinen anamnestischen Daten, steht für das Risikoprofil, steht für die Laborwerte, steht für die Medikation, ist das Logo des Herzzentrums im Klinikum Wuppertal, etc. Diese Symbole stammen aus dem Anwendungsbereich und haben für die Anwender eine klare Bedeutung.

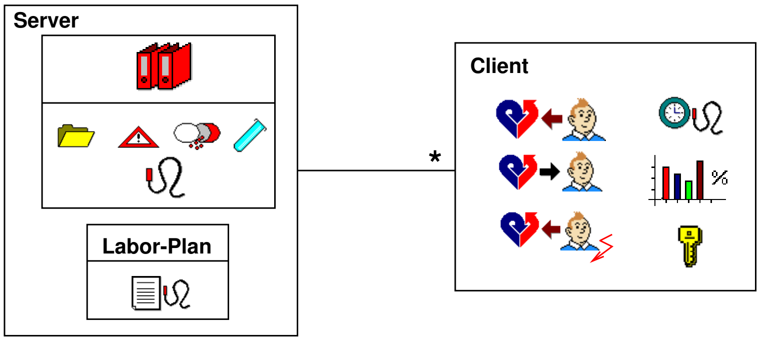


**Abbildung 5:** Die grobe Software-Architektur.

Eine Anmerkung zur eingesetzten Analysetechnik: Im Gegensatz zu einigen Ansätzen zur objektorientierten Modellierung (z.B. [RBP$^+$91]) sind wir nicht der Ansicht, daß sich Klassendiagramme als guter Ausgangspunkt zur gemeinsamen Analyse der Anforderungen mit den Anwendern an ein neues Informationssystem eignen, bevor ein funktionales Modell konstruiert wird: auch wenn die zukünftigen Anwender ein Klassendiagramm (glauben zu) verstehen, ist damit die *Funktionalität* eines neuen Systems i.a. noch nicht ermittelt. Anwender erwarten üblicherweise eine gewisse Funktionalität von einem Informationssystem und nicht eine bestimmte Struktur, in der relevante Daten zu speichern sind. Klassendiagramme eignen sich aber gut zur Problembereichsanalyse (domain analysis) [BHM$^+$96], worauf wir in diesem Beitrag allerdings nicht genauer eingehen.

Trotzdem ist es sehr sinnvoll, Datenmodelle für die Anforderungsanalyse mit den zukünftigen Anwendern zu diskutieren, um zu überprüfen, ob die Modellierer konzeptionelle Fehler gemacht haben (siehe auch Abschnitt 3.2). Anwender verstehen Datenmodelle, in denen auf hohem Niveau Begriffe aus ihrem Arbeitsbereich verwendet werden, wenn sie durch einen Modellierer erklärt werden. Zur Modellierung der Funktionalität sind jedoch andere Modelle notwendig (z.B. Beschreibungen von Arbeitsabläufen oder von Dynamik in einer Anwendung).

### 3.2 Entwurf

Die Wahl einer geeigneten Software-Architektur ist ein zentraler Faktor für die erfolgreiche Erfüllung der Anforderungen durch ein zu entwickelndes System [Boa95, SG96]. Die ermittelten Anforderungen müssen/sollten erfüllt werden. Ein zu lösendes Problem besteht darin, daß sich das 'ideale was' (die Anforderungen) im 'tatsächlichen wie' (der Realisierung) wiederfinden sollte: Die Art der Realisierung beeinflußt die Erfüllung bzw. Erfüllbarkeit der Anforderungen erheblich.

Im vorgestellten Projekt wurde die Software-Architektur als Client/Server-System konzipiert. Die Aufgabenverteilung zwischen Client und Server ist in Abbildung 5 als einfaches Software-Architekturmodell illustriert. Damit ist dann auch grob skizziert, welche Systemkomponenten welche Anforderungen erfüllen sollen. Der Server ist im wesentlichen für die Datenspeicherung zuständig, während auf den Clients die verschiedenen Operationen angestoßen werden (Änderungen in der Datenbank werden natürlich durch den Server geregelt). Der Labor-Plan ordnet die Herzkatheter-Anforderungen. Das Piktogramm symbolisiert hier das Zusammenstellen des Labor-Plans. Der * an der Kante spezifiziert, daß ein Server mit mehreren Clienten verbunden sein kann.

Entsprechend dem '4+1'-Schichtenmodell für Software-Architekturen [Kru95] ist das Modell in Abbildung 5 der *Prozeßsicht* zuzuordnen, in der auch nicht-funktionale Anforderungen, wie z.B. die Verteilung von Daten und Funktionen, modelliert werden. Die Prozeßsicht basiert auf der logischen Sicht, in der die angeforderte Funktionalität der Anwender modelliert wird (entspricht bei uns den Kooperationsdiagrammen).

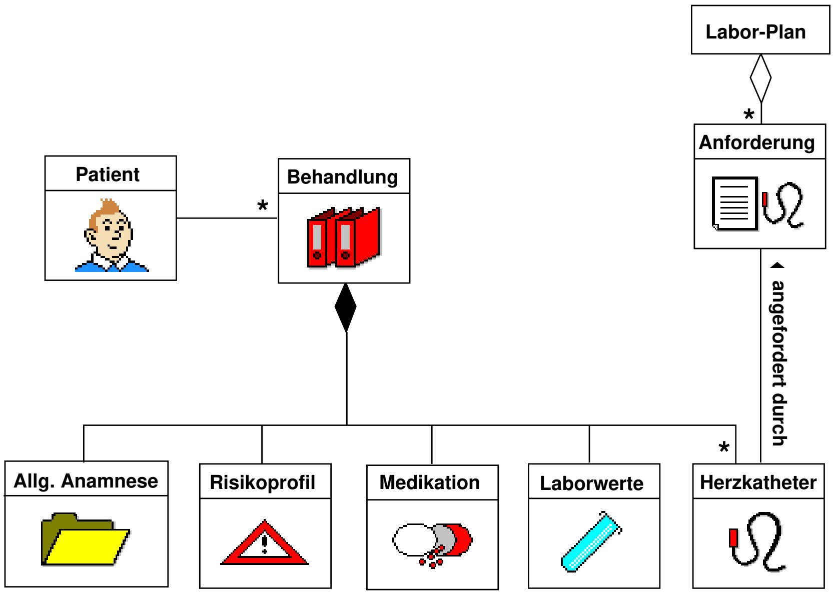


**Abbildung 6:** Ein Ausschnitt des Datenmodells für die Patientenakten.

Die Konsistenzwahrung verschiedener Sichten einer Software-Architekturbeschreibung ist ein zu lösendes Problem [Sha95]. Piktogramme bieten hier einen Ansatzpunkt zur Integration verschiedener Sichten einer Architekturbeschreibung, indem Verbindungen zwischen den Vorkommen eines Piktogramms in verschiedenen Modellen hergestellt werden (siehe auch Abschnitt 4).

In Abbildung 6 ist ein Ausschnitt des Datenmodells für die Patientenakten dargestellt. Einem Patienten können mehrere Behandlungen zugeordnet werden, wobei jede Behandlungsakte mehrere (angeforderte) Herzkatheter-Untersuchungen enthalten kann. Die anamnestischen Daten (Krankengeschichte) werden je Behandlung genau einmal erfaßt.

Zur Modellierung des Datenmodells wird die UML-Notation für Klassendiagramme verwendet [RSC97]. Der * an einer Assoziation spezifiziert die Kardinalität $\geq 0$ (die Grundeinstellung ist die Kardinalität $1$). Der kleine Pfeil an Assoziationsnamen deutet die Leserichtung an. Die gefüllte Raute modelliert *ungeteilte* Aggregation (Komposition): die aggregierten Teile gehören zu genau einem Aggregat, wobei die Existenz der Teile an die Existenz des Aggregats gebunden ist [RSC97]. Teile mit der Kardinalität $1$ dürfen nicht entfernt werden. UML erlaubt auch die Spezifikation der *geteilten* Aggregation, wobei Teile zu mehreren Aggregaten gehören dürfen und auch ohne enthaltende Aggregate existieren können [RSC97]. Herzkatheter-Anforderungen können auch existieren, ohne einem Labor-Plan zugeordnet zu sein.

Auch im Datenmodell werden einige der in den Kooperationsdiagrammen verwendeten Piktogramme benutzt. So erhalten wir einen systematischen und nachvollziehbaren Weg von der funktionsorientierten Darstellung der Kooperationsbilder hin zur objektorientierten Modellierung. Das Datenmodell wurde mit den zukünftigen Anwendern diskutiert, um zu überprüfen, ob bei der Modellierung konzeptionelle Fehler gemacht wurden.

In Abbildung 7 ist die Dynamik der Benutzungsschnittstelle als Statechart [Har87] modelliert, wobei auch hier teilweise Piktogramme zur Beschreibung der Zustände und der Zustandsübergänge eingesetzt werden. Die Navigation durch die Patientenakte wurde hierarchisch durch einen geschachtelten Zustand modelliert.

Im vorgestellten Projekt wurde nur der Informationsfluß innerhalb einer Abteilung genauer untersucht. Für eine angemessene Unterstützung des Informationsflusses in der gesamten Klinik wäre

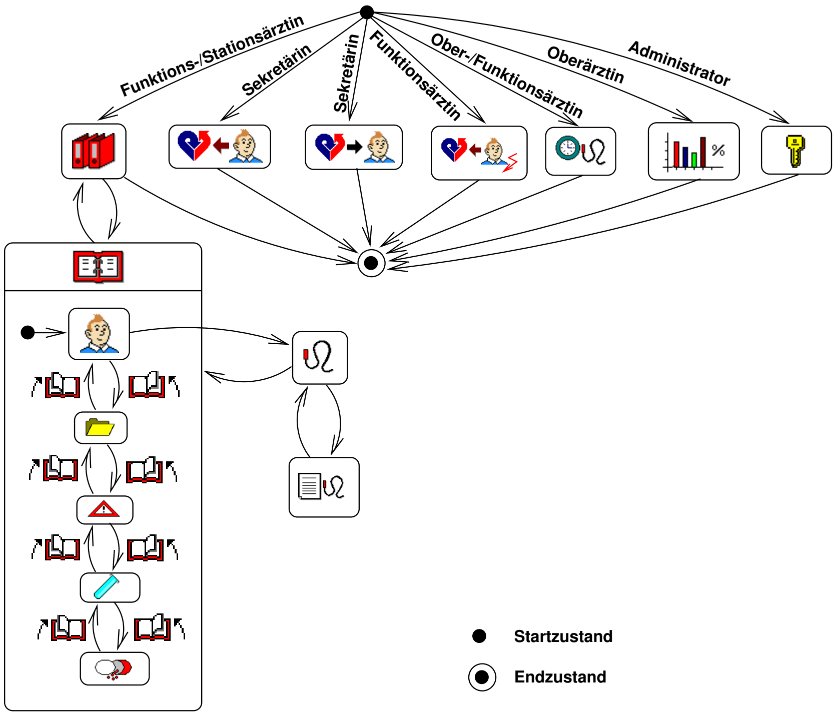


**Abbildung 7:** Die Dynamik der Benutzungsschnittstelle modelliert als Statechart. Teilweise werden Piktogramme zur Beschreibung der Zustände und der Zustandsübergänge eingesetzt.

es notwendig, auch den Informationsaustausch zwischen verschiedenen Abteilungen zu betrachten. In Abbildung 4 auf Seite 7 würde sich dann z.B. die Anforderung ergeben, die Laborwerte () aus anderen Laboren direkt elektronisch zu erhalten, statt sie auf der kardiologischen Abteilung von den erhaltenen Papierunterlagen nochmals zu erfassen. Das gleiche gilt für die Stammdaten, die bereits in die Rechner der Verwaltung eingegeben wurden (sofern kein Notfall vorliegt).

Hier ergeben sich die typischen Probleme durch die Existenz isolierter Abteilungssysteme: Mehrfache Eingabe der gleichen Daten mit redundanter Speicherung, die potentielle Inkonsistenzen verursacht. Ein wichtiges Ziel muß es sein, die verteilten, heterogenen Informationssysteme der einzelnen Abteilungen zu integrieren, um die Anforderungen angemessen zu erfüllen. Dabei müssen auch die Besonderheiten der Integration in Krankenhausinformationssystemen berücksichtigt werden: In Krankenhausinformationssystemen ist es nicht das primäre Ziel, einen gleichförmigen Zugriff auf 'alles' zu erreichen, wie es z.B. häufig beim Zugriff auf Datenbanken per WWW erreicht werden soll. Vielmehr gibt es verschiedene Sichten auf Teilbereiche der Information im Krankenhaus, wobei replizierte Information konsistent gehalten werden muß. Für eine genauere Diskussion dieser Aspekte sei auf [Has97] verwiesen. Für eine angemessene Erfüllung der Anforderungen sind grundsätzlich zwei Aspekte zu beachten, wobei sich der vorliegende Beitrag auf den ersten konzentriert:

1. Ist es nachvollziehbar, daß wir das *richtige* System bauen?
2. Bauen wir das System so, daß die Anforderungen durch die Realisierung des Systems angemessen erfüllt werden?

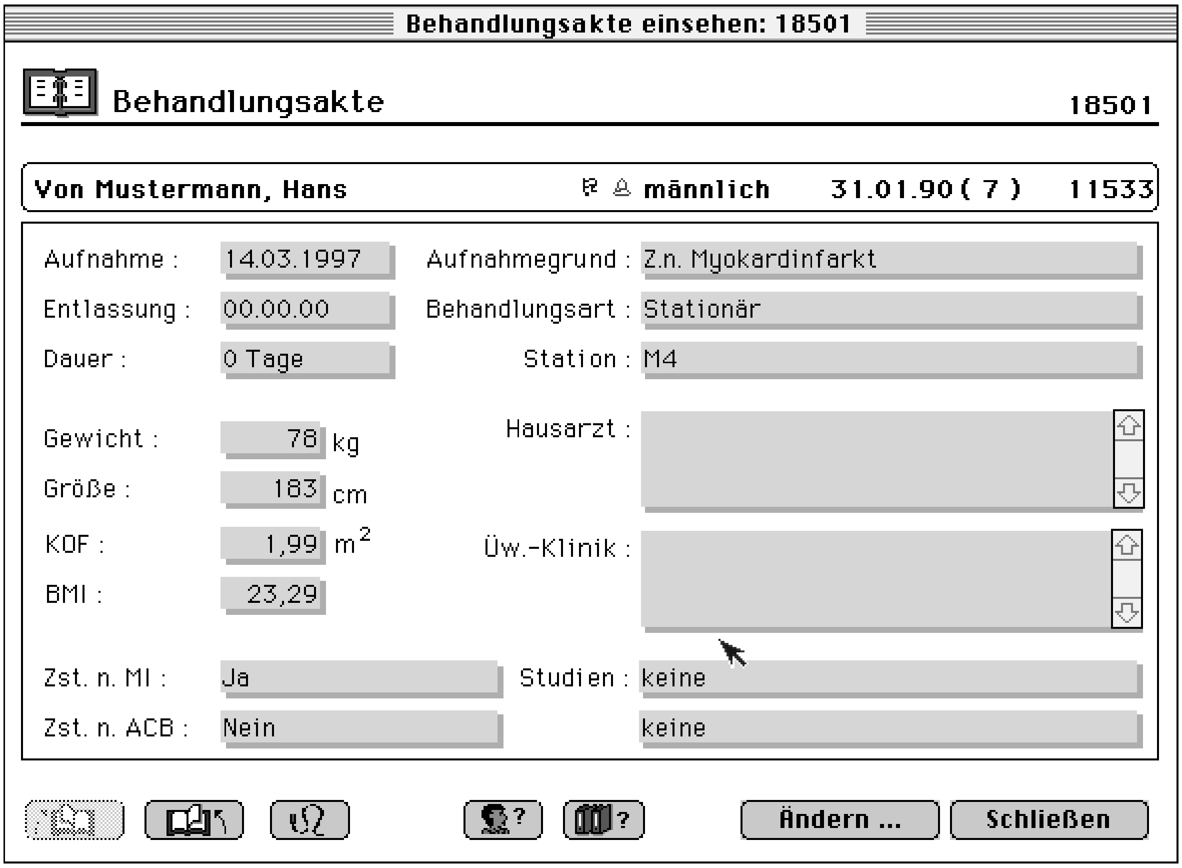


**Abbildung 8:** Eine Eingabemaske der entwickelten Anwendung: Einstieg in die Behandlungsakte. Die Piktogramme dienen zur Kennzeichnung und zur Navigation.

### 3.3 Implementierung des Informationssystems

Eine frühzeitige Überprüfung der Ergebnisse der Anforderungsanalyse durch Ausführung und Bewertung von Prototypen ist sinnvoll, um eine inkrementelle und evolutionäre Entwicklung zu ermöglichen [BKKZ92]. Für das Prototyping müssen natürlich bestimmte technische und organisatorische Voraussetzungen erfüllt sein, was im vorgestellten Projekt der Fall war. Wichtig ist, daß beim Prototyping nicht nur Oberflächenmasken (sogenannte Mock-Up-Prototypen) zur Verfügung stehen, sondern daß auch ein gewisses Maß an Funktionalität vorhanden ist, damit die Arbeitsprozesse am Prototypen nachvollzogen werden können.

Im vorgestellten Projekt wurden den Anwendern frühzeitig Prototypen zur Unterstüzung der einzelnen Arbeitsschritte zur Verfügung gestellt, um die Ergebnisse der Anforderungsanalyse durch die Kooperationsdiagramme zu überprüfen und zu verfeinern. Zur systematischen Bewertung der Prototypen wurde für jede Maske ein strukturierter Fragebogen entworfen, der zusammen mit den Anwendern ausgewertet wurde.

Abbildung 8 zeigt eine Eingabemaske der Benutzungsschnittstelle des implementierten Systems: der Einstieg in die Behandlungsakte. Auf den Eingabemasken der Benutzungsschnittstelle wurden die zugehörigen Piktogramme zur einfachen Wiedererkennung dargestellt. Die Piktogramme dienen in der gesamten Anwendung zur Kennzeichnung und zur Navigation zwischen den einzelnen Masken der Benutzungsschnittstelle. Zur Identifikation der Maske in Abbildung 8 dient das Piktogramm (geöffnete Akte) und zur Navigation werden auf den Knöpfen der unteren Leiste einige Piktogramme aus den Kooperationsdiagrammen der Anforderungsanalyse in Abbildung 4, der Software-Architektur in Abbildung 5, dem Datenmodell in Abbildung 6 bzw. der Beschreibung der Dynamik in Abbildung 7 verwendet.

Ein weiterer Aspekt für die Überprüfung der Ergebnisse der Anforderungsanalyse ist eine frühzeitige Simulation von Modellen. Für Kooperationsdiagramme ist das jedoch kaum möglich, da hier noch kein Kontrollfluß explizit modelliert wird. Zustandsautomaten (Statecharts) können durch Simulation ausgeführt werden [HG97] (das gilt z.B. auch für Petri-Netze); dieses konnten wir allerdings nicht einsetzen, weil uns auf der Zielplattform keine entsprechenden Werkzeuge zur Verfügung standen.

Die Implementierung erfolgte auf einem Netzwerk von Apple Macintosh Rechnern auf Basis der Client/Server-Version der Datenbank 4D [ACI96]. Die Strukturierung der Daten basiert auf dem Datenmodell aus Abbildung 6 und wurde in Zusammenarbeit mit Statistikern der Universität Dortmund für die spätere statistische Auswertung hin optimiert. Für ad-hoc Auswertungen wurden einfache statistische Verfahren direkt im Informationssystem implementiert (Tortendiagramme, verschiedene Mittelwertberechnungen, etc). Für komplexe Auswertungen steht eine Export-Schnittstelle zum Statistiksystem SAS [DJS92] zur Verfügung. Aus dem alten System konnten ca. 90% der Behandlungsdaten übernommen werden. Diese Altdaten wurden mit einem entsprechenden Vermerk versehen, um später differenzierte Auswertungen zu ermöglichen.

Auch im Handbuch werden die Piktogramme verwendet, um für die Anwender einen einfachen Wiedererkennungseffekt zu erreichen. Das Kooperationsdiagramm aus der Anforderungsanalyse (Abbildung 4) und das Statechart für die Spezifikation der Dynamik in der Benutzungsschnittstelle (Abbildung 7) dienen hier als überblick für die Funktionalität der Applikation.

## 4 Werkzeugunterstützung für die Modellierung mit Piktogrammen

Die Modellierung mit Piktogrammen sollte auch durch geeignete Werkzeuge unterstützt werden. Es existieren verschiedene Werkzeuge, die speziell die Anforderungsanalyse unterstützen sollen. Beispielsweise verknüpft ARTS [DF84] Anforderungsbeschreibungen in einer hierarchischen Struktur. Es ist auch naheliegend, die Nachvollziehbarkeit durch den Einsatz von Hypertext-Techniken [Big88, GS90, CR92] zu unterstützen, um eine explizite Sichtbarkeit von Abhängigkeiten erreichen und pflegen zu können. Die Links sollten geeignet in einem CASE-Repository gespeichert werden [Big88]. Beispiele für Werkzeuge zur Anforderungsanalyse, die auch Hypertext unterstützen sind IBIS [CB88], PRO-ART [Poh96], RETH [Kai93] und TOOR [PG96]. Das CASE-Werkzeug Objectory unterstützt insbesondere die Nachvollziehbarkeit von der Anforderungsanalyse zum Entwurf [LS96].

Es reicht hier allerdings nicht aus, einfache Hypertext-Funktionalität, wie sie z.B. HTML bietet, einzusetzen. Ein *Link* verbindet einen Ausgangsanker mit einem Zielanker und erlaubt vom Ausgangsanker zum Zielanker zu gelangen (z.B. durch Anklicken des Ausgangsankers). Falls wir nun die Piktogramme als Anker verwenden möchten, wird es i.a. jedoch mehrere Zielanker in möglicherweise verschiedenen Modellen für einen Ausgangsanker geben. Abbildung 9 zeigt, wie ein Werkzeug damit umgehen kann: Nach Anklicken eines Ankers wird bei mehreren möglichen Zielen zunächst eine hierarchische Auswahlliste (Kontextmenü) geöffnet, die es erlaubt, das gewünschte Ziel zu erreichen. Das Kontextmenü sollte insofern kontextsensisitiv sein, als nur solche Ziele ausgewählt werden können, in denen das entsprechende Element auch vorkommt. Im Prinzip erhalten wir so *typisierte* Links [Big88]. Der jeweilige Link-Typ ergibt sich auch durch den Kontext des Ausgangs- und Zielankers.

Prä-Nachvollziehbarkeit (siehe Abschnitt 2) kann z.B. durch spezielle Arten von Links unterstützt werden. In Abbildung 9 kann dazu der Link auf 'Verantwortlichkeiten' genutzt werden, der evtl. noch hierarchisch untergliedert sein kann, um eine detaillierte Unterscheidung von Verantwortlichkeiten,

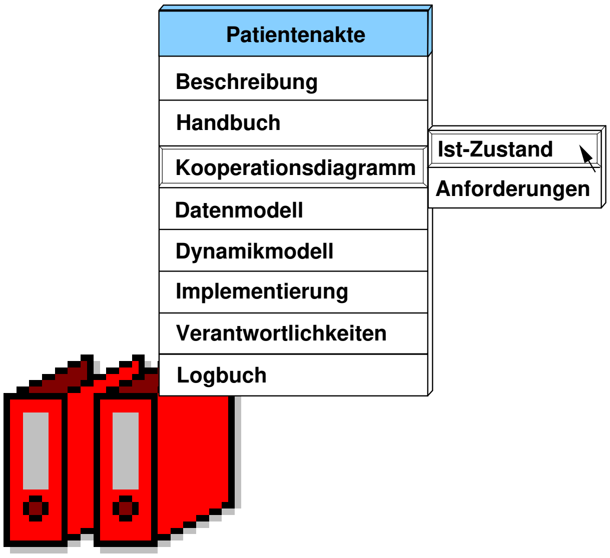


**Abbildung 9:** Ein kleines Szenario für die hypertext-basierte Werkzeugunterstützung: ein Piktogramm wurde als Modellierungselement in einer Spezifikation ausgewählt, um zu anderen Spezifikationen zu gelangen, in denen dieses Piktogramm ebenfalls eine Rolle spielt. Statt direkt zu einem Ziel gehen zu können, wird eine (hierachische) Auswahlliste geöffnet, aus der ein Link gewählt werden kann. Im Szenario wird für das Patientenaktensymbol das Kooperationsdiagramm für den Ist-Zustand ausgewählt.

wie sie z.B. in [GF95] vorgeschlagen werden, zu ermöglichen. Zusätzlich zu einer derartigen Werkzeugunterstützung sind natürlich auch geeignete organisatorische Mechanismen zur systematischen Protokollierung dieser Informationen nötig.

Abbildung 10 illustriert Prä- und Post-Nachvollziehbarkeit im Zusammenhang mit den involvierten Stakeholdern und den produzierten Spezifikationen bzw. der implementierten Anwendung als Endprodukt. Piktogramme unterstützen hier insbesondere die Nachvollziehbarkeit von den Anforderungen zum realisierten System. Z.B. wird das Symbol für die Patientenakte () in allen Phasen der Software-Entwicklung verwendet. Entwicklungswerkzeuge sollten durch geeignete Hypertext-Funktionalität die Nachvollziehbarkeit zwischen den verschiedenen Modellen und damit zwischen den verschiedenen Phasen im Entwicklungsprozeß unterstützen.

Abbildung 11 illustriert den Unterschied zwischen *historischer* und *life-cycle* Nachvollziehbarkeit. Historische Nachvollziehbarkeit kann durch Versions- und Konfigurationsmanagement unterstützt werden, wobei das Logbuch dann eine wichtige Rolle spielt. Das *Logbuch* (siehe Abbildung 9) verwaltet im Sinne einer Versionsverwaltung eine Historie aller Änderungen, die dieses Piktogramm betreffen. Das ist auch deshalb sinnvoll, weil diese Information i.a. automatisch protokolliert werden kann. ESE [RUPT90] ist z.B. ein Werkzeug, das historische und life-cycle Nachvollziehbarkeit u.a. durch Konfigurationsmanagement unterstützt.

In [Cor96] wird die historische Nachvollziehbarkeit als *horizontale* Nachvollziehbarkeit und die life-cycle Nachvollziehbarkeit als vertikale Nachvollziehbarkeit bezeichnet. Andererseits verwenden Lindvall und Sandahl [LS96] den Begriff *horizontale Nachvollziehbarkeit* für life-cycle Nachvollziehbarkeit und vertikale Nachvollziehbarkeit für Nachvollziehbarkeits-Links innerhalb eines Modells (nicht zu anderen Versionen eines Elements, sondern zu anderen Elementen). In unserer Terminologie umfaßt die life-cycle Nachvollziehbarkeit Prä- und Post-Nachvollziehbarkeit (siehe Abbildung 11). Die Nachvollziehbarkeit innerhalb eines Modells würden wir als *Intra-Modell-Nachvoll-*

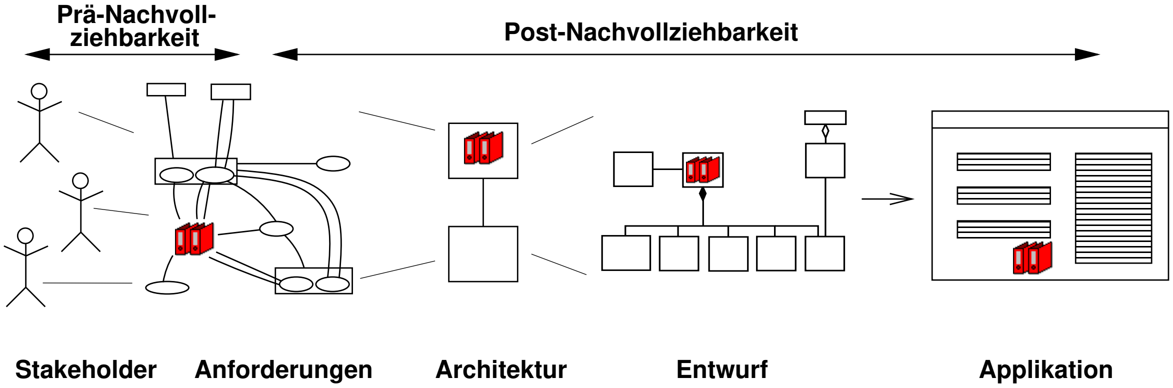


**Abbildung 10:** Prä- und Post-Nachvollziehbarkeit.

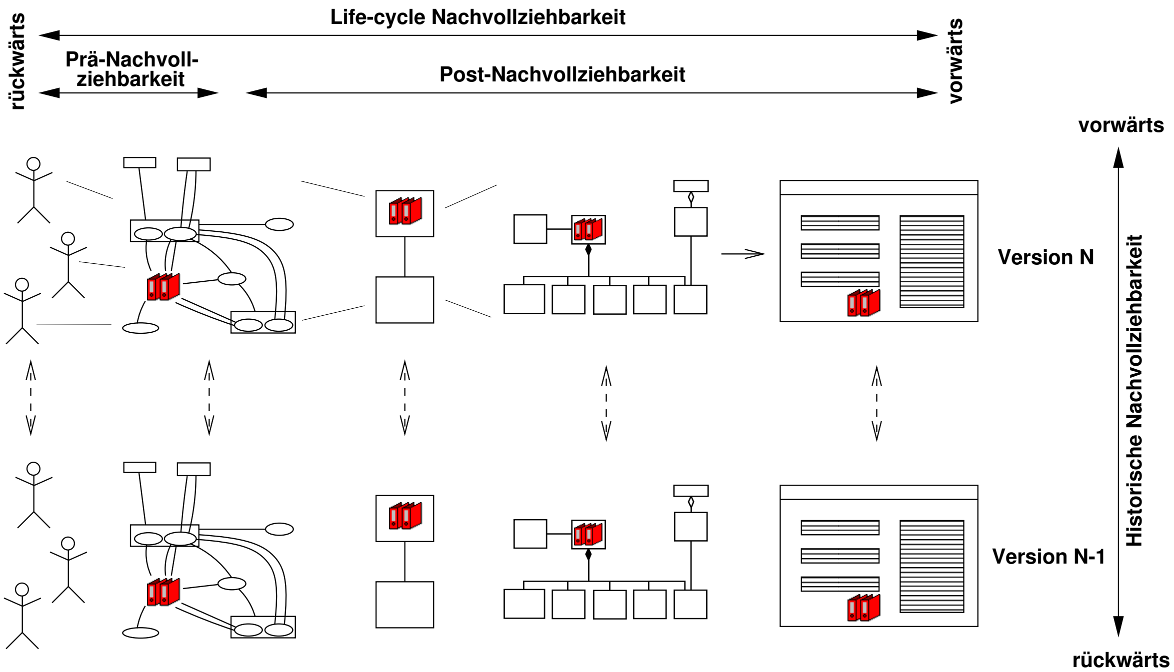


**Abbildung 11:** Historische und life-cycle Nachvollziehbarkeit.

*ziehbarkeit* bezeichnen. Intra-Modell-Nachvollziehbarkeit ist in Abbildung 11 nicht illustriert. Offensichtlich hängt die Bedeutung der Begriffe *vertikale* und *horizontale* Nachvollziehbarkeit von der Art ab, wie diese Zusammenhänge graphisch dargestellt werden, so daß wir diese Begriffe nicht für besonders geeignet ansehen, um die entsprechenden Zusammenhänge zu benennen.

Die Nachvollziehbarkeits-Links sollten weitgehend automatisch erzeugt werden können, weil manuelle Verknüpfungen i.a. sehr subjektiv und individuell gestaltet werden. Zum einen kann eine integrierte Versionsverwaltung hierzu genutzt werden, andererseits ist es auch denkbar alle Aktionen, die während der Modellierung stattfinden, zu speichern, wie es z.B. in [Poh96] vorgeschlagen wird. Zum Nachvollziehen des Entwicklungsprozesses ist dann eine selektive Auswahl der Nachvollziehbarkeits-Links erforderlich.

Die Menge der Nachvollziehbarkeits-Links sollte aber nicht zu groß werden. Dazu wird in [PDJ97] z.B. vorgeschlagen, durch einen Projektleiter die zu speichernde Nachvollziehbarkeits-Information in Bezug auf die zu erwartende Nutzung projekt-spezifisch und prozeß-orientiert zu definieren. Es wird auch definiert, welche Information automatisch bzw. nach vorheriger Interaktion mit dem Modellierer gespeichert werden soll.

Auch eine dynamische Generierung von Nachvollziehbarkeits-Links basierend auf einer Architekturbeschreibung erscheint sinnvoll [TN97]. Architekturbeschreibungen dienen insbesondere zur Überbrückung der Kluft zwischen der Beschreibung der Anforderungen und dem Entwurf eines Systems (siehe auch Abbildung 10). Architekturbeschreibungen sind i.a. nicht so detailliert wie Entwürfe und enthalten auch nicht alle Details einer Anforderungsbeschreibung, so daß die Menge der automatisch generierten dynamischen Nachvollziehbarkeits-Links nicht zu groß wird. Mit unserem Ansatz können gleiche Piktogramme in verschiedenen Modellen als Anker für zu generierende Links genutzt werden.

Möglichkeiten zur dynamischen Erzeugung, Änderung und Entfernung von Knoten und Piktogrammen während der Entwicklung sind erforderlich. Z.B. sollten nach dem Löschen eines Knotens alle Links, für die dieser Knoten ein Anker ist, mitgelöscht werden (referentielle Integrität). Wir entwickeln z.Z. Werkzeuge mit der objektorientierten Datenbank $O_2$ [BDK92] als Repository. Durch den Einsatz der objektorientierten Datenbank kann die referentielle Integrität im Repository weitgehend automatisch gewährleistet werden: durch das Löschen eines Knotens werden automatisch alle in Beziehung stehenden Links gelöscht. Auch bei Änderungen von Links wird die referentielle Integrität gewahrt.

Piktogramme werden nicht als Wert-Attribute von Objekten, die entsprechende Modellierungselemente im Repository repräsentieren, gespeichert. Statt dessen erhalten die Datenbankobjekte, die Modellierungselemente repräsentieren, Zeiger auf Objekte, die dann die Hypertext-Links verwalten. Das ist auch hilfreich bei einer dynamischen Generierung der Nachvollziehbarkeits-Links, wobei eine bidirektionale Verknüpfung sinnvoll ist. Abbildung 12 illustriert diesen Mechanismus. In diesen speziellen *Nachvollziehbarkeits-Objekten* innerhalb des Repository müssen nicht ausschließlich Piktogramme verwaltet werden. Auch rein textuelle Modellierungselemente könnten so als Nachvollziehbarkeits-Objekte verwaltet werden, falls zu Elementen, die für die Nachvollziehbarkeit wichtig sind, (noch) keine geeigneten Piktogramme zugewiesen wurden.

In Abbildung 4 auf Seite 7 werden an den Kanten im Kooperationsdiagramm die übermittelten, einzugebenden bzw. gelesenen Informationen anschaulich durch Piktogramme dargestellt, bei denen es sich um neue Piktogramme als spezifische Erweiterungen für die Kardiologie des Klinikums in Wuppertal handelt. Ein CASE-Werkzeug sollte also auch unterstützen, eigene Piktogramme in die Modellierung einzubinden bzw. neue Piktogramme zu kreieren. Mit $O_2$ können die Piktogramme einfach als Bitmap verarbeitet werden. Durch eine vordefinierte Methode kann prinzipiell jedes Piktogramm direkt mit einem Editor der Datenbank bearbeitet werden und so können auch neue Piktogramme kreiert werden. Auch ein Import von Piktogrammen, die extern mit anderen Programmen erzeugt wurden, soll möglich sein. Eine geeignete graphische Präsentation der Versionsbäume, Ver-

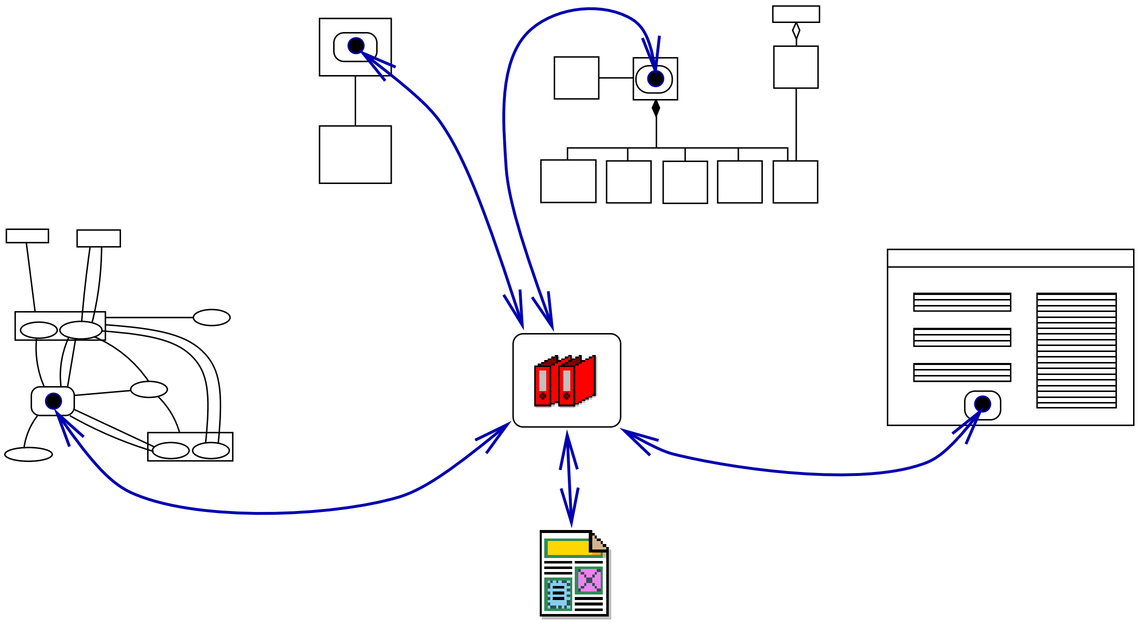

**Abbildung 12:** Nachvollziehbarkeits-Objekte: Datenbankobjekte, die ein Element in einem Modell repräsentieren, erhalten Zeiger auf Nachvollziehbarkeits-Objekte, die die Hypertext-Links verwalten. Idealerweise sollten hier auch Verbindungen zur Dokumentation und zur Implementierung verwaltet werden können.

antwortlichkeitsstrukturen (evtl. Organigramme), aber auch der life-cycle Nachvollziehbarkeits-Links ist sinnvoll. Falls ein Modellierungselement innerhalb eines Modells mehrfach vorkommt, muß dieser Sachverhalt entsprechend in der hierachischen Auswahlliste berücksichtigt werden.

Wie in [Whi94] diskutiert wird, sollten Entwickler über Änderungen durch andere Entwickler/Modellierer in Modellen, die sie betreffen, (aktiv) informiert werden. Im Kontext unseres Ansatzes bietet es sich hier an, Modellierern eine Art 'Abonnement' für Änderungen, die bestimmte Piktogramme oder andere Modellierungselemente betreffen, zu ermöglichen. Solch ein Abonnement würde sich dann auf alle Vorkommen dieser Modellierungselemente (Piktogramme) in verschiedenen Modellen beziehen. Diese Vorkommen können als eine Äquivalenzklasse für ein Abonnement angesehen werden. Technisch können die Abonnenten in den Nachvollziehbarkeits-Objekten vermerkt werden und ein Benachrichtigungsmechanismus, wie ihn z.B. das CASE-Repository H-PCTE bietet [PK96], könnte zur effizienten Aktivierung genutzt werden. Daher erwägen wir auch eine H-PCTE-basierte Realisierung der Werkzeuge.

# 5 Zusammenfassung

Dieses Papier berichtet über die Erfahrungen mit dem Einsatz von Piktogrammen bei der Neuentwicklung eines medizinischen Informationssystems in einem interdisziplinären Kooperationsprojekt zwischen Informatikern, Statistikern und Medizinern. Piktogramme unterstützen dabei die Nachvollziehbarkeit zwischen

- Kooperationsdiagrammen zur Ist- und Anforderungsanalyse,
- Software-Architekturbeschreibung, Daten- und Dynamikmodellen im Entwurf,
- Benutzungsoberflächen in der Implementierung und dem

- Handbuch.

Insbesondere erhalten wir für die Anwender auch eine Nachvollziehbarkeit von der Anforderungsanalyse hin zum entwickelten System. Das hat einen erheblichen Einfluß auf die Akzeptanz eines neuen Software-Systems. Die frühzeitige Überprüfung der Ergebnisse der Anforderungsanalyse durch Ausführung und Bewertung von Prototypen durch strukturierte Fragebögen hat sich ebenfalls sehr positiv auf die Akzeptanz ausgewirkt.

Die Kooperationsdiagramme haben es uns auf sehr einfache Art erlaubt, zusammen mit den Anwendern die Anforderungen an das neue Informationssystem auf der Ebene des Informationsflusses in der Klinik zu analysieren. Die Kooperationsdiagramme boten die Grundlage für die Systemarchitektur, die Strukturierung der Daten in der Datenbank und für die Gestaltung der Benutzungsschnittstellen.

Für die partizipative Ermittlung von Anforderungen an Krankenhausinformationssysteme hat sich die Kombination von Fragebögen, Interviews und Prototyping bewährt, was z.B. auch durch die Studie, über die in [SFR+92] berichtet wird, bestätigt wurde. In [CdB93] wurden mehrere Studien zur partizipativen Software-Entwicklung ausgewertet. Eine wichtige Erkenntnis ist, daß das organisatorische Umfeld eine interdisziplinäre Zusammenarbeit erlauben muß und daß Prototyping eine geeignete Technik zur Einbindung der Anwender darstellt. Diese Erkenntnisse haben sich im hier vorgestellten Projekt bestätigt. Zusätzlich konnten wir die Partizipation der Anwender durch den Einsatz von anwendungsorientierten Piktogrammen in den verschiedenen Phasen der Software-Entwicklung unterstützen.

Anwendungsorientierte Piktogramme als *Anker* der Nachvollziehbarkeits-Links (und damit als Ansatzpunkte für die Nachvollziehbarkeit) sind zur Unterstützung der Nachvollziehbarkeit besser geeignet als die ausschließliche Verwendung rein textueller Begriffe. Eine hypertext-basierte Werkzeugunterstützung, wie sie in Ansätzen diskutiert wurde, ist naheliegend. Ein Piktogramm repräsentiert eine graphische Metapher, die sich mit reinen Texten (bzw. Akronymen) so nicht erreichen läßt [Git86]. Ein Modellierungwerkzeug sollte trotzdem das Umschalten (bzw. die gemeinsame Darstellung) zwischen dem Namen und dem Symbol eines Piktogramms sowohl im Gesamtmodell als auch für ausgewählte Bereiche erlauben. Eine empirische Studie, über die in [GTK89] berichtet wird, zeigte, daß durch die (Möglichkeit zur) Kombination von verschiedenen Modalitäten (hier Piktogramm und Text) eine optimale Informationsvermittlung erreicht wird. Diese Studie hat auch ergeben, daß es nicht ausreicht, sich auf vorgegebene (standardisierte) Symbole zu beschränken. Trotzdem ist es natürlich nicht sinnvoll, für *jedes* Modellierungselement ein eigenes Piktogramm einzuführen. In den in Abschnitt 3 präsentierten Modellausschnitten wurden teilweise relativ viele Piktogramme verwendet, um die Ideen zu illustrieren.

Das neu entwickelte Informationssystem befindet sich seit Februar 1997 im Einsatz und ist, auch aufgrund der partizipativen Entwicklung, bei den Anwendern auf sehr gute Akzeptanz gestoßen. Die vorgestellten Analysetechniken werden zur Zeit im größeren Rahmen auch in anderen Bereichen der Klinik eingesetzt, um weitere Erfahrungen damit zu sammeln.

## Danksagung

An dieser Stelle sei Oliver Alsbach, Andreas Christmann und Klaus Emmerich für die sehr gute Zusammenarbeit in diesem interdisziplinären Projekt gedankt.

## Literatur

[ACI96] ACI GmbH. *4D Dokumentation*, 1996.

[BDK92] F. Bancilhon, C. Delobel, und P. Kanellakis. *Building an Object-Oriented Database System: The Story of $O_2$*. Morgan Kaufman, 1992.

[BHM+96] J. Bodemann, W. Hasselbring, D. Mehlstäubler, T. Jahnke, und A. Röser. Eine objektorientierte Problembereichsanalyse für die elektronische Patientenakte. In *Abstracts zur 41. GMDS-Jahrestagung*, Seite 127, Bonn, September 1996.

[Big88] J. Bigelow. Hypertext and CASE. *IEEE Software*, Seiten 23–27, März 1988.

[BKKZ92] R. Budde, K. Kautz, K. Kuhlenkamp, und H. Züllighoven. *Prototyping — An Approach to Evolutionary System Development*. Springer-Verlag, 1992.

[Boa95] M. Boasson. The Artistry of Software Architecture. *IEEE Software*, 12(6):13–16, November 1995.

[Bra97] R. Brami. Icons: a unique form of painting. *ACM interactions*, 4(5):15–28, September/Oktober 1997.

[CB88] J. Conklin und M. Begeman. IBIS: A hypertext tool for exploratory policy discussion. *ACM Transactions on Office Information Systems*, Seiten 303–331, April 1988.

[CdB93] A. Clement und P. Van den Besselaar. A retrospective look at PD projects. *Communications of the ACM*, 36(4):29–37, Juni 1993.

[Cor96] J.-P. Corriveau. Traceability Process for Large OO Projects. *IEEE Computer*, 29(9):63–68, September 1996.

[CR92] J.L. Cybulski und C. Reed. A Hypertext Based Software Engineering Environment. *IEEE Software*, Seiten 62–68, März 1992.

[Dav90] A.M. Davis. *Software requirements analysis and specification*. Prentice Hall, Englewood Cliffs, NJ, 1990.

[DF84] M. Dorfman und R. Flynn. ARTS – An automated requirements traceability system. *Journal of Systems and Software*, 4(1):63–74, 1984.

[DG94] W. Deiters und V. Gruhn. The FUNSOFT Net Approach to Software Process Management. *International Journal of Software Engineering and Knowledge Engineering*, 4(2):229–256, 1994.

[DJS92] J. Dufner, U. Jensen, und E. Schumacher. *Statistik mit SAS*. Teubner Studienbücher, 1992.

[FGH+93] A. Finkelstein, D. Gabbay, A. Hunter, J. Kramer, und B. Nuseibeh. Inconsistency Handling in Multi-Perspective Specifications. *IEEE Transactions on Software Engineering*, 20(8):569–578, 1993.

[GF94] O. Gotel und A. Finkelstein. An Analysis of the Requirements Traceability Problem. In *Proc. 1st International Conference on Requirements Engineering*, Seiten 94–101, Colorado Springs, CO, April 1994. IEEE Computer Society Press.

[GF95] O. Gotel und A. Finkelstein. Contribution Stuctures. In *Proc. Second IEEE International Symposium on Requirements Engineering*, Seiten 100–107. IEEE Computer Society Press, März 1995.

[Git86] D. Gittins. Icon-Based Human-Computer Interaction. *International Journal of Man-Machine Studies*, 24(6):519–543, 1986.

[GS90] P.K. Garg und W. Scacchi. A Hypertext System to Manage Software Life-Cycle Documents. *IEEE Software*, Seiten 90–98, Mai 1990.

[GTK89] S.J. Guastello, M. Traut, und G. Korienek. Verbal versus Pictorial Representations of Objects in a Human-Computer Interface. *International Journal of Man-Machine Studies*, 31(1):99–120, 1989.

[Har87] D. Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, Seiten 231–274, Juli 1987.

[Has97] W. Hasselbring. Federated Integration of Replicated Information within Hospitals. *International Journal on Digital Libraries*, 1(4), Dezember 1997.

[HG97] D. Harel und E. Gery. Executable Object Modeling with Statecharts. *Communications of the ACM*, 30(7):31–42, Juli 1997.

[Kai93] H. Kaindl. The missing link in requirements engineering. *ACM Software Engineering Notes*, 19(2):30–39, April 1993.

[Kru95] P.B. Kruchten. The 4+1 View Model of Architecture. *IEEE Software*, 12(6):42–50, November 1995.

[KWR97] A. Krabbel, I. Wetzel, und S. Ratuski. Anforderungsanalyse für Krankenhausinformationssysteme: Definition von Kernsystem und Ausbaustufen. In W. Hasselbring, Hrsg., *Erfolgsfaktor Softwaretechnik für die Entwicklung von Krankenhausinformationssystemen*. Krehl Verlag, Münster, 1997.

[LS96] M. Lindvall und K. Sandahl. Practical Implications of Traceability. *Software – Practice and Experience*, 26(10):1161–1180, Oktober 1996.

[NJJ+96] H.W. Nissen, M.A. Jeusfeld, M. Jarke, G.V. Zemanek, und H. Huber. Managing Multiple Requirements Perspectives with Metamodels. *IEEE Software*, 13(3):37–48, März 1996.

[PDJ97] K. Pohl, R. Dömges, und M. Jarke. Towards Method-Driven Trace Capture. In *Proc. 9th International Conference on Advanced Information System Engineering (CAiSE'97)*, Seiten 103–116, Barcelona, Spain, Juni 1997. Springer-Verlag, LNCS 1250.

[PG96] F.A.C. Pinheiro und J.A. Goguen. An Object-Oriented Tool for Tracing Requirements. *IEEE Software*, 13(3):52–64, März 1996.

[PJ94] K. Pohl und S. Jacobs. Traceability between Cross-Functional Teams. NATURE Report Series 94-13, RWTH Aachen, 1994. (Published at 1st Intl. Conf. on Concurrent Engineering, Pittsburgh, USA, August 1994).

[PK96] D. Platz und U. Kelter. Konsistenzhaltung von Fensterinhalten in einer Repository-basierten SEU. In *Proc. GI-Fachtagung Softwaretechnik 96*, Softwaretechnik-Trends 16/3, Seiten 73–80, Koblenz, Germany, September 1996.

[Poh96] K. Pohl. PRO-ART: Enabling Requirements Pre-Traceability. In *Proc. 2nd International Conference on Requirements Engineering*, Seiten 76–84, Colorado Springs, CO, April 1996.

[RSC97] Rational Software Corporation. The Unified Modeling Language. UML Document Set Version 1.1, Santa Clara, CA, September 1997. (available from www.rational.com).

[RBP+91] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, und W. Lorensen. *Object-Oriented Modelling and Design*. Prentice Hall, Englewood Cliffs, NJ, 1991.

[RUPT90] C. V. Ramamoorthy, Y. Usuda, A. Prakash, und W. T. Tsai. The Evolution Support Environment System. *IEEE Transactions on Software Engineering*, 16(11):1225–1234, November 1990.

[SFR+92] G. Symon, M. Fitter, C. Radstone, I. Kunkler, und B. Hancock. The Process of Deriving Requirements for a Hospital Information System. *Behaviour and Information Technology*, 11(3):131–140, 1992.

[SG96] M. Shaw und D. Garlan. *Software architecture: perspectives on an emerging discipline*. Prentice Hall, 1996.

[Sha95] M. Shaw. Comparing architectural design styles. *IEEE Software*, 12(6):27–41, November 1995.

[TN97] E. Tryggeseth und Ø. Nytrø. Dynamic Traceability Links Supported by a System Architecture Description. In *Proc. International Conference on Software Maintenance (ICSM'97)*, Bari, Italy, September 1997.

[Whi94] S. White. Traceability for complex systems engineering. Technischer Bericht, Grumman Aerospace & Electronics, Bethpage, NY, 1994.

[WZB+96] A. Winter, R. Zimmerling, O. Bott, S. Gräber, W. Hasselbring, R. Haux, A. Heinrich, R. Jaeger, I. Kock, D.P.F. Möller, O. Penger, J. Ritter, A. Terstappen, und A. Winter. Das Management von Krankenhausinformationssystemen: Eine Begriffsdefinition. In *Tagungsband zur 41. GMDS-Jahrestagung*, Seiten 34–38, Bonn, September 1996. MMV Medizin Verlag.