Erprobung einer Middleware-Architektur im Gesundheitswesen

Inaugural-Dissertation
zur Erlangung des Grades eines Doktors der Humanbiologie
des Fachbereichs Humanmedizin
der Justus-Liebig-Universität Gießen

vorgelegt von Ralf Schweiger
aus Crailsheim, Baden-Württemberg

Gießen 2000

aus dem Medizinischen Zentrum für Ökologie
Institut für Medizinische Informatik
Leiter: Prof. Dr. J. Dudeck
des Klinikums der Justus-Liebig-Universität Gießen

Gutachter: Prof. Dr. Dudeck
Gutachter: Prof. Dr. Prokosch
Tag der Disputation: 27.10.2000

Inhaltsverzeichnis

1 Einführung

Fortschritte in der Informationstechnologie (IT) haben oft einen Einfluß auf die Architektur von Informationssystemen, auch im Gesundheitswesen. Das Ziel eines Informationssystems im Gesundheitswesen, das auch als *Healthcare Information System* (HIS) bezeichnet wird, ist unter anderem wie folgt definiert:

> [Hripcsak97] *"... provides health care users with the information they need, when and where they need it".*

Das Ziel der Informationsversorgung sind demzufolge Personen des Gesundheitswesens wie Ärzte [Ruan98], Pflegekräfte [Hannah95] und Patienten [Masys98]. Die IT kann in diesem Zusammenhang die Güte der Informationsversorgung verbessern [Wingert79]. Mögliche Gütekriterien sind die Relevanz, die Vollständigkeit und die Aktualität der Information. Laboranforderungen, Befunde und andere Daten werden dazu in eine elektronische Form gebracht (Datenerfassung) und können dann rasch übermittelt, zusammengeführt und ausgewertet werden. Ein Arzt sucht zum Beispiel für die Diagnose- und Therapiefindung die komplette Krankengeschichte eines Patienten oder für Forschungszwecke alle Krankheitsverläufe zu einer gegebenen Diagnose. Der Arzt gibt zu diesem Zweck nur noch die Suchbegriffe (Patient-Identifikation, Diagnose) ein und überläßt die Suche entsprechenden HIS-Funktionen. Andere Funktionen unterstützen die Pflegedokumentation oder die Erstellung von Arztbriefen aus bereits erfaßten Daten. Mit der zunehmend geteilten Versorgung von Patienten (shared care) gewinnt die Kommunikationsfunktion eines HIS wachsende Bedeutung. In [Prokosch94] wird z.B. von einem Krankenhaus-Kommunikationssystem gesprochen, das die Basis für ein Krankenhaus-Informationssystem darstellt.

Die vorliegende Arbeit beschäftigt sich mit einer modernen IT-Architektur für Informationssysteme im Gesundheitswesen (HIS). Einführend soll daher die architektonische Entwicklung solcher Informationssysteme im Sinne eines Überblicks

aufgezeigt werden. Frandji [Frandji95] unterscheidet im wesentlichen drei IT-Architekturen eines HIS, die im folgenden vorgestellt werden:

In der Großrechner-Ära (bis ungefähr 1985) wurde die Funktionalität eines HIS weitgehend von einem Software-Hersteller bestimmt. Die Grenzen dieser Architektur ergaben sich aus der Abhängkeit von einem Hersteller und aus der Abhängigkeit von einem Rechner. Die Abhängigkeit von einem Hersteller erschwerte insbesondere die Anpassung der HIS-Funktionalität an neue Bedürfnisse (mangelnde funktionale Flexibilität). Häufig war das HIS auf den administrativen Anwendungsbereich eines Krankenhauses (Rechnungswesen, Aufnahme von Patienten) beschränkt. Darüber hinaus entstand eine starke wirtschaftliche Verflechtung zwischen dem HIS-Hersteller und dem Krankenhaus. Aber auch die Abhängigkeit von einem Rechner setzte Grenzen. Waren die Ressourcen eines Großrechners erschöpft, so musste ein noch größerer und teurerer Rechner angeschafft werden. Eine Vernetzung mehrerer Rechner war zu dieser Zeit nicht möglich. Die frühen HIS-Architekturen waren also mit Blick auf die Hard- und Software *zentral* organisiert.

Mitte der achtziger Jahre änderte sich diese Situation durch zwei technologische Fortschritte, die Entwicklung leistungsstarker Arbeitsplatz-Rechner und die Reifung der Netztechnologie [Tanenbaum95, 447]. Die HIS-Architektur entwickelte sich weg von einem Ein-Rechner-Ein-Hersteller-HIS hin zu einem Mehr-Rechner-Mehr-Hersteller-HIS. Anbieter von HIS-Funktionen spezialisieren sich zunehmend auf einzelne klinische Bereiche wie Station, Labor und Radiologie. Man spricht in diesem Zusammenhang auch von Subsystemen (Labor-Subsystem etc.) oder Anwendungssystemen. Jedes Krankenhaus kann sich auf diese Weise sein eigenes HIS maßschneidern. Die Integration der Subsysteme erfolgt durch den Austausch von Nachrichten. Das Laborsystem kann zum Beispiel eine Befundnachricht an das Arbeitsplatzsystem des behandelnden Arztes schicken. Eine Kommunikationsbeziehung zwischen zwei Subsystemen entsteht auch dann, wenn beide Subsysteme (z.B. Labor und Station) dieselben Informationen (Name des Patienten) speichern. Die Aktualisierung der Information im Stationssystem erfordert dann auch eine Aktualisierung der Information im Laborsystem. Die Struktur, Codierung und das Format der ausgetauschten Inhalte wird durch Kommunikationsstandards wie HL7

(Health Level Seven, 1987) und DICOM (Digital Imaging and COmmunications in Medicine, 1993) festgelegt. Kommunikatonsstandards ersparen Absprachen und Übersetzungen zwischen den Subsystemen und erleichtern somit ihre Integration. Die frühen zentralen Architekturen wurden also durch *dezentrale* Architekturen abgelöst.

Die Integration und Kommunikation der Subsysteme in dezentralen Architekturen stellt dennoch ein Problem dar. Erfahrungen zeigen, daß trotz Kommunikationsstandards noch zahlreiche Fragen bei Schnittstellenverhandlungen geklärt werden müssen, da nicht alle realen Kommunikationsbeziehungen und Geschäftsprozesse durch ein Kommunikationsmodell abgedeckt werden können [Heitmann99] oder die Anwendungssysteme nur einen Teil des Kommunikationsstandards unterstützen. Darüber hinaus kann ein einzelnes Anwendungssystem im voraus nicht wissen, welche Kommunikationspartner und Kommunikationsbeziehungen in einem HIS existieren. Der HIS-Entwickler muß folglich den Nachrichtenaustausch zwischen den Anwendungssystemen selbst verwalten (Verteilung der Nachrichten, Übertragungsfehler, Abhörsicherheit etc.). Allein für die Integration ist also eine Menge an Software aufzubringen. Viele Gesundheitseinrichtungen können sich den Entwicklungsaufwand für die Integrationssoftware nicht leisten und bleiben daher schwach integriert. Aus diesem Grund wird derartige Integrations- bzw. Kommunikationssoftware zunehmend am Markt als sogenannte *Middleware* angeboten, um den Eigenaufwand bei der Integration zu reduzieren. Middleware bezeichnet somit jede Art von Software, welche die Integration von Softwaresystemen erleichert. Middleware-Funktionen verwalten zum Beispiel gemeinsame Daten oder machen den Nachrichtenaustausch transparent. Seit Mitte der neunziger Jahre spricht man auch von *Middleware*-Architekturen.

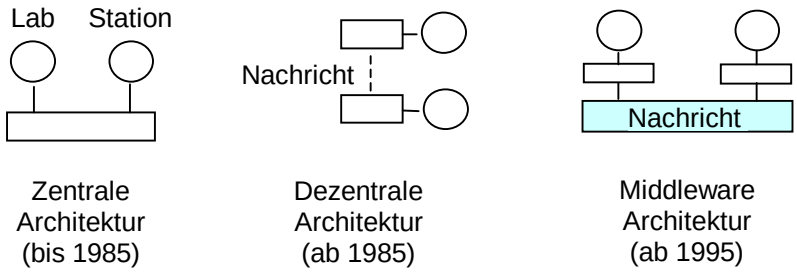

Abbildung 1: IT-Architekturen eines Health Information System (HIS) nach [Frandji95]. Kreise symbolisieren Benutzergruppen, Rechtecke symbolisieren Funktionalität (Software). Starke Integration und geringe funktionale Flexibilität bei der zentralen Architektur. Schwache Integration und hohe Flexibilität bei der dezentralen Architektur. Starke Integration und hohe Flexibilität bei der Middleware-Architektur. Middleware verringert die Kosten der Integration, indem sie den Nachrichtenaustausch zwischen den Anwendungssystemen transparenter macht.

2 Zielsetzung der Arbeit

2.1 Das europäische Projekt HANSA

Das Thema dieser Arbeit ist die „Erprobung einer Middleware-Architektur im Gesundheitswesen“. Mit dieser Zielsetzung wurde Anfang 1996 das europäische Projekt HANSA (Healthcare Advanced Networked System Architecture) ins Leben gerufen. Middleware bezeichnet wie bereits beschrieben eine Software, welche die Integration von anderen Softwaresystemen unterstützt. Die Middleware war zu diesem Zeitpunkt bereits implementiert und wurde den Projektpartnern (Mitglieder aus Industrie und Forschung) in Form des DHE (Distributed Healthcare Environment) zur Verfügung gestellt [Ferrara96a/f/i]. Eine Beschreibung der DHE-Middleware ist in Abschnitt 3.1 zu finden. HANSA umfaßte zunächst neun Länder (HANSA West: Belgien, Dänemark, Deutschland, Griechenland, Italien, Niederlande, Portugal, Spanien und Schweden) und wurde später in Richtung Osteuropa (HANSA East) und Mittleren Osten (HANSA Med) ausgedehnt.

HANSA verfolgte das Ziel, existente Systeme und Anwendungen im Gesundheitswesen mittels DHE zu integrieren.

> [HANSA95] *"To install the DHE middleware in the environment of several European hospitals, and to integrate it with a limited, but significant sample of the applications already existing, with a view of demonstrating its applicability and validity in practice."*

Diese Zielsetzung impliziert folgende Fragestellungen: Wird die Integration der Anwendungssysteme durch das DHE erleichtert? Ist die Funktionalität der DHE-Middleware ausreichend?

2.2 Gießener Zielsetzung im Rahmen von HANSA

Jeder HANSA-Partner wurde mit einer eigenen DHE-Installation ausgestattet und sollte eine oder mehrere Anwendungen mit dem DHE verbinden. Die Aufgabe der Projektpartner bestand nun darin, geeignete Anwendungen zu finden und über die Erfahrungen bei der Integration mit dem DHE zu berichten. Ein entsprechender Erfahrungsbericht findet sich im Ergebnisteil dieser Arbeit.

Eine konkrete Integrationsaufgabe stellte sich für das Gießener Tumordokumentationssystem (GTDS). Das GTDS [Altmann95] unterstützt in über 30 deutschen Krankenhäusern die Basisdokumentation für Tumorkrankheiten und versorgt den Arzt bei der Nachsorge von Tumorpatienten mit Informationen zu vorausgegangenen diagnostischen und therapeutischen Maßnahmen. Eine Beschreibung des GTDS wird in Abschnitt 3.2 gegeben. Als klinisches Krebsregister enthält das GTDS Informationen, die an verschiedenen Stellen eines Informationssystems dokumentiert und wieder abgerufen werden. Das GTDS eignet sich insofern für eine Integration mit anderen Systemen. Konkret stellte sich die Aufgabe, das GTDS mit dem Arbeitsplatz des Arztes auf der Station zu integrieren. Das GTDS wurde zu dieser Zeit überwiegend von speziellem Dokumentationspersonal bedient, das seinerseits Papier-Dokumente (Erfassungsbögen, Übersichtsberichte und Arztbriefe) mit den Stationen austauschte. Eine direkte Bedienung des GTDS durch den Arzt war erwünscht, d.h. es mußte eine elektronische Verbindung zwischen dem GTDS und dem Arbeitsplatzsystem des Arztes geschaffen werden. Diese Integrationsaufgabe sollte mit dem DHE gelöst werden.

Die Aufgabe der Integration von GTDS und Stationsarbeitsplatz mittels DHE sollte in einer Zusammenarbeit der beiden Projektpartner Gießen und Magdeburg gelöst werden. Beide Universitäten verfügen über eine DHE- und eine GTDS-Installation. Die Magdeburger Aufgabe bestand in der Entwicklung einer Schnittstelle zwischen dem DHE und dem Arbeitsplatz des Arztes. Ziel war es, dem Arzt an seinem Arbeitsplatz lesenden Zugang zu den Tumordaten zu verschaffen. Die Gießener Aufgabe bestand in der Herstellung einer GTDS-DHE-Verbindung, welche an Magdeburg zur Demonstration der Gesamtintegration weitergegeben werden sollte.

Über die Gießener GTDS-DHE-Verbindung und die Magdeburger DHE-Station-Verbindung konnte dann insgesamt die Integration zweier unabhängiger Systeme (GTDS, Stationsarbeitsplatzsystem) mittels DHE demonstriert werden.

Das Ziel der vorliegenden Arbeit im Rahmen des HANSA-Projektes war die Lösung der Gießener Integrationsaufgabe, die in der Verbindung von GTDS und DHE bestand:

Abbildung 2: Gießener Zielsetzung im Rahmen von HANSA: Wie können GTDS und DHE verbunden werden?

Zusammenfassend können für diese Arbeit also folgende Aufgaben identifiziert werden:

1) Anbindung des GTDS an das DHE
2) Untersuchung der Integrationserfahrungen
3) Ableitung von Vorschlägen zur Verbesserung des DHE

3 Umfeld der Arbeit

Gemäß Zielsetzung soll das GTDS (Gießener Tumordokumentationssystem) mit dem DHE (Distributed Healthcare Environment) verbunden werden. In diesem Kapitel werden die gegebenen Systeme GTDS und DHE genauer beschrieben. Die Betonung liegt in diesem Zusammenhang auf dem DHE, d.h. der zu erprobenden Middleware. Die gesuchte Verbindung der beiden Systeme wird in einem getrennten Kapitel beschrieben.

3.1 Das Distributed Healthcare Environment (DHE)

3.1.1 Das Modell: Informationen im DHE

DHE ist der Name der Middleware (Integrationssoftware), die im Rahmen dieser Arbeit erprobt werden soll. Das DHE stellt eine Middleware für das Gesundheitswesen (Krankenhaus, niedergelassene Ärzte etc.) dar. Die Zielsetzung der DHE-Middleware für das Gesundheitswesen kommt in dem zugrundeliegenden Datenmodell zum Ausdruck [Ferrara96i]. Dieses Datenmodell repräsentiert Informationen, die übergreifend für mehrere Gesundheitseinrichtungen (Station, Labor, Radiologie etc.) von Interesse sind. Das DHE-Datenmodell unterstützt somit die inhaltliche Integration und Kommunikation von Anwendungssystemen im Gesundheitswesen.

Abbildung 3 zeigt nur 4 von über 200 *Entitäten* des DHE-Modells: den *Patienten*, das *Medizindatum* (Health datum), den *Medizindatentyp* (Type of health data) und die *Maßnahme* (Act). Maßnahmen sind Aktivitäten, die der Diagnostik (Labor-, Röntgenuntersuchung), der Therapie (Bestrahlung, Operation, Medikation) oder der sonstigen Versorgung des Patienten dienen. Medizindaten sind beliebige klinisch relevante Informationen wie z.B. die Lokalisation und Histologie eines Tumors oder die Vitalparameter eines Patienten. Jede Entität verfügt über mehrere Merkmale bzw. *Attribute*. Ein Patient hat zum Beispiel die Attribute *Nachname*, *Geschlecht* und *Geburtsdatum*. Da eine Entität 30 und mehr Attribute haben kann, sind in dem vorliegenden Diagramm nicht alle Attribute aufgelistet. Darüber hinaus sind *Beziehungen* zwischen den Entitäten bzw. ihren Attributen dargestellt. Ein Medizindatum wird z.B. durch einen Medizindatentyp beschrieben, gehört zu einem

Patienten und kann Ergebnis einer Maßnahme sein. In Klammern sind ggfs. die DHE-Bezeichner der Entitäten (pmPA) und Attribute (pa_lname) angegeben. Diese Bezeichner werden in Abschnitt 4.5 eine Rolle spielen.

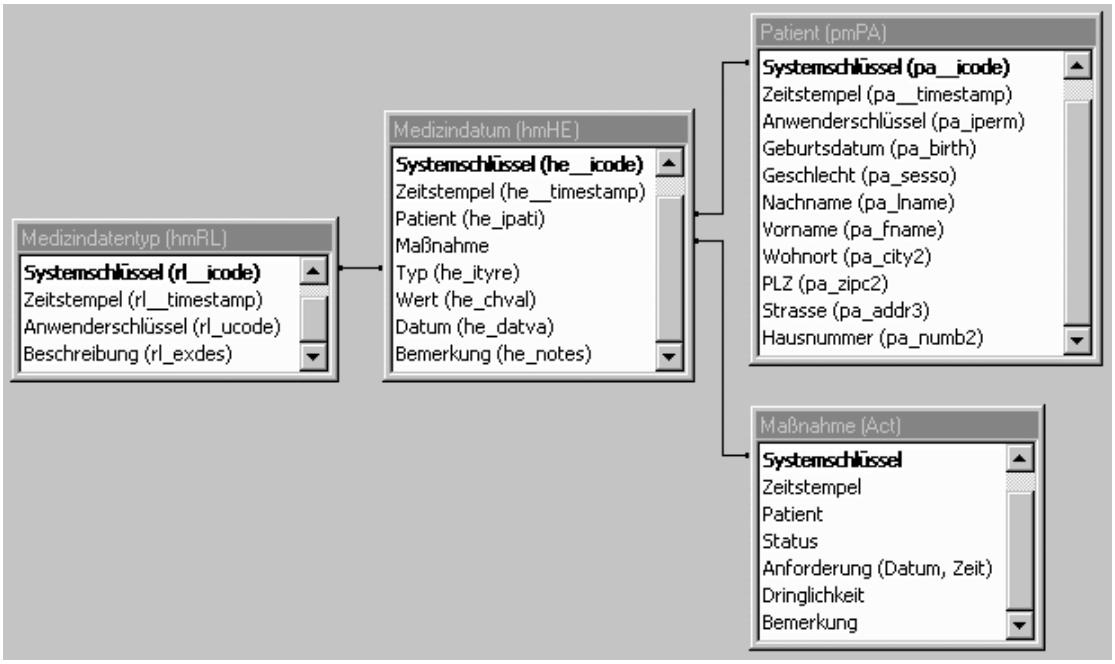

Abbildung 3: Ausschnitt aus dem Datenmodell des DHE.

Das DHE verfügt ferner über die Möglichkeit, zu einer beliebigen Entität sogenannte *BLOBs* (Binary Large OBject) hinzuzufügen. BLOBs sind Daten mit beliebigem Datenformat wie zum Beispiel Bildformate (Röntgenbild), Videoformate (Ultraschall) und Programmformate (Bildverarbeitung). Die Interpretation der BLOB-Formate erfolgt durch die Anwendung und nicht durch das DHE, das diese Formate gar nicht kennen muß. Im DHE-Jargon werden BLOBs auch als *Extended data* bezeichnet.

Die Entitäten des DHE-Modells können in zwei Kategorien eingeteilt werden. Es gibt Entitäten wie den *Medizindatentyp*, die ein Informationssystem im Gesundheitswesen beschreiben. Ein Medizindatentyp klassifiziert und beschreibt Medizindaten. Beispielsweise könnte man für die Pflegedokumentation einen Medizindatentyp „Blutdruck in mmHg nach Riva Rocci“ einführen und fortan entsprechende Medizindaten („120/80“, „200/120“ usw.) mit dem DHE verwalten. Weitere beschreibende Entitäten sind die *Leistungsstelle* (Point of service) und die *Leistung*

selbst (Type of activity). Durch solche Beschreibungen kann jedes Informationssystem individuell konfiguriert werden (Konfigurationsdaten). Das eine System verfügt zum Beispiel über eine Leistungsstelle *Radiologie*, das andere nicht. Abhängig von der Verfügbarkeit eines Computertomographen kann die Radiologie eine *CT*-Untersuchung anbieten. Andere Entitäten wie der *Patient*, die *Maßnahme* und das *Medizindatum* (Diagnose, Befund) repräsentieren hingegen Informationen, die im laufenden Betrieb eines Krankenhauses entstehen (Betriebsdaten).

Auch die Attribute des DHE-Modells können in zwei Kategorien eingeteilt werden. Es gibt Attribute wie den *Systemschlüssel* und den *Zeitstempel*, die jede Entität besitzen muß und die vom DHE vergeben werden (*Systemattribute*). Bei der Aufnahme eines neuen Patienten erzeugt das DHE automatisch einen Systemschlüssel für diesen Patienten. Der Systemschlüssel wird benutzt, um Beziehungen zwischen den Entitäten auszubilden. Beispielsweise wird dem Patienten eine Maßnahme zugeordnet, indem der Systemschlüssel des Patienten in dem Patient-Attribut der Maßnahme-Entität abgelegt wird (siehe Diagramm). Der Systemschlüssel erlaubt somit die Identifikation des Patienten unabhängig von der Verfügbarkeit der Patientendaten (Aufnahme von Komapatienten). Der Zeitstempel signalisiert die letzte Veränderung einer Entität und wird vom DHE automatisch aktualisiert. Auf diese Weise kann das DHE Konflikte erkennen, die bei der gleichzeitigen Veränderung einer Entität durch mehrere Benutzer entstehen. Im Unterschied zu den Systemattributen werden die übrigen Attribute wie das *Geschlecht* oder das *Geburtsdatum* des Patienten nicht vom DHE, sondern von der Anwendung bestimmt (*Anwenderattribute*). Solche anwendungs-eigenen Identifikatoren können jedoch nicht für die Ausbildung von Beziehungen zwischen den Entitäten benutzt werden, d.h. es muß immer zuerst der entsprechende Systemschlüssel ermittelt werden.

Das zentrale Konzept des DHE-Modells ist die Maßnahme bzw. der *Act*. Dieses Konzept ist in [Frandji95] definiert als *„provision of care to the patient"* und repräsentiert jede Versorgungsleistung für den Patienten. Beispiele für Acts sind Untersuchungen, Behandlungen und sonstige Leistungen (Hämoglobin bestimmen, Ultraschall, Gastroskopie etc.). Ein Act ist gekennzeichnet durch einen *Act status*, der

beispielsweise die Werte *angefordert, eingeplant, begonnen, vorläufig berichtet* und *abgeschlossen* annehmen kann. Der Status einer Maßnahme wird in der Regel von mehreren Leistungsstellen bearbeitet und kann bei einer Röntgenuntersuchung zum Beispiel folgenden Verlauf nehmen:

Bearbeiter des Acts	**setzt Act status auf**
behandelnder Arzt	angefordert
Röntgen-Sekretariat	eingeplant
MTA	durchgeführt
Radiologe	vorläufig befundet
erfahrener Radiologe (Oberarzt, Chefarzt)	abgeschlossen

Tabelle 1: Möglicher Verlauf eines Acts am Beispiel der Röntgenuntersuchung.

Der Werteverlauf des Act status wird als *Act life cycle* bezeichnet und hat im Beispiel der Röntgenuntersuchung die Ausprägung *(angefordert, eingeplant, durchgeführt, vorläufig befundet, abgeschlossen)*. Über den Act status kann der behandelnde Arzt jederzeit den Stand der Untersuchung erfahren. Zum aktuellen Stand der Untersuchung zählen neben dem Bearbeitungsschritt der Maßnahme auch *Zwischenergebnisse* wie ein *Termin*, eine *Bemerkung* (Hämatologielabor: zu wenig Serum für eine Hb-Bestimmung), ein *Verdacht* oder ein *vorläufiger Befund*.

Das nächste Diagramm betont einen weiteren Aspekt des Act-Konzepts. Im Zentrum dieser Abbildung steht die *elektronische Patientenakte* (Patient medical record), die durch das Maßnahmekonzept „gefüttert“ wird. Mit jedem Bearbeitungschritt einer Maßnahme werden Ergebnisse dokumentiert und somit bereits zum Zeitpunkt ihrer Entstehung elektronisch erfaßt. Die elektronische Patientenakte erreicht auf diesem Weg ein hohes Maß an Aktualität. Ferner kann die Akte von allen an der Versorgung eines Patienten beteiligten Personen eingesehen und konsultiert werden. Verschiedene Sichten auf die Akte für Ärzte, Pflegekräfte usw. können realisiert werden. Das Maßnahmekonzept verringert also auf natürliche Weise die Zeitabstände der

elektronischen Datenerfassung. Werden zum Beispiel Befunde noch vor der Diagnose elektronisch dokumentiert, so können wissensbasierte Funktionen dem Arzt bei der Diagnosefindung entsprechende Vorschläge unterbreiten.

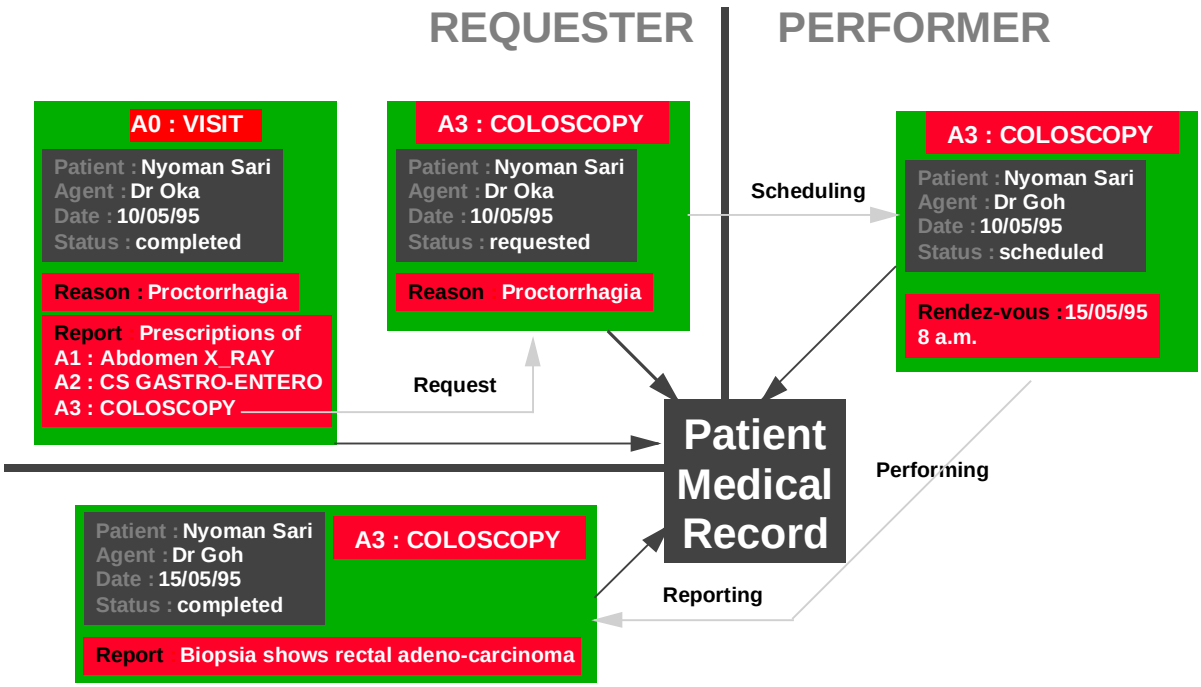

Abbildung 4: Ergebnisse von Maßnahmen werden in der elektronischen Krankenakte (patient medical record) gesammelt. Entnommen aus [Frandji95]. Bei einer Visite (Act A0) wird die Notwendigkeit einer Koloskopie (Act A3) festgestellt. Der Patient wird an den entsprechenden Spezialisten verwiesen, ein Termin ist eingeplant. Die Koloskopie wird durchgeführt und ein Bericht wird erstellt. Alle Informationen, die während der Maßnahme entstehen, werden in der elektronischen Patientenakte (Patient medical record) gesammelt.

3.1.2 Was ist eine Middleware?

Während im vorigen Abschnitt das Modell des DHE in Auszügen betrachtet wurde, soll in den folgenden Abschnitten die DHE-Software beschrieben werden [Ferrara96f]. Das DHE wurde als eine Middleware eingeführt. In diesem Abschnitt soll der Middleware-Begriff definiert werden.

In verteilten Informationssystemen wie dem Krankenhaus erfolgt die Kommunikation durch den Austausch von Nachrichten. Bei der Integration von verteilten Systemen muß der Softwareentwickler für die Implementierung der Kommunikationsbeziehungen zahlreiche Einzelheiten über die Kommunikationspartner (Adresse, von außen benötigte und von außen abrufbare Informationen, Repräsentierung der Information, physikalische Verbindung etc.) wissen. In größeren Systemen kann der Entwickler dabei schnell den Überblick verlieren und die Integration wird sehr aufwendig. Dieser Umstand motiviert die Frage, ob man nicht viele Einzelheiten der Integration vor dem Entwickler verbergen kann. In der Datenverarbeitung (DV) spricht man in diesem Zusammenhang auch von *Transparenz* (Ortstransparenz, Replikationstransparenz etc.). Middleware soll diese Transparenz schaffen, weniger für den Anwender als vielmehr für den Entwickler:

> [Tanenbaum95, 474] *„Am einfachsten ist es, die Verteiltheit vor den Benutzern zu verbergen. ... Auf einer tieferen Ebene ist es möglich, wenn auch schwieriger, ein System transparent für die Programme zu machen."*

Nach [DeJesus96] ist Middleware eine Software, welche die Verbindung von entfernten Softwaresystemen erleichtert:

> [DeJesus96] *„The goal is to ease the strains of remote connection."*

Die Anstrengungen (*„strains of remote connection"*) liegen oftmals in der Durchführung des Nachrichtenaustauschs mit primitiven Funktionen zum Verschicken und Empfangen von Nachrichten. Middleware gibt dem Entwickler mächtigere Funktionen an die Hand. In der Literatur [Rymer96, Spector98, Dolgicer99] werden unter anderem folgende Beispiele für Middlewarefunktionen genannt:

- *Brokering (Vermittlungsdienste)*
- *Message Queueing (Nachrichtenverwaltung)*
- *Transaction processing (Transaktionsverarbeitung)*
- *Directory services (Verzeichnisdienste)*
- *Security services (Sicherheitsdienste)*

Broker machen das Nachrichtenprotokoll weitgehend transparent. Der Entwickler muß sich nicht mehr darum kümmern, wie die Datenelemente in der Nachricht begrenzt sind oder wie die Informationen in der Nachricht codiert werden. Darüber hinaus können Broker den Nachrichtenverkehr (z.B. Sendung einer Frage-Nachricht und Empfang einer Antwort-Nachricht) regeln. *Message queuing* bezeichnet eine Art Mailbox für Programme, d.h. die Nachrichten sind nicht für den Anwender sondern für das Programm bestimmt. Der Entwickler bekommt u.a. Methoden zum Beantworten (Reply) und zur Verteilung von Nachrichten (Forward) an die Hand. *Transaktionsmonitore* überwachen die Alles-oder-Nichts-Ausführung einer Folge von Aktionen. Beispiele für Aktionsfolgen bzw. Transaktionen sind „Lastschrift, Gutschrift“ oder „Änderung des Patientennamens im Stationssystem, Änderung des Patientennamens im Laborsystem“. Der Entwickler kann eine Transaktion als Ganzes durchführen (Commit) oder zurücknehmen (Rollback). Auf diese Weise wird sichergestellt, daß im ersten Fall keine Gutschrift ohne Lastschrift erfolgt, und im zweiten Fall der Patientenname auf der Station nicht verändert werden kann, ohne daß er auch im Laborsystem geändert wird (Kontrolle von Datenredundanz). In verteilten Systemen werden zu diesem Zweck sogenannte Commit-Protokolle durchgeführt, die für den Entwickler transparent sind. *Verzeichnisdienste* verwalten Beziehungen zwischen Begriffen wie Untersuchungsleistung und Leistungsstelle oder Leistungsstelle und Rechneradresse. *Sicherheitsdienste* ver- und entschlüsseln die Nachrichten und schützen den Nachrichteninhalt vor unbefugtem Zugriff (elektronische Unterschrift etc.).

Die DHE-Middleware integriert ebenfalls Anwendungssysteme, allerdings über inhaltliche Konzepte wie *Patient*, *Befund* und *Untersuchung*, die spezifisch für das Gesundheitswesen sind (siehe 3.1.1). Das DHE wird daher einschränkend als *Healthcare Middleware* bezeichnet. Nach einer Untersuchung von [Spahni98] gibt es einen Trend zur Entwicklung von domänenspezifischer Middleware. Die Einschränkung auf eine Domäne wie das Gesundheitswesen hat den Vorteil, daß die Middleware um ein domänenspezifisches Modell bereichert werden kann und somit die Integration durch noch mächtigere Funktionen unterstützen kann. Beispiele für

domänenspezifische Middleware sind ein Codeverzeichnis für die im Klinikum angebotenen Untersuchungen oder eine Anforderungsfunktion, die automatisch die Terminplanung der leistungserbringenden Stelle sowie das Verwaltungssystem für die spätere Abrechnung kontaktiert. Das DHE liegt mit seinem Datenmodell im Trend dieser Entwicklungen. Middleware kann demzufolge nicht nur technische, sondern auch inhaltliche Aspekte der Integration umfassen. Zusammenfassend wird der Middleware-Begriff nun wie folgt definiert:

> *Middleware: Software, die den Entwickler bei der Integration von Softwaresystemen unterstützt.*

3.1.3 Das DHE - Eine Middleware?

Nachdem der Middleware-Begriff definiert ist, sollen in diesem Abschnitt die Middlewarefunktionen des DHE erörtert werden.

Es gibt prinzipiell zwei Möglichkeiten, wie ein Kommunikationsmodell in Software umgesetzt werden kann. Eine Möglichkeit besteht in der Definition von Nachrichten, die von Software-Einheiten, den Nachrichtenschnittstellen, zusammengesetzt, übertragen und wieder zerlegt werden. Das DHE nutzt eine andere Möglichkeit, die in der Implementierung einer Datenbank besteht. Das DHE steuert mit seinem Modell ein entsprechendes Datenbankschema bei. Eine Datenbank speichert und organisiert große Datenmengen so, daß gesuchte Informationen ggfs. schnell wieder selektiert werden können. Datenbanksysteme wurden mit der Zielsetzung geschaffen, anwendungsübergreifende Daten zu verwalten. Datenbanken dienen also der *Datenintegration* von Anwendungen, d.h. eine Anwendung kann die Daten der anderen Anwendungen einbeziehen [Duden88, 137-142]. Eine Datenbank stellt insofern eine integrationsunterstützende Software, d.h. eine Middleware dar. Damit ist ein erster Middleware-Aspekt des DHE gefunden.

Weitere Middleware-Aspekte des DHE findet man, wenn man die Software-Architektur des DHE betrachtet (Abbildung 5). Das DHE besteht aus einer datenbankseitigen DHE-Komponente, dem *DHE Server*, und einer anwendungsseitigen

DHE-Komponente, dem *Application Programming Interface* (API). Die Anwendung schlüpft dabei in die Rolle eines *DHE Client*. Nach [Duden88, 517-518] beschreibt die *Schnittstelle* (engl. interface) eines Systems alle von außen benötigten und alle von außen abrufbaren Größen. Entsprechend beschreibt das API dem Anwendungsentwickler, wie auf das DHE aus einem Programm heraus zugegriffen werden kann. Softwaremäßig entspricht das API einer Bibliothek von Funktionen, die der Entwickler zu seinem Programm dazubinden und somit aus seinem Programm heraus aufrufen kann.

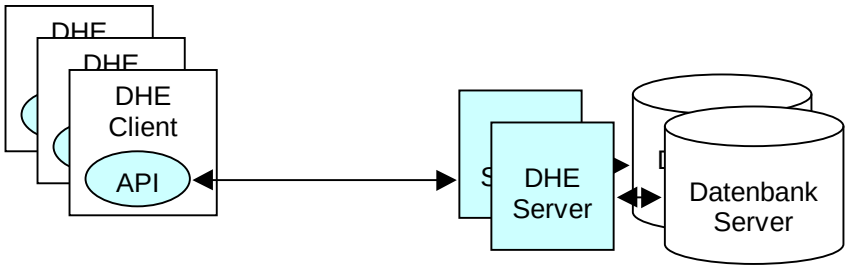

Abbildung 5: Software-Architektur des DHE. Das DHE besteht aus anwendungsseitigen APIs (Application Programming Interface) und datenbankseitigen DHE Servern. Die Anwendung fungiert als DHE Client. Mögliche DHE-Klienten sind das Laborsystem und das Stationssystem. Für den Entwickler sind folgende technische Einzelheiten transparent: die Client/Server-Kommunikation, die zugrundeliegende Datenbank (relational, objekt-orientiert) und die DHE-Replikation bei Versorgung einer großen Anzahl von Anwendungen.

Aus der Software-Architektur des DHE lassen sich die folgenden Middlewarefunktionen ableiten: Transparenz der *Entfernung*, Transparenz der *Skalierung* und Transparenz der *Plattform*. Transparenz wird in diesem Zusammenhang nicht im Sinne von „durchsichtig“, sondern im Sinne von „nicht sichtbar, nicht bemerkbar“ verwendet. *Entfernungstransparenz* heißt, daß der Anwendungsentwickler sich nicht mehr um die Entfernung zum DHE-Server und zu

den anderen Anwendungen in einem DHE-System kümmern muß, d.h. der Nachrichtenaustausch wird durch einfache API-Aufrufe ersetzt. *Skalierungstransparenz* bedeutet, daß die DHE-Datenbanken repliziert werden können, ohne daß sich der Entwickler um die Replikationskontrolle kümmern muß. Die DHE-Server verständigen sich zu diesem Zweck untereinander. Die Skalierung von Client/Server-Systemen stellt in einem Klinikum mit vielen Anwendungssystemen durchaus ein Problem dar. Das DHE verbessert überdies die *Plattform-Transparenz*, indem für verschiedene Datenbanksysteme entsprechende DHE Server bereitgestellt werden. Der DHE Client ist unabhängig davon, welches Datenbankmodell (relational, objekt-orientiert etc.) vom DHE Server verwendet wird. Eine Weiterentwicklung der Server-Plattform berührt den Anwendungsentwickler somit nicht mehr. Während relationale Datenbanken beispielsweise nur über einen Datentypen verfügen, nämlich die Relation zusammen mit der relationalen Algebra, können zu objekt-orientierten Datenbanken neue Datentypen wie das Bild zusammen mit Methoden der Bildverarbeitung (Kontrastierung, Subtraktion etc.) hinzugefügt werden. Außerdem wird das API an verschiedene Client-Plattformen angepaßt. Zur Client-Plattform gehört neben dem Betriebssystem (Unix, Windows) auch die Programmiersprache (C, C++, Visual Basic, Java), in der die Anwendungen implementiert werden.

Einen weiteren Middleware-Aspekt des DHE bilden die *Verzeichnisse*, die das DHE bereithält. Beispielsweise kann im DHE ein Verzeichnis der Leistungsstellen eines Krankenhauses angelegt werden, wobei zu jeder Leistungsstelle die Adresse, Beschreibungen der angebotenen Leistungen und andere Informationen verzeichnet werden. Solche elektronischen Nachschlagewerke können von der Anwendung dazu genutzt werden, um zum Beispiel automatisch, d.h. ohne die Adresse vom Arzt zu erfragen, einen Befund zum richtigen System weiterzuleiten. Verzeichnisse bilden den Schlüssel zu generischen Anwendungen, wobei die Verzeichnisse mit individuellen Daten gefüllt werden (Konfigurierbarkeit) und den Arzt oder die Pflegekraft bei der Dateneingabe entlasten.

3.1.4 Das API: Programmierung mit dem DHE

Das *API* (Application Programming Interface) des DHE beschreibt einem Softwareentwickler, wie er auf das DHE aus einem Programm heraus zugreifen kann [Ferrara96a]. Das DHE stellt jeder Entität im Datenmodell drei Zugriffsfunktionen zur Seite für die

- *Abspeicherung von Entität-Instanzen (Maintain)*
- *Selektion von Entität-Instanzen (List)*
- *Ermittlung der Attribute einer Entität-Instanz (Row)*

In Klammern sind die DHE-Bezeichnungen für die Zugriffsfunktionen angegeben. Diese Funktionen sollen nun am Beispiel der Patient-Entität (pmPA) beschrieben werden.

Für die Abspeicherung von Patientendaten im DHE gibt es eine Funktion *pmPA__M* (Maintain Patient). Die zu speichernden Daten werden in einer Datenstruktur namens *pmPA__IM* (Input for Maintain) abgelegt, die als Parameter an die Funktion übergeben wird. Die Datenstruktur umfaßt im wesentlichen die Attribute der Patient-Entität.

Für die Selektion einzelner Patienten aus der Gesamtheit der gespeicherten Patienten stellt das DHE eine Funktion *pmPA__L* (List Patient) bereit. In der Datenstruktur *pmPA__IL* (Input for List) werden die Selektionskriterien (Geburt in 1976, Name beginnt mit M etc.) abgelegt. Für jeden gefundenen Patienten liefert die Funktion eine Datenstruktur *pmPA__OL* (Output of List), welche die Systemattribute (Systemschlüssel, Zeitstempel) und einige wichtige Anwenderattribute (Name, Geschlecht, Geburtsdatum) des Patienten enthält. Mit der List-Funktion sucht man zu gegebenen Attributen alle dazugehörigen Entitäten (Systemschlüssel).

Die List-Funktion liefert aber nicht alle Attribute eines Patienten. Beispielsweise enthält die Datenstruktur *pmPA__OL* nicht die Anschrift des Patienten. Zur Ermittlung sämtlicher Attribute eines Patienten gibt es die Funktion *pmPA__R* (Patient Row), an die im wesentlichen der Systemschlüssel des Patienten übergeben wird. Als Ergebnis

liefert diese Funktion eine Datenstruktur *pmPA__OR* (Output of Row), die alle Attribute des Patienten enthält. Mit der Row-Funktion sucht man zu einer gegebenen Entität (Systemschlüssel) alle Attribute.

Das folgende API-Programm verwendet die List- und die Row-Funktion der Patient-Entität, um die Anschriften aller Patienten mit Geburt in 1976 auszugeben. Die Attributnamen *pa_birth*, *pa__icode* und *pa_addr* repräsentieren das Geburtsdatum, den Systemschlüssel und die Adresse des Patienten.

```
pmPA__IL.pa_birth = "1976"
pmPA__L(pmPA__IL, pmPA__OL)
index = 0
Solange pmPA__OL[index] definiert ist
        pmPA__IR.pa__icode = pmPA__OL[index].pa__icode
        pmPA__R(pmPA__IR, pmPA__OR)
        Ausgabe pmPA__OR.pa_addr
        erhöhe index um 1
```

Abbildung 6: Vereinfachtes API-Programm zur Ermittlung der Anschriften aller Patienten mit Geburt in 1976.

Der Aufbau der API-Dokumentation soll am Beispiel der Funktion *pmPA__M* verdeutlicht werden. Jede Funktion zeigt den Erfolg des Zugriffs an. Der Zugriff kann zum Beispiel scheitern, wenn während einer Veränderung der Patientendaten eine andere Anwendung die Patientendaten verändert hat. Der gelesene Zeitstempel ist dann nicht mehr aktuell und das DHE verweigert den Zugriff. Auf diese Weise kann die Veränderung der anderen Anwendung nicht ungelesen überschrieben werden. Der zurückgegebene Fehlercode ermöglicht also eine differenzierte Fehlerbehandlung durch das Programm. Darüber hinaus sind in der API-Dokumentation Anforderungen (Constraints) an die Eingabeparameter zu finden. Beispielsweise werden die Patientendaten nur dann gespeichert, wenn mindestens der Name, das Geschlecht und das Geburtsdatum des Patienten gegeben sind und das Geburtsdatum dem DHE-Format YYYY/MM/DD genügt.

Service: pmPA__M

SYNOPSES: Maintain (i.e. add, modify, delete) one full instance of object: Patient

API:

err = pmPA__M (char * i_action, pmPA__IM * is_instance, dh__Obj_OM * os_iden)

DETAIL OF PARAMETERS

* i_action	Action to be performed: A[dd] / M[odify] / D[elete]	char
* is_instance	Structure with the full data on the instance to be entered/modified	pmPA__IM
* os_iden	Structure with the returned identifier of the entered/modified instance	dh__Obj_OM

COMPOSITION OF STUCTURES

pmPA__IM

1 pa__icode	Unique identifier of the instance (defined automatically when generated)	SYSCOD
9 pa_iperm	Permanent identifier of the patient in the hospital	LONG_CODE
16 pa_lname	Last name of patient	PAT_NAME
17 pa_sesso	Sex of patient M / F / N	ONE
18 pa_birth	Date of birth	DATE

dh__Obj_OM

1 _icode	Internal identifier of the instance which has been entered/modified	SYSCOD
2 _timestamp		TIMESTAMP

CONSTRAINTS & DEFAULTS

if (IS_EMPTY(is_instance->pa_lname)) RETURN_ERROR(GS_NO_INPUT)

if (NOT IS_ONE(is_instance->pa_sesso, „MFN“)) RETURN_ERROR(GS_OUT_DOMAIN)

Abbildung 7: Auszug aus dem Application Programming Interface des DHE [Ferrara96a]. Die Datenstruktur pmPA__IM enthält die Attribute des Patienten, wobei nicht alle Attribute (siehe Nummern) aufgelistet sind. Vorbedingung für den Aufruf der Funktion ist u.a., daß der Nachname des Patienten gegeben ist und daß das Geschlecht mit M, F oder N codiert ist (Constraints).

3.1.5 Manager im DHE – Eine Componentware?

Das Datenmodell des DHE umfaßt über 200 Entitäten. Es ist daher sinnvoll, die Informationen im DHE noch weiter zu strukturieren. Entitäten wie der *Patient* und der *Krankenhausaufenthalt* dienen zum Beispiel der Verwaltung von Patienten und werden daher zu einer Komponente, dem *Patient manager*, zusammengefaßt. Dies erklärt auch, warum der Patient im DHE mit *pmPA* bezeichnet wird (patient manager, PAtient). Das DHE unterscheidet darüber hinaus einen *Act manager*, einen *Health data manager*, einen *Resource manager* (Personal, Medizin Technik, Material) und einen *User manager* (Autorisierung der Zugriffe). Die Zusammenfassung von Entitäten zu Managern beschleunigt nicht nur die Suche nach Informationen innnerhalb des Modells (Übersicht), sondern hat auch einen sehr pragmatischen Hintergrund: Das Gießener Klinikum verfügt zum Beispiel bereits über eine Patientenverwaltung und ist daher nicht am Kauf der kompletten DHE-Middleware interessiert. Unterteilt man die DHE-Middleware dagegen in entsprechende Softwarekomponenten, so kann das Klinikum den Health data manager unabhängig vom Patient manager erwerben. Das DHE folgt damit den Methoden des *Component-based software development* (CBD). Die mit Methoden des CBD entwickelte Software bezeichnet man auch als *Componentware*. Im folgenden wird die Zielsetzung des CBD sowie die Einordnung des DHE in das CBD beschrieben.

Der zentrale Begriff des CBD ist die Software-*Komponente*, die zunächst definiert werden soll. Das charakteristische Merkmal einer Komponente ist die Wiederverwendbarkeit bei der Anwendungsentwicklung:

> [*Williamson97*] *"Component: A software module designed to be used repeatedly in developing applications, either with or without modification."*

Neben der Wiederverwendung einzelner Software-Module fördert das CBD die Wiederverwendung ganzer Architekturen (Architektur: Beziehung von Komponenten), die an definierten Stellen angepaßt werden können. Anpassbare Architekturen nennt man auch *Frameworks*:

> [Roundtable98] Pree W, Pomberger G: *„Single-component reuse is less attractive. It means that programmers build the overall software system architecture on their own. They have to locate the appropriate components in a Lego building block library and define their interactions. Framework architectures form the enabling technology of plug & play software, where most adaptations can be achieved by exchanging components. Visual, interactive composition tools are ideally built on top of domain-specific frameworks.“*

Zur Beschreibung von Komponenten und ihren Beziehungen werden ADLs (*Architecture Description Language*) eingesetzt [Roundtable98]. Ein bekannter ADL-Vertreter ist die Unified Modeling Language [Fowler97]. Die Unabhängigkeit der Beschreibungssprache von der Implementierungstechnologie (Objekt-Orientierung, Client/Server, Middleware etc) ermöglicht die Verwendung der Komponenten über lange Zeiträume.

> [Wilkes99] *"CBD is not prescriptive in the way in which systems are implemented physically. CBD is about how applications are architected, not about the technology used to implement them, although new technologies might make some of this easier to achieve."*

Die Zielsetzung des CBD ist somit die Wiederverwendung von Software zur Beschleunigung der Softwareproduktion. Das IEEE (Institute of Electrical and Electronics Engineers) versucht Methodologie und Terminologie der Software-Wiederverwendung zu standardisieren, indem es sogenannte *Wiederverwendungs-prozesse* definiert. Die folgende Tabelle gibt Beispiele für solche Wiederverwendungs-prozesse [IEEE97]:

Reuse process	Description
Reuse definitions	Reuse vocabulary such as Grey Box Reuse (Reuse of a part by applying a few changes to the part) and Generator (Mechanism that creates assets by inclusion of information that is internal to the generator)
Reuse tools	Generator, Catalog descriptors & Search tools ...
Application engineering	Building a single software product with reusable assets (asset = component)
Domain engineering	Creating reusable assets within a domain to be used in developing software products

Tabelle 2: Wiederverwendungsprozesse gemäß [IEEE97]. Das IEEE (Institute of Electrical and Electronics Engineers) versucht die Terminologie und Methodologie der Software-Wiederverwendung zu standardisieren.

Im DHE kommen vor allem die Methoden *Domain engineering* und *Application engineering* zur Anwendung. Das Domain engineering manifestiert sich in der Definition der *DHE-Manager*, d.h. wiederverwendbarer Komponenten für das Gesundheitswesen. Das Application engineering äußert sich in Vorschriften zur Anwendungsentwicklung, konkret der sogenannten *DHE-Konformanz*. Die Konformanz definiert eine klare Verwaltungskompetenz für anwendungsübergreifende Daten, wobei Kompetenz im Sinne von Fähigkeit und Zuständigkeit zu verstehen ist. Diese Kompetenz liegt beim DHE, denn die Integration der verschiedenen Anwendungen wird nur dann funktionieren, wenn auch alle Anwendungen das DHE benutzen. Eine Anwendung, die Patientendaten ausschließlich mit lokalen Funktionen, d.h. ohne den Patient manager des DHE verwaltet, verhält sich demzufolge nicht DHE-konform.

Middleware und Componentware wollen also beide die Software-Entwicklung beschleunigen, aber mit unterschiedlichen Methoden: Middleware unterstützt die Software-Integration, Componentware die Software-Wiederverwendung. Die Formel für die zukünftige Anwendungsentwicklung könnte also wie folgt lauten, wobei das DHE etwas Componentware und etwas Middleware beisteuert:

Anwendung = Componentware + Middleware + Eigenentwicklung

3.1.6 Fundierung auf dem Standard HISA

Die wesentlichen Konzepte des DHE wurden bereits in dem Forschungsprojekt *RICHE* (Reseau d'Information et de Communication Hospitalier Europeen, 1989-1992) erarbeitet. Das Ziel von RICHE war die Entwicklung eines offenen Informationssystems, das wie folgt definiert wird [Frandji95]:

> *(1) system which function is perfectly defined according to the user needs,*
> *(2) is actually provided by a partition of system components,*
> *(3) the definitions of interfaces belong to the public domain,*
> *(4) allowing free usage by different suppliers participating to the market in an open, competitive environment.*

Das Ziel einer offenen Architektur ist die rasche Befriedigung der Benutzer (Ärzte, Pflegekräfte und sonstiges Personal) mit Anwendungsfunktionalität (1). Die Offenheit bezieht sich also auf die Anwendungsfunktionalität eines Informationssystems. Ein einzelner Hersteller wird wegen des Umfangs und der Dynamik der Benutzeranforderungen diese Aufgabe nicht erfüllen können, woraus sich die Forderungen (2), (3) und (4) ableiten. Mit *system components* sind in diesem Zusammenhang das Laborsystem, das Verwaltungssystem und andere Anwendungssysteme gemeint. Damit sich unabhängig voneinander entwickelte Anwendungssysteme verstehen können, müssen die Schnittstellen standardisiert werden (3). Die vorliegende Definition eines offenen Systems macht dabei keine Aussage darüber, was genau die Schnittstelle ist. Die Schnittstelle zwischen den Anwendungen kann ein Nachrichtensystem wie HL7 oder eine Middleware wie das DHE sein. Im Falle von RICHE war dies eine Middleware-Architektur, die dann später in Form des DHE implementiert wurde. Eine Anwendung verhält sich *konform*, wenn sie zur Kommunikation mit anderen Anwendungen die definierte Schnittstelle benutzt. Langfristig möchte RICHE einen Markt für RICHE-konforme Anwendungen schaffen, die sehr einfach in ein RICHE-basiertes Informationssystem integriert werden können

(*plug & play*). RICHE kann in diesem Zusammenhang durch DHE ersetzt werden, das eine Realisierung der RICHE-Konzepte darstellt.

> [Frandji95] *"Customers can request RICHE-conformant branded products when buying, and will be able to mix and match combinations of RICHE conformant systems and services."*

Was genau muß nun am DHE standardisiert werden? Standardisiert werden muß vor allem das API, das die Kommunikationsschnittstelle zu anderen Anwendungen darstellt. Die Veröffentlichung des API ist für die allgemeine Anwendung und Verbreitung dieser Kommunikationsschnittstelle notwendig. Das Ergebnis der Standardisierungsbemühungen ist der vorläufige CEN-Standard (Comité Européen de Normalisation) *HISA* [CEN97].

HISA (Healthcare Information System Architecture) definiert und standardisiert Middleware-Komponenten, die auf einem für das Gesundheitswesen spezifischem Modell beruhen. Beispiele für HISA-Komponenten sind die DHE-Komponenten. Die HISA-Komponenten werden in einen architektonischen Rahmen namens *HIF* (Healthcare Information Framework) eingebracht, der die Funktionen in drei Schichten einteilt: *Bitways*, *Middleware* und *Applications* [CEN95]. Diese Schichtung entspricht einer Richtlinie für europäische Telekommunikationsnetze gemäß [Bangemann94]. Die Bitways-Schicht liefert Funktionen zum Austausch von Nachrichten (TCP/IP, Remote Procedure Call, Client/Server-Protokolle etc.), wodurch die Verteiltheit der Anwendungen zum Ausdruck kommt. Die Middleware-Schicht umfaßt neben den HISA-Komponenten auch *Generic Common Services* (siehe 3.1.2), die nicht an das Gesundheitswesen gebunden sind. Die Middlewarefunktionen benutzen die Funktionen aus der Bitways-Schicht für den Nachrichtenaustausch. Die Anwendungsschicht umfaßt Anwendungen für Ärzte, Pflegekräfte und andere Benutzer in einem klinischen Umfeld. Die Anwendungen benutzen die gemeinsamen Middlewaredienste für die Integration mit anderen Anwendungen.

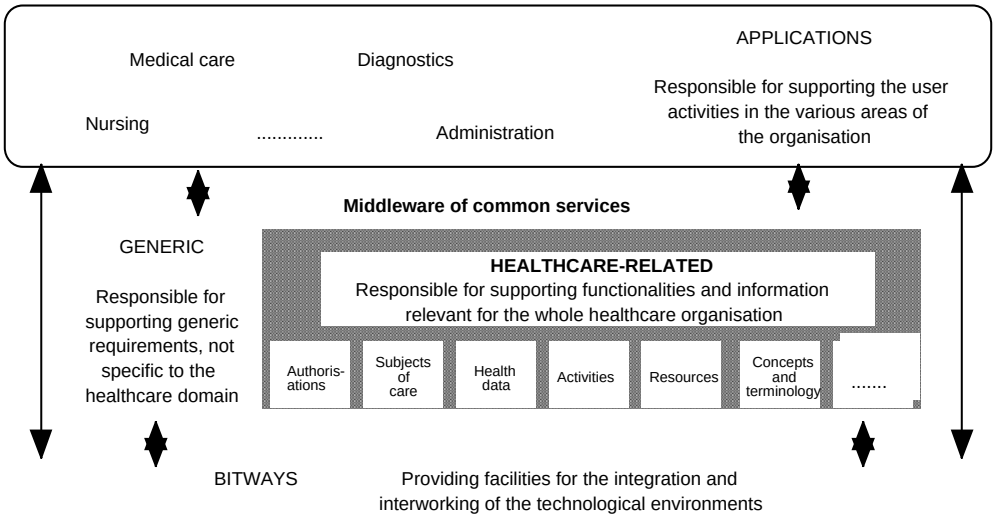

Abbildung 8: Einordnung der HISA-Komponenten (Healthcare-related Common Components) in das HIF (Healthcare Information Framework). Entnommen aus [CEN97]. Das HIF teilt die Krankenhaus-Software in die Schichten Bitways, Middleware und Applications ein. Die Bitways-Schicht umfaßt Funktionen zum Nachrichtenaustausch. Die Middleware-Schicht stellt unter Benutzung der Bitways-Schicht Funktionen bereit, welche die Integration der Anwendungen vereinfachen. Die Middleware-Schicht kann neben den DHE-Komponenten – wie angedeutet - noch weitere Komponenten umfassen. Beispiele für allgemeine Middlewarefunktionen (Generic Common Services) sind in 3.1.2 beschrieben. Die Anwendungsschicht umfaßt klinische Anwendungen, die durch Benutzung der gemeinsamen Middlewaredienste integriert sind.

RICHE, DHE und HISA entsprechen also der Konzeption, der Implementierung und der Standardisierung einer Middleware für das Gesundheitswesen.

3.1.7 Abgrenzung zu anderen Middleware-Ansätzen

In diesem Abschnitt soll das DHE gegen die Middleware-Ansätze *Kommunikationsserver*, *CORBA* und das *Internet* abgegrenzt werden. Die wesentlichen Konzepte dieser Ansätze werden zunächst getrennt vorgestellt und anschließend in einer Tabelle mit dem DHE verglichen.

Kommunikationsserver vermitteln zwischen Sender und Empfänger einer Nachricht. Nachrichten sind in diesem Zusammenhang zur Übertragung bestimmte Kommunikationseinheiten (Anfrage, Antwort, Mitteilung, Dokument etc.). Beispiele für die Vermittlungsdienste eines Kommunikationsservers sind die Verteilung einer Nachricht an viele Anwendungen, die garantierte Zustellung der Nachricht (ggfs. muß die Nachricht wiederholt werden) und die Übersetzung zwischen Nachrichtenformaten [Health-Comm97]. Ein Kommunikationsserver kann auch anhand des Nachrichteninhalts entscheiden, an welche Empfänger die Nachricht weiterzuleiten ist [ROKD98]. Kommunikationsserver beherrschen meist mehrere Nachrichtenstandards (HL7, EDIfACT) und sind mit grafischen Konfigurationswerkzeugen zur Abbildung der Nachrichtenformate und zur Vernetzung der Systeme (Labor, Radiologie etc.) ausgestattet.

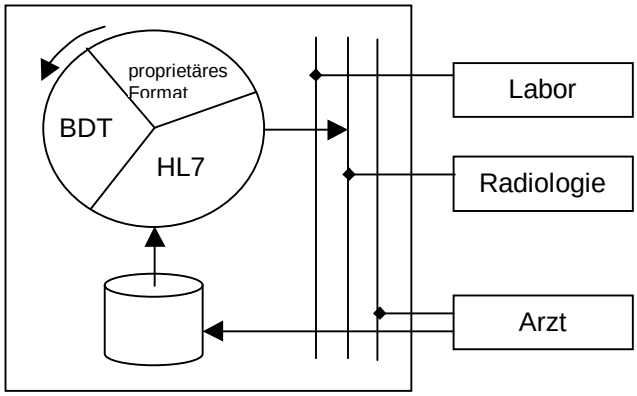

Abbildung 9: Funktionen eines Kommunikationsservers: Speicherung (zur Sicherung und Archivierung), Konversion und Verteilung von Nachrichten.

DHE und Kommunikationsserver unterscheiden sich grundsätzlich in der Art, wie Systeme miteinander verbunden werden. Die Verbindung von Sender und Empfänger ist beim DHE der *gemeinsame Speicher,* beim Kommunikationsserver die *gemeinsame Nachricht* [Schweiger98]. Der Vorteil der Speicherkommunikation liegt in der möglichen Vermeidung von Datenredundanz (Name, Geburtsdatum und Geschlecht eines Patienten werden nur einmal gespeichert) bei gleichzeitiger Einschränkung der

Datenstruktur-Freiheit (Arztsystem A speichert die Behandlung als Teil der Diagnose, Arztsystem B die Diagnose als Teil der Behandlung).

CORBA [www.corba.org, Siegel96] steht für Common Object Request Broker Architecture und handelt von verteilten Objekten. Objekte sind Einheiten aus Daten und Methoden (Funktionen), wobei die Daten eines Objekts ausschließlich über die Objektmethoden zugreifbar sind (*Datenkapselung*). Darüber hinaus können Objekte die Eigenschaften, d.h. die Daten und Methoden anderer Objekte erben und erweitern (*Vererbung*). Die Methoden eines Objekts werden in einer von der Programmiersprache unabhängigen Interface Definition Language (*IDL*) beschrieben. Der Nachrichtenaustausch zwischen den entfernten Objekten erfolgt über das netzwerk-unabhängige Protokoll *GIOP* (General Inter Orb Protocol). Das GIOP definiert die Nachrichten zwischen den Objekten (Request/Response) und codiert die Datentypen in einer Programmiersprache-unabhängigen Common Data Representation (*CDR*). Sogenannte Object Request Broker (*ORB*) machen das GIOP dem Anwendungsentwickler gegenüber transparent.

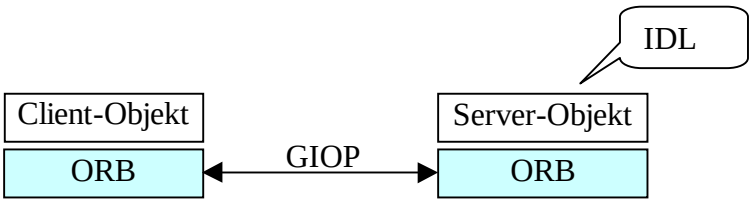

Abbildung 10: CORBA. Der Client muß für die Kommunikation lediglich die IDL-Beschreibung des Servers kennen. Der Nachrichtenaustausch (GIOP) wird durch die ORBs geregelt und ist für den Objekt-Entwickler transparent.

Die Chancen des CORBA-Ansatzes für Informationssysteme im Gesundheitswesen sollen an drei Beispielen verdeutlicht werden. Ein Patient-Objekt kann z.B. durch Erweiterung eines Person-Objekts (Name, Geburtsdatum etc.) um spezielle Merkmale (Allergien) und Methoden (Aufnahme, Entlassung) definiert werden. Das Person-Objekt kann dann für die Definition der Objekte Arzt und Pflegekraft wiederverwendet

werden. Entsprechendes gilt für die Ableitung von organ-spezifischen Tumordokumentationen aus einer Tumor-Basisdokumentation oder die Entwicklung von Abteilungssytemen aus einem generischen Objekt "Order entry and Result reporting". Die Vererbung spart somit Entwicklungsarbeit. Außerdem müssen die einzelnen Abteilungen (Station, Labor etc.) in einem CORBA-System die Verwaltung ihrer Daten nicht an eine zentrale Datenverwaltung wie das DHE abgeben sondern behalten die Kontrolle über ihre Daten, indem sie entsprechende Zugriffsmethoden bereitstellen. In diesem Fall wird die Verteilung und die Kapselung der Objekte ausgenutzt. Schließlich können Suchmaschinen für Objekte bereitgestellt werden. Der Arzt möchte z.B. die Wechselwirkungen zwischen Medikamenten prüfen lassen. Zu diesem Zweck gibt der Arzt zunächst den Suchbegriff „Medikation“ ein. Die Suchmaschine findet ein Medikationsobjekt und kann den Arzt anhand der IDL-Beschreibung des Objekts zur Auswahl einer Methode (Prüfung auf Wechselwirkungen) sowie zur Eingabe der notwendigen Parameter (Medikamente) auffordern, automatisch den entsprechenden Methodenaufruf generieren und das Resultat präsentieren (keine Wechselwirkungen). Man spricht in diesem Zusammenhang auch von „Object shopping“. Wegen der genannten Chancen versucht die CORBAmed ein domänenspezifisches Objektmodell für das Gesundheitswesen zu entwickeln und zu standardisieren [OMG98].

Das *Internet* geht ähnlich wie CORBA von einer Verteilung der Daten und Programme, den sogenannten *Ressourcen*, aus. Diese Ressourcen werden über ein standardisiertes Adressformat, die *URI* (Uniform Resource Identifier), angesprochen. Das Internet umfaßt darüber hinaus mehrere standardisierte Übertragungsprotokolle wie das *HTTP* (HyperText Transfer Protocol), das *SMTP* (Simple Mail Transfer Protocol), das *FTP* (File Transfer Protocol) oder das *IIOP* (TCP/IP + GIOP), die vom Anwender und Entwickler in einer transparenten Art und Weise für den Ressourcenzugriff benutzt werden können. Das HTTP unterscheidet mehrere Datenformate (Textformate, Bildformate, Videoformate) und wird vor allem im *WWW* (World Wide Web) eingesetzt. Das SMTP wird für den Austausch von elektronischer Post (*Email*), das FTP zur Übertragung großer Datenmengen und das IIOP für den *entfernten Programmaufruf* verwendet. Darüber hinaus stellt das Internet für die Anwendungsentwicklung (Benutzerschnittstelle, Datenbankzugriff, Netzwerkzugriff

etc.) zahlreiche APIs (Programmiersprache Java) zur Verfügung. Das Internet zeichnet sich also durch eine Vielfalt an Formaten zur Beschreibung und Übertragung von Ressourcen aus.

	DHE	**KS**	**CORBA**	**Internet**
Was wird integriert (einbezogen)?	Daten	Nachrichten	Objekte	Ressourcen
Definiert ein Modell für das Gesundheits-wesen?	Datenmodell des DHE (3.1.1)	Ein KS liefert kein Modell, unterstützt aber meist standardisierte Nachrichten-modelle wie HL7 oder EDIFACT	Die CORBAmed definiert ein Objektmodell für das Gesundheits-wesens	nein
Middleware?	Organisation, Übertragung und Replikation von Daten (3.1.3)	Verteilung, Sicherung und Konversion von Nachrichten, Vernetzungs-werkzeuge	Brokering (GIOP Transparenz), Verzeichnis der Objekte (Name server)	Zahlreiche Protokolle für den Ressourcen Zugriff
Component-ware?	Manager, Konformanz (3.1.5)	Einzelne Komponente, ansonsten keine CBD-Methoden (siehe 3.1.5)	Objekt-Orientierung, CORBAmed	APIs für zahlreiche Bereiche
Benutzung?	Schwach verbreitet, langsam zunehmend	Stark verbreitet	Schwach verbreitet, langsam zunehmend	Stark verbreitet, rasch zunehmend

Tabelle 3: Vergleich der Middleware-Ansätze DHE, Kommunikationsserver (KS), CORBA und Internet. Die Ansätze decken durchaus verschiedene Bereiche ab und können kombiniert werden. Die umfassendste Unterstützung liefert das Internet, ohne jedoch ein Modell für das Gesundheitswesen bereitzustellen.

3.2 Das Gießener Tumordokumentationssystem (GTDS)

Die Entwicklung des Gießener Tumordokumentationssystems *GTDS* begann 1991, nachdem im Vorjahr die Basisdokumentation für Tumorkranke revidiert wurde und ein neues System für die Erfassung dieser Daten bereitgestellt werden sollte. Das System befindet sich heute in über 30 deutschen Tumorzentren im Einsatz, vor allem in den neuen Bundesländern. Es ist für den Einsatz in Registern konzipiert, bietet aber auch zahlreiche Funktionen zur Unterstützung klinischer Abläufe. Bei der Konzipierung haben zahlreiche Fachleute aus bestehenden Registern mitgewirkt.

3.2.1 Das Modell: Informationen im GTDS

Analog zum DHE soll das GTDS zunächst anhand seines Datenmodells vorgestellt werden. Ziel des GTDS ist die Unterstützung der Tumordokumentation. Das Modell des GTDS umfaßt dementsprechend Konzepte (Entitäten) wie den *Patienten*, den *Tumor* und die *Metastase*, die ihrerseits durch weitere Attribute (*Name* des Patienten, *TNM*-Klassifikation des Tumors etc.) beschrieben sind. Das folgende Diagramm zeigt einen kleinen Ausschnitt aus dem GTDS-Modell, indem einige Konzepte und Konzeptbeziehungen dargestellt sind.

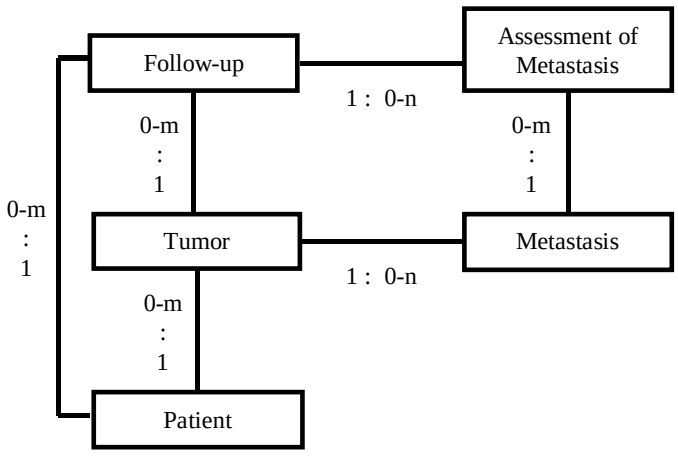

Abbildung 11: Auszug aus dem GTDS-Modell. Entnommen aus [Bürkle99]. Zu einem Tumor gehören 0 bis beliebig viele (m, n) Metastasen und 0 bis beliebig viele Nachsorgeuntersuchungen.

3.2.2 Funktionen des GTDS

Das GTDS liefert unter anderem folgende Funktionen [Altmann95, www.med.uni-giessen.de/akkk]:

1) Dokumentation von Diagnose, Therapie, Verlauf und Abschluß von Tumorkrankheiten (Basisdokumentation und organspezifische Erweiterungen)
2) Datenaustausch mit Kliniken, Arztpraxen und anderen Registern über Nachrichtenstandards wie BDT oder HL7
3) Auswahl von Patienten für klinische Studien
4) Unterstützung der Nachsorge durch Erinnerung von Arzt und Patient an durchzuführende Untersuchungen bzw. Maßnahmen
5) Erzeugung von Arztbriefen und Übersichtsberichten
6) Berechnung der individuellen Medikation bei Chemotherapie

Die Funktionen (1) und (2) sichern die vollständige Dokumentation eines Tumors und leisten so einen wichtigen Beitrag für die Langzeitbeobachtung von Tumorpatienten. Die Verfügbarkeit der Krankheitsgeschichte ist bei langwierigen und interdisziplinär behandelten Krankheiten wie dem Krebs von besonderer Bedeutung. Sie gibt Aufschluß über den Erfolg von Therapien und vermeidet die unnötige Wiederholung von patient-belastenden Untersuchungen. Die Funktion (4) gewährleistet eine regelmäßige Kontrolle des Tumorstatus und generiert entsprechende Hinweise (erzeugt automatisch Briefe), wenn lange Zeit keine Kontrolluntersuchung stattgefunden hat. Die Funktion (6) erlaubt die Berechnung der an Gewicht, Oberfläche und Zustand des Patienten angepaßten Dosisangaben auf der Basis von Chemotherapie-Protokollen.

3.2.3 Abgrenzung zum DHE

Das DHE und das GTDS unterscheiden sich in ihren Datenmodellen und in ihrer Funktionalität. Das DHE-Modell umfaßt nur sehr allgemeine Konzepte des Gesundheitswesens wie z.B. den Medizindatentyp und das Medizindatum (siehe 3.1.1). Das GTDS-Modell hingegen enthält sehr konkrete Konzepte wie den Tumor oder die Metastase. Auf der anderen Seite können im DHE neue Medizindatentypen definiert werden, d.h. es können ohne Veränderung des Modells Konzepte wie der Tumor oder die Metastase hinzugefügt werden. Das Modell des DHE ist also einfacher anzupassen als das Modell des GTDS. Der Unterschied in den Datenmodellen kann durch die unterschiedlichen Zielsetzungen von DHE und GTDS begründet werden.

Das DHE ist eine Middleware und dient der Integration von beliebigen klinischen Anwendungen. Als *Integrationsmodell* muß das DHE-Modell viele Bereiche abdecken und richtet sich vorrangig an den *Entwickler*. Das DHE verfügt dementsprechend über eine Entwicklerschnittstelle (API). Das GTDS hingegen ist ein Anwendungssystem für die Tumordokumentation. Als *Anwendungsmodell* muß das GTDS-Modell die Konzepte der Tumordokumentation kennen, um den *Anwender* (Dokumentationspersonal, Arzt) angemessen unterstützen zu können. Das GTDS verfügt dementsprechend über eine Anwenderschnittstelle. Das GTDS kann zum Beispiel bei der Datenerfassung automatisch die Verträglichkeit der Tumorlokalisation und Tumorhistologie mit dem Alter und Geschlecht des Patienten überprüfen (Integritätsbedingungen gemäß IARC (International Agency for Research on Cancer)). Diese Anwendungsfunktion muß unter anderem Kenntnis vom Aufbau eines Tumorerfassungsbogens sowie der Codierung der Tumorlokalisation und -histologie haben.

4 Methodik zur Integration von GTDS und DHE

Nachdem in Kapitel 1 die gegebenen Systeme GTDS und DHE getrennt vorgestellt wurden, soll nun die angestrebte Verbindung von GTDS und DHE beschrieben werden. In diesem Kapitel wird also beschrieben, wie die gestellte Integrationsaufgabe gelöst wurde. Was soll diese Verbindung nun genau leisten? Gemäß Zielsetzung sollen DHE-basierte Anwendungen - zum Beispiel die von Magdeburg bereitgestellte Arztanwendung – lesend auf die Informationen im GTDS zugreifen können. DHE-basierte Anwendungen haben i.a. keine Kenntnis vom GTDS. Die Verbindung von GTDS und DHE muß demnach im weitesten Sinne die Übertragung der Information vom GTDS zum DHE leisten.

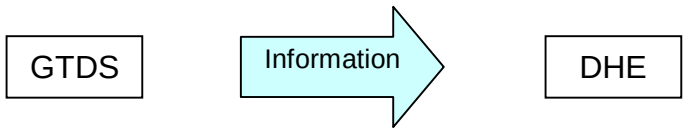

Abbildung 12: Die Verbindung leistet im weitesten Sinne die Übertragung der Informationen vom GTDS zum DHE.

Die Übertragung der Informationen vom GTDS zum DHE bedingt die Übersetzung des GTDS-Modells in das DHE-Modell. Zwei Wege der Modellübersetzung waren grundsätzlich möglich:

- Verwendung des Act-Konzepts (3.1.1)
- Verwendung eines Nachrichtenstandards

Die folgenden beiden Abschnitte beschreiben die beiden Wege und begründen unsere Entscheidung für den zweiten Weg.

4.1 Fehlansatz: Integration über das Act-Konzept

In Abschnitt 3.1.1 wurde der *Act* (Versorgung des Patienten, Maßnahme) als das zentrale Konzept im DHE-Modell hervorgehoben. Der erste Gedanke war daher, das Act-Konzept für die Integration von GTDS und DHE zu benutzen. Das DHE – genauer die DHE-Datenbank – wird zu diesem Zweck mit einem Act-Typ „GTDS Anfrage“ und einem Medizindatentyp „GTDS Antwort“ konfiguriert. Die GTDS-Antwort wird je nach Bedarf durch weitere Medizindatentypen wie Lokalisation, Histologie und TNM-Schlüssel strukturiert.

Der folgende Anwendungsfall illustriert den Verbindungsbetrieb zwischen DHE und GTDS. Der Arzt sucht Informationen zu einem Tumorpatienten. Das Arbeitsplatzsystem des Arztes, d.h. ein arztseitiger DHE-Client erzeugt für die Informationsanfrage in der DHE-Datenbank einen Act vom Typ GTDS-Anfrage. Ein GTDS-seitiger DHE Client schaut von Zeit zu Zeit im DHE nach, ob Anfragen für das GTDS anliegen und beantwortet diese gegebenenfalls. Zu diesem Zweck liest der GTDS-seitige Client die Informationen aus der GTDS-Datenbank und schreibt sie als GTDS-Antwort (Medizindaten vom Typ GTDS-Antwort) in die DHE-Datenbank. Der arztseitige Client kann nun die Informationen aus dem DHE lesen und dem Arzt präsentieren.

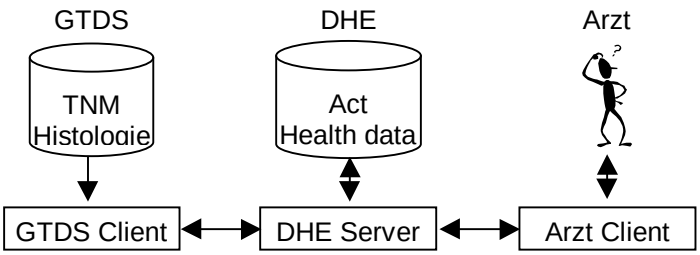

Abbildung 13: Verbindung von GTDS und Arztanwendung über das Act-Konzept. Der arztseitige DHE Client schreibt einen Act vom Typ GTDS-Anfrage in das DHE. Der GTDS-seitige DHE-Client liest die Anfrage aus dem DHE, holt die angeforderten Informationen aus dem GTDS und schreibt sie als Medizindaten vom Typ GTDS-Antwort in das DHE. Der arztseitige Client kann nun die Informationen dem Arzt präsentieren.

Der Ansatz wurde aus konzeptuellen Gründen verworfen. Konzeptuell repräsentiert der Act einen Zeitverlauf und ist durch einen Act life cycle charakterisiert. Der Arzt sollte allerdings nicht auf die Informationen warten müssen. Die Verbindung von GTDS und DHE ist eine Kommunikation zwischen Programmen und sollte idealerweise sogar zeitlos erfolgen, was der Beschreibung durch einen Life cycle widerspricht.

In dem vorgestellten Ansatz wird eine direkte Übersetzung des GTDS-Modells in das DHE-Modell vorgenommen. Die Modellübersetzung erfolgt dabei in dem GTDS-seitigen DHE-Client. Über eine solche Verbindung kann nur das GTDS an das DHE angebunden werden. Der folgende Abschnitt beschreibt einen Weg, der bei vergleichbarem Entwicklungsaufwand die Anbindung mehrerer Systeme an das DHE ermöglicht.

4.2 Erfolgsansatz: Integration über den Nachrichtenstandard BDT

Viele existente DV-Systeme verfügen über die Möglichkeit, Informationen in Form von Nachrichten auszutauschen. Das GTDS zum Beispiel verwendet für den Austausch von Tumordaten den Nachrichtenstandard *BDT* (BehandlungsDatenTräger), der an anderer Stelle in diesem Abschnitt beschrieben wird. Solche BDT-Nachrichten können auch für die Verbindung des GTDS mit dem DHE genutzt werden. Dabei wird das GTDS-Modell zunächst in das BDT-Modell übersetzt, und dann das BDT-Modell in das DHE-Modell. Die Übersetzung des GTDS-Modells in das BDT-Modell wird vom GTDS geleistet. Die Verbindung des GTDS mit dem DHE „reduziert“ sich somit auf die Übersetzung des BDT-Modells in das DHE-Modell. Der Vorteil der indirekten Modellübersetzung liegt nun darin, daß über die zu entwickelnde BDT-DHE-Schnittstelle auch andere BDT-exportierende Systeme an das DHE angebunden werden können.

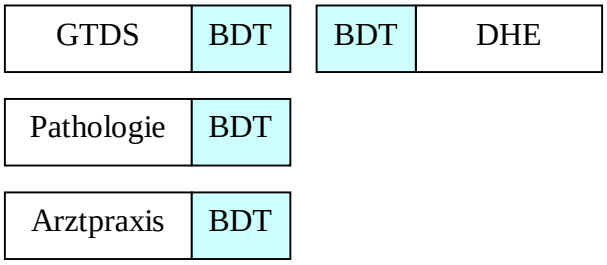

Abbildung 14: Verbindung des GTDS mit dem DHE über den Nachrichtenstandard BDT. Die zu entwickelnde BDT-Schnittstelle zum DHE kann für die Anbindung weiterer BDT-exportierender Systeme (Pathologiesystem, Arztpraxissystem) verwendet werden.

Der BDT (Behandlungsdatenträger) ist ein deutscher Standard zum Austausch von Behandlungsdaten zwischen Praxiscomputersystemen und wird vom Zentralinstitut für Kassenärztliche Versorgung (ZI) gepflegt. Ein besonderer Vorteil des BDT ist die explizite Kennzeichnung der Inhalte innerhalb des übertragenen Datensatzes. Der BDT definiert also keine starre Struktur mit festen Feldpositionen. Auf diese Weise können

neue Inhalte in den BDT eingeführt werden, ohne daß gleich alle BDT-austauschenden Programme angepaßt werden müssen. Das Kennzeichnungsprinzip verleiht dem BDT eine hohe Flexibilität bei wechselnden Inhalten klinischer Daten. Da Praxissysteme einen potentiellen Kommunikationspartner für klinische Register darstellen, wurde auch für die Tumordokumentation ein BDT-Standard formuliert [Altmann98].

Die Übertragungseinheit innerhalb des BDT-Transfers wird Datenpaket genannt. Ein Datenpaket besteht aus Sätzen, die medizinische (z.B. Behandlungsdaten) und organisatorische (z.B. Absender) Informationen enthalten. Jeder BDT-Satz besteht aus mehreren Feldern. Die ersten beiden Felder eines Satzes enthalten Informationen zu Satzart und Satzlänge. Darauf folgen Felder mit den eigentlichen Inhalten, meist beginnend mit dem Feld Patientennummer. Jedes Feld hat folgende Struktur: 3 Zeichen Feldlänge, 4 Zeichen Feldkennung, n Zeichen Feldinhalt und 2 Zeichen Feldabschluß (nicht sichtbares Carriage Return und Line Feed). Die Feldlänge errechnet sich folglich zu 3 + 4 + n + 2 = 9 + n Zeichen.

Die Felder innerhalb eines Satzes können zu Gruppen zusammengefaßt und als Ganzes wiederholt werden. Jedes Feld bekommt dazu eine implizite Ebene, die der BDT-Dokumentation zu entnehmen ist. Felder der Ebene 2 gehören zu Feldern der Ebene 1. Der BDT unterscheidet bis zu fünf Hierarchieebenen. Dadurch können beispielsweise einem Patienten (1. Ebene) mehrere Behandlungstermine (2. Ebene) und jedem Behandlungstermin die erhobenen Daten (3. Ebene) zugeordnet werden. Die folgende Abbildung zeigt einen Ausschnitt aus der BDT-Dokumentation. Man kennzeichnet das Feld *Chemotherapie Intention* mit 7880 und das Feld *Chemotherapie Behandlungsbeginn* mit 7881. Ferner erhält das Feld 7880 die Ebene 1 und das Feld 7881 die Ebene 2. Dadurch ist festgelegt, daß zu einer Chemotherapie Intention eine oder mehrere Chemotherapien gehören. Hat der Patient zwei Chemotherapien, so kann das durch die Feldfolge 7881, 7882 ... , 7881, 7882 ... ausgedrückt werden. Dabei gehört das zweite Behandlungsende 7882 zum zweiten Behandlungsanfang 7881, da die aufsteigende Ordnung der Feldkennungen nur bei Wiederholung gestört werden darf.

Kenn.	Ebene 1 2 3 4 5	Beschreibung	Inhalt
7880	1	Chemotherapie Intention	a = adjuvant k = kurativ n = neoadjuvant p = palliativ x = unbekannt
7881	n	Chemotherapie Behandlungsbeginn	TTMMJJJJ
7882	1	Chemotherapie Behandlungsende	TTMMJJJJ
7883	1	Chemotherapie Protokoll	Freitext; übliche Abkürzung des Protokollnamens
3622	1	Größe des Patienten	cm
3623	1	Gewicht des Patienten	kg
7884	n	Chemotherapie Medikament	Name#Pharmazentralnr.
7885	1	Chemotherapie Medikament Zyklusdosis	
7886	1	Chemotherapie Medikament Zyklusdosis Einheit	

Abbildung 15: Auszug aus der BDT-Dokumentation. Die Spalten enthalten von links nach rechts die Kennung, die Ebene, die Beschreibung und mögliche Werte der Felder. Die Ebene ist durch Einrückung gekennzeichnet. Ein n bedeutet, daß sich das Feld (Chemotherapie Medikament) als Ganzes, d.h. inklusive aller untergeordneten Felder (Zyklusdosis samt Einheit) wiederholen darf.

Abschließend soll noch ein einfaches Beispiel für eine BDT-Nachricht gegeben werden. Zur besseren Lesbarkeit wird die Feldkennung unterstrichen und die Interpretation des Feldes (Feldbeschreibung) am rechten Rand angegeben. Längere Texte wie z.B. der Diagnosetext 7709 werden unter Umständen in mehrere Zeilen aufgebrochen:

01380006100	Satzart Patientenstamm
01330001242	Patient Identifikation 1242
01633101Alphorn	Name Alphorn
01433102Heidi	Vorname Heidi
01731028101933	Geburt 28. Oktober 1933
02633106Frankfurt am Main	Ort ...
02433107Auf der Wiese 5	Strasse ...
01033110W	Geschlecht weiblich
01380006201	Satzart Tumordokumentation
01330001242	Patient Identifikation 1242
0107770 01	Tumor Identifikation 1
01777012 0031967	Diagnosedatum 20. März 1967
03477709Mamma-Ca. intraduktal re.	Diagnosetext ...
02777709unt. äuß. Quadrant	Diagnosetext ...
03977709Befall ax. LK Stadium T2 N1 M0	Diagnosetext ...
01380006201	Satzart Tumordokumentation
01330001242	Patient Identifikation 1242
0107770 02	Tumor Identifikation 2
02477709Mamma-Ca. links	Diagnosetext Mamma-Ca. links

Abbildung 16: Beispiel einer BDT-Nachricht. Die Feldkennungen sind zur besseren Lesbarkeit unterstrichen. Auf die Kennung folgt der Feldinhalt. Am Ende der Zeile ist eine Feldbeschreibung angegeben, die nicht Bestandteil der Nachricht ist. Vor der Kennung ist die Anzahl der Zeichen auf einer Zeile inklusive der nicht-sichtbaren Zeichen Carriage return und Line feed angegeben.

4.3 Grobverfahren und Aufwandseinschätzung

Nachdem wir uns auf die Verwendung des Nachrichtenstandards BDT für die Verbindung von GTDS und DHE festgelegt haben, soll in diesem Abschnitt das grobe Vorgehen bei der Implementierung dieser Lösung beschrieben werden. Da für Demonstrationszwecke nicht der komplette BDT-Standard berücksichtigt werden mußte, trafen wir zunächst eine Auswahl der zu übertragenden Informationen. 17 BDT-Felder sollten zunächst vom GTDS zum DHE übertragen werden:

3000 Patientenkennung
3101 Name des Patienten
3102 Vorname des Patienten
3103 Geburtsdatum des Patienten (Basisdok. 1.3)
3106 Wohnort des Patienten
3107 Straße des Patienten
3110 Geschlecht des Patienten (Basisdok. 1.2)
3201 Name des Hauptversicherten
3203 Geburtsdatum des Hauptversicherten
7700 TumorID
7701 Tumordiagnosedatum (Basisdok. 2.6)
7709 Diagnosetext
7715 Freitext Lokalisation
7725 Freitext Histologie
7753 TNM T-Kategorie (Basisdok. 2.14.2)
7757 TNM N-Kategorie (Basisdok. 2.14.2)
7760 TNM M-Kategorie (Basisdok. 2.14.2)

Abbildung 17: In das DHE zu importierende BDT-Felder. Zu jedem BDT-Feldschlüssel ist eine BDT-Feldbeschreibung angegeben.

Im folgenden wird skizziert, was für die Konfiguration und den Betrieb des DHE zu tun ist, damit BDT-Nachrichten in das DHE importiert werden können. Die Aufgaben der Konfiguration und des Betriebs werden jeweils auf einer Modellebene (siehe dazu auch 3.1.1) und einer Programmebene beschrieben.

Da das DHE die Konzepte Tumordiagnose, Histologie und TNM (Tumor Nodes Metastases ist ein Klassifikationssystem für Tumorkrankheiten) nicht kennt, muß es zunächst konfiguriert werden. Gemäß DHE-Modellausschnitt 3.1.1 können zu einem Patienten sogenannte Medizindaten gespeichert werden. Die Konfiguration des DHE besteht folgerichtig in der Anlage entsprechender Medizindatentypen. Auf Modellebene entspricht eine BDT-Feldbeschreibung also einem DHE-Medizindatentyp.

Da die Beschreibung der BDT-Felder bereits in elektronischer Form (eine gemäß Abbildung 17 strukturierte Textdatei) verfügbar ist, muß im wesentlichen ein Programm geschrieben werden, das die BDT-Feldbeschreibungen als Medizindatentypen in das DHE importiert. Ein solches Programm soll im folgenden als Importprogramm oder kurz *Importer* bezeichnet werden. Die Konfigurationsaufgabe läßt sich insgesamt wie folgt darstellen:

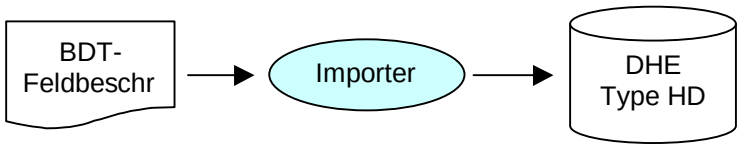

Abbildung 18: Konfiguration des DHE. Es ist ein DHE-Importprogramm zu entwickeln, das BDT-Feldbeschreibungen in DHE-Medizindatentypen (Type of health data) überführt.

Mit dieser Konfiguration des DHE können nun die Informationen der BDT-Nachricht in das DHE übertragen werden. Die BDT-Patientenstammdaten können im DHE auf die Attribute der Patient-Entität abgebildet werden. Für den BDT-Histologiebefund wird im DHE ein Medizindatum vom Typ Histologie angelegt. Die übrigen Informationen der BDT-Nachricht werden analog zum Histologiebefund behandelt.

Analog zu den Konfigurationsdaten wird wieder ein Importprogramm benötigt, das die BDT-Daten in das DHE überträgt. Im Unterschied zu den Konfigurationsdaten ergibt sich bei den laufenden Daten aber eine zusätzliche Forderung: Da dieselben Informationen nun in mehr als einem System (GTDS und DHE) gespeichert werden (Replikation), müssen bei einer Veränderung der Information stets alle Systeme aktualisiert werden (*Replikationskontrolle*). In unserem Fall wird lediglich eine Replikationskontrolle vom GTDS zum DHE benötigt, da auf die Tumordaten im DHE

nur lesend zugegriffen wird. Im GTDS dagegen können die Daten geändert werden. Abbildung 19 zeigt die Laufzeitmodule der GTDS-DHE-Verbindung und skizziert ihr Zusammenwirken. Auf dem Weg vom GTDS zum DHE durchlaufen die Informationen den *Exporter*, den *Listener* und den *Importer*, deren Funktionen im folgenden beschrieben werden.

Der *Exporter* liest die Informationen aus dem GTDS, verpackt sie in einer BDT-Nachricht und schreibt die Nachricht mittels FTP (File Transfer Protocol) in ein DHE-seitiges Verzeichnis. Der Exporter wird über entsprechende Trigger der GTDS-Datenbank aktiviert. Datenbank-Trigger überwachen Ereignisse in der Datenbank wie die Änderung bestimmter Daten und stoßen ggfs. weitere Aktivitäten, in unserem Fall den Exporter, an. Trigger und Exporter wissen, welche Informationen im GTDS für das DHE relevant sind.

Der *Listener* schaut von Zeit zu Zeit im Verzeichnis nach, ob BDT-Nachrichten angekommen sind, aktiviert für jede angekommene Nachricht den Import in das DHE und entfernt die Nachricht anschließend aus dem Verzeichnis. Das Verzeichnis fungiert in diesem Zusammenhang als Sammelbecken und Zwischenspeicher für Aktualisierungsnachrichten.

Der *DHE-Importer* packt die Informationen wieder aus der BDT-Nachricht aus und speichert sie im DHE.

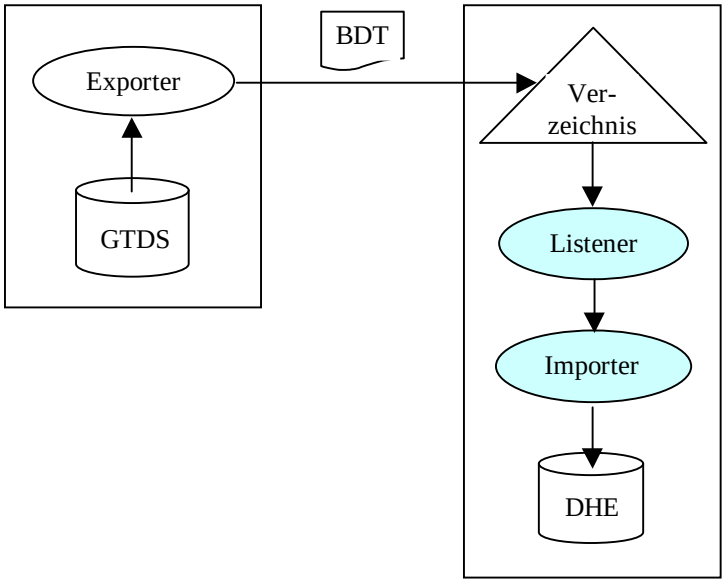

Abbildung 19: Laufzeitmodule der GTDS-DHE-Verbindung. Der Exporter übersetzt das GTDS-Modell in das BDT-Modell. Der Importer übersetzt das BDT-Modell in das DHE-Modell. Die Replikationskontrolle erfolgt über Trigger der GTDS-Datenbank und den Listener. Die grauen Module sind noch zu entwickeln, wobei der wesentliche Entwicklungsaufwand beim Importprogramm liegt.

Insgesamt sind für die Verbindung vom GTDS zum DHE also drei Module zu entwickeln: ein Programm für die Konfiguration des DHE, ein Programm für den Import von BDT-Nachrichten in das DHE und ein Modul zur Kontrolle der Replikation zwischen GTDS und DHE (Listener). Die Replikationslogik ist mit wenigen Betriebssystemkommandos realisierbar, sodaß der wesentliche Entwicklungsaufwand bei den DHE-Importprogrammen liegt. Die Entwicklung von Importprogrammen scheint eine wiederkehrende Tätigkeit zu sein, die nach einer maschinellen Lösung sucht. Eine solche Lösung wird in den folgenden Abschnitten erarbeitet. Die DHE-Importprogramme werden in diesem Zusammenhang auch als *Schnittstellen* bezeichnet. Eine Schnittstelle ist unter anderem wie folgt definiert:

> [Duden88, 517-518] *Die Schnittstelle (engl. interface) eines Systems beschreibt alle von außen benötigten (Importschnittstelle) und alle von außen abrufbaren (Exportschnittstelle) Größen.*

Das System entspricht in unserem Fall dem DHE und die Größen entsprechen den Informationen bzw. Daten. Darüber hinaus wollen wir die Schnittstelle nicht nur als eine Beschreibung ansehen, sondern auch als ein Programm, das konkrete Daten in das DHE importiert.

4.4 Das Schlüsselkonzept: Die Generierung von Schnittstellen

Das Grobverfahren hat deutlich gemacht, worin der Aufwand bei der Anbindung existenter Systeme an das DHE besteht. Die Problematik äußert sich in der folgenden Fragestellung: Wie können gegebene Daten ohne viel Aufwand im DHE repliziert werden? Selbst bei einem so begrenzten Vorhaben wie dem GTDS-DHE-Demonstrator sind schon mindestens zwei Importprogramme nötig. Das Schlüsselkonzept liegt nun darin, diese Importprogramme von einem weiteren Programm erzeugen zu lassen [Schweiger97]. Ein solches Programm wird als *Generator* bezeichnet:

> [IEEE97] *"Generator: A mechanism that creates assets* (DHE-Importprogramme) *by inclusion of information that is internal to the generator"*

Der Nutzen des Generators liegt in der „*information that is internal to the generator*“, d.h. dem vom Entwickler zum Generator verlagerten Wissen. Worin konkret besteht nun dieses interne, d.h. dem Entwickler gegenüber transparente Wissen? Gemeint ist im weitesten Sinne die Programmierung. Der Entwickler sollte sich zum Beispiel nicht mehr mit dem API des DHE auseinandersetzen müssen (API-Transparenz). Der Entwickler muß also nicht mehr wissen, welche API-Funktionen es gibt, wie die Funktionen parametrisiert sind, welche Datentypen die Parameter haben und wie die Eingabeparameter initialisiert werden.

Der Generator kann dagegen nicht wissen, welchem Datenmodell die zu importierenden Daten folgen und wie der Entwickler dieses Importmodell auf das DHE-Modell abbilden möchte. Die Abbildung zwischen dem Importmodell und dem DHE-Modell wird dazu in einer als *Map* bezeichneten Datei untergebracht, die als Eingabe für den Generator dient:

> *Map: Abbildung des Importmodells (Modell der in das DHE zu importierenden Daten) auf das DHE-Modell.*

Die Beziehung zweier Datenmodelle soll im weiteren auch als *Schnittstellenmodell* bezeichnet werden. Das von uns erarbeitete Schlüsselkonzept zum Import gegebener Daten in das DHE wird in der folgenden Übersichtsskizze noch einmal zusammengefaßt. Map und Generator werden in den folgenden Abschnitten ausführlich beschrieben.

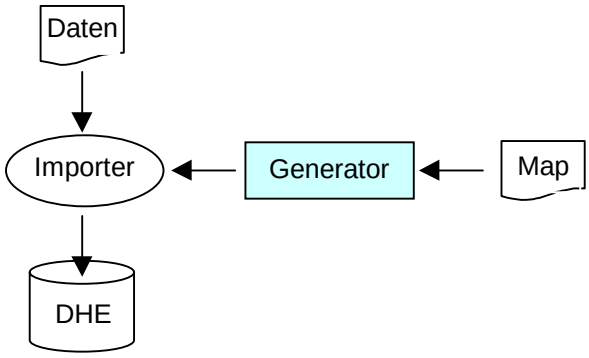

Abbildung 20: Das Generatorkonzept. In der Map wird das Importmodell, d.h. das Datenmodell der in das DHE zu importierenden Daten, auf das DHE-Modell abgebildet. Die Map enthält somit das Schnittstellenmodell. Aus der Map erzeugt der Generator dann ein entsprechendes Schnittstellenprogramm (Importer), das die Daten in das DHE importiert. Der Generator hebt also die Schnittstellenentwicklung von der Daten- und Programmebene auf die Modellebene an.

Mit dem Generator wird sowohl eine Middleware als auch eine Componentware geschaffen. Der Generator untersützt die Anbindung existenter Systeme an das DHE

und damit auch indirekt die Integration der Systeme untereinander. Insofern ist der Generator eine Middleware (siehe 3.1.2). Als wiederverwendbar konzipiertes Modul ist der Generator auch eine Componentware (siehe 3.1.5).

4.5 Die Map: Inhaltliche Abbildung zwischen BDT und DHE

Gemäß 4.4 teilt die Map dem Generator mit, wie das Importmodell auf das DHE-Modell abzubilden ist. Die wesentlichen Elemente einer Map sollen nun am Beispiel der BDT-Map erläutert werden.

4.5.1 Semantische Elemente einer Map

Um das Importmodell auf das DHE-Modell abbilden zu können, müssen die Elemente in beiden Modellen zunächst bezeichnet werden. Es werden also *Bezeichner* für die Modellelemente benötigt. Für die Bezeichnung der BDT-Felder bietet sich die Feldkennung an (siehe 4.2), die man der BDT-Beschreibung entnimmt. Die Feldkennung für das Geschlecht des Patienten ist zum Beispiel *3110*. Die Bezeichnung der DHE-Elemente entnimmt man dem DHE-Modell 3.1.1. Die DHE-Bezeichner haben die Gestalt *managerENTITY.attribute*, wobei *manager* die DHE-Komponente, *ENTITY* die Entität in der Komponente und *attribute* das Attribut der Entität bezeichnet. Bei dem Bezeichner *pmPA.pa_sesso* handelt es sich zum Beispiel um das Geschlecht *pa_sesso* (italienische Vokabel) des Patienten *PA*, der zum Patient Manager *pm* gehört. Möchte man auf die ganze Entität, d.h. alle Attribute Bezug nehmen, so läßt man den Punkt und das Attribut beim DHE-Bezeichner weg. Die komplette Patient-Entität wird also mit *pmPA* angesprochen. Synonym zu Bezeichner wird auch der Begriff *Tag* (engl. Etikett) verwendet.

Ein weiterer Bestandteil einer Map sind die *Zuordnungen*. Zuordnungen stellen Beziehungen zwischen den Elementen des Importmodells und den Elementen des DHE-Modells her. Im Falle des Patientengeschlechts kann die Abbildung zwischen BDT und DHE wie folgt ausgedrückt werden:

3110(pmPA.pa_sesso) oder auch
pmPA.pa_sesso = 3110

Zuordnungen verbinden demnach das Importmodell mit dem DHE-Modell.

Verweise zwischen den DHE-Entitäten können in der Map wie folgt formuliert werden. Angenommen, man möchte zu einem gegebenen Patienten einen Befund im DHE abspeichern. Im DHE-Modell wird ein Befund mit *hmHE* (HEalth data im health data manager) bezeichnet. Innerhalb der *hmHE*-Entität erfolgt ein Verweis auf den Patienten über das Attribut *he_ipati*. Der Patientenbezug des Befundes kann dann in der Map durch den Ausdruck hergestellt werden:

hmHE.he_ipati = pmPA

Im Unterschied zu den Zuordnungen repräsentieren Verweise also Beziehungen innerhalb des DHE-Modells.

Vor der Zuordnung von Elementen des Importmodells zu Elementen des DHE-Modells müssen diese unter Umständen noch richtig granuliert (zerlegt oder zusammengefaßt) und/oder codiert werden. Zu diesem Zweck können *Konversionsfunktionen* in der Map benutzt werden. Ein Beispiel für eine Granulierung ist die Zerlegung der BDT-Straßenangabe (Bezeichner *3107*) in den Straßen-Namen und die Hausnummer. Man kann dazu eine Konversionsfunktion *hausnummer* definieren, die aus der BDT-Strassenangabe die Hausnummer extrahiert. Anschließend kann die Hausnummer dem entsprechenden DHE-Attribut zugeordnet werden:

pmPA.pa_numb2 = hausnummer(3107)

Ein Beispiel für eine Umcodierung von BDT nach DHE ist das Geschlecht des Patienten (BDT: *3110*, DHE: *pmPA.pa_sesso*). Man wird z.B. eine Konversionsfunktion *geschlecht* definieren, die das weibliche Geschlecht von *W* nach *F* umcodiert. Damit kann die Zordnung von BDT zu DHE erfolgen:

pmPA.pa_sesso = geschlecht(3110)

Konversionsfunktionen ermöglichen die 1-1-Zuordnung von Elementen des Importmodells zu Elementen des DHE-Modells und sind insofern für die

Modellübersetzung relevant. Konversionsfunktionen sind daher ein Bestandteil des Schnittstellenmodells, d.h. der Map.

Häufig enthalten die Importdaten nur soviel Information, um bereits existente Informationen in der DHE-Datenbank zu identifizieren. Ein übliches Beispiel ist die bloße Angabe der Patientennummer für die Identifikation der bereits gespeicherten Patientendaten. Zu diesem Zweck kann in der Map für jede Entität im DHE (Patient, Befund etc.) definiert werden, welche inhaltlichen Merkmale bzw. Attribute zur Identifikation der Entität herangezogen werden sollen. Das folgende Konstrukt definiert zum Beispiel die Identifikation der Patient-Entität *pmPA* durch die Patientennummer *pa_iperm*:

pmPA[pa_iperm)

Solche Konstrukte sollen als *inhaltliche Schlüssel* bezeichnet werden, die im Unterschied zu den Systemschlüsseln (siehe 3.1.1) nicht vom DHE, sondern vom Schnittstellenentwickler definiert werden. Die Abbildung des inhaltlichen Schlüssels auf den Systemschlüssel erfolgt durch den Generator und ist für den Entwickler transparent. Wird kein inhaltlicher Schlüssel definiert, so wird ohne vorherige Existenzprüfung ein neuer Datensatz (Systemschlüssel) angelegt.

Inhaltliche Schlüssel können auch mehrere Attribute umfassen. Ein Beispiel dafür ist der folgende Schlüssel, der die Identifikation des Patienten durch das Geburtsdatum *pa_birth*, das Geschlecht *pa_sesso* und den Nachnamen *pa_lname* definiert:

pmPA[pa_birth, pa_sesso, pa_lname]

Durch die abschließende Klammer (rund bzw. eckig) wird ausgedrückt, wie die identifizierte Entität verwendet werden darf, für einen Verweis oder für eine Aktualisierung. Wir unterscheiden dementsprechend zwischen *Verweisschlüssel* (runde Klammer) und *Aktualisierungsschlüssel* (eckige Klammer). Für die Aktualisierung einer Entität müssen in der Regel mehr Attribute gegeben sein als für die bloße Identifikation. Für die Aktualisierung der Patientendaten im DHE müssen zum Beispiel neben der Patientennummer noch das Geburtsdatum, das Geschlecht und der

Nachname des Patienten angegeben werden, obgleich die Patientennummer bereits identifizierend ist.

Konstrukt	**Beschreibung**	**Beispiele**
Bezeichner (Tags)	Bezeichnung der Modellelemente	3110 (Geschlecht des Patienten in der BDT-Nachricht), pmPA.pa_sesso (Geschlecht des Patienten in der DHE-Datenbank)
Zuordnungen	Beziehungen zwischen dem Importmodell und dem DHE-Modell	3101(pmPA.pa_lname) oder pmPA.pa_lname=3101 (Nachname des Patienten)
Verweise	Beziehungen innerhalb des DHE-Modells	hmHE.he_ipati=pmPA (Zuordnung eines Befundes zum Patienten)
Konversionsfunktionen	Umwandlung von Importdaten vor der Speicherung im DHE	Extraktion der Hausnummer aus der BDT-Strassenangabe, Umcodierung des Datums vom BDT-Format TTMMJJJJ in das DHE-Format JJJJ/MM/TT
Inhaltliche Schlüssel	Identifikation von Datensätzen (Entitäten) im DHE anhand inhaltlicher Merkmale (Attribute)	pmPA[pa_lname, pa_birth) Identifikation des Patienten anhand seines Nachnamens und seines Geburtsdatums. Die abschließende Klammer (rund, eckig) bestimmt die Verwendbarkeit der identifizierten Entität (für Verweise, Aktualisierung)

Tabelle 4: Semantische Konstrukte einer Map. Die Abbildung des Importmodells auf das DHE-Modell wird durch die Kombination der Konstrukte ausgedrückt. Beispielsweise definiert die Kombination der Konstrukte pmPA[pa_lname) und 3101(pmPA.pa_lname) die Identifikation eines Patienten in der DHE-Datenbank über den in der BDT-Nachricht enthaltenen Patientennamen.

4.5.2 Syntaktische Elemente einer Map

Die semantischen Elemente einer Map werden aus syntaktischen Elementen wie *tag* (Bezeichner des Importmodells), *entity*, *attribute* (Bezeichner des DHE-Modells) und *function* konstruiert. Die Regeln zum syntaktischen Aufbau einer Map sind in einer formalen Sprache definiert:

```
MAP ::= ENTITY-LIST TAG-LIST end-of-file
ENTITY-LIST ::= entity [ ATTRIBUTE-LIST ] ENTITY-LIST |
entity [ ATTRIBUTE-LIST ) ENTITY-LIST | nil
TAG-LIST ::= tag ( OPERATION-LIST ) TAG-LIST | nil
ATTRIBUTE-LIST ::= attribute , ATTRIBUTE-LIST | attribute
INSTRUCTION-LIST ::= INSTRUCTION , INSTRUCTION -LIST | INSTRUCTION
INSTRUCTION ::= FUNCTION | ASSIGNMENT
FUNCTION ::= function ( ARGUMENT-LIST ) | function ( )
ASSIGNMENT ::= ATTRIBUTE = RIGHT-ARGUMENT | ATTRIBUTE
ARGUMENT-LIST ::= ARGUMENT , ARGUMENT-LIST | ARGUMENT
ATTRIBUTE ::= entity . attribute
RIGHT-ARGUMENT ::= FUNCTION | ATTRIBUTE | entity | constant | tag
ARGUMENT ::= RIGHT-ARGUMENT | & ATTRIBUTE
```

Abbildung 21: Formale Sprache zum Aufbau einer Map.

Die Sprache besteht aus einer Menge von Regeln, die folgende Gestalt haben: Ein Symbol links von *::=* ist durch Symbole rechts von *::=* zu ersetzen. Alternative Ersatzausdrücke werden durch einen senkrechten Strich | getrennt. Eine Anweisung (*INSTRUCTION*) zum Beispiel kann durch eine Funktion (*FUNCTION*) oder eine Zuweisung (*ASSIGNMENT*) ersetzt werden. Zu ersetzende Symbole, die sogenannten Nichtterminale, werden groß geschrieben. Gestartet wird mit dem Nichtterminal *MAP*, dem sogenannten Axiom. Ersetzt wird, bis nichts mehr zu ersetzen ist. Aus der Map-Sprache kann demnach folgende Map abgeleitet (eine Ersetzung pro Zeile):

```
MAP
ENTITY-LIST TAG-LIST end-of-file
nil TAG-LIST end-of-file
nil tag ( INSTRUCTION-LIST ) TAG-LIST end-of-file
nil tag ( INSTRUCTION-LIST ) nil end-of-file
nil tag ( INSTRUCTION ) nil end-of-file
nil tag ( ASSIGNMENT ) nil end-of-file
nil tag ( ATTRIBUTE ) nil end-of-file
nil tag ( entity . attribute ) nil end-of-file
```

Läßt man noch die in der Map nicht-sichtbaren Symbole *nil* und *end-of-file* weg, so ergibt sich die folgende einfache Map:

```
tag(entity.attribute)
```

In dieser Map wird das durch *tag* bezeichnete Importkonzept auf das durch *entity.attribute* bezeichnete DHE-Konzept abgebildet. Es wurde also eine Zuordnung, d.h. ein inhaltliches Map-Element aus den syntaktischen Map-Elementen konstruiert. Die Map-Sprache ist mit zwölf Produktionsregeln sehr einfach aufgebaut und erlaubt durch das Funktionskonzept die Integration jeder nur denkbaren Logik, die für den Übersetzungsprozess notwendig ist. Eine Funktion kann beliebig viele Argumente haben. Als Argumente sind Wertparameter (*RIGHT-ARGUMENT*) und Adressparameter (*& ATTRIBUTE*) erlaubt. Der Ampersand (&) signalisiert also eine Veränderung des angegebenen Attributs durch die Funktion.

4.6 Modularisierung von Schnittstellen: Was kann generiert werden?

Gemäß 4.4 Abbildung 20 erzeugt der Generator aus der Map (*Schnittstellenmodell*) ein entsprechendes Importprogramm (*Schnittstellenprogramm*). In diesem Abschnitt wird die Umsetzung des Generatorkonzepts beschrieben. Die Schnittstellengenerierung beruht auf einer als *Datenabstraktion (Data hiding)* bekannten Methode, die im folgenden motiviert wird. Im Zusammenhang mit objekt-orientierten Programmiersprachen spricht man statt von Datenabstraktion auch von *Datenkapselung*.

Die Schwierigkeit des Generators besteht darin, Schnittstellenprogramme für beliebige Datenformate (BDT-Feldbeschreibungen, BDT-Nachrichten etc.) zu erzeugen. Wie kann er das tun, ohne alle Datenformate zu kennen? Die Lösung liegt in der Kapselung der Daten durch wohldefinierte Funktionen, d.h. auf die Daten darf ausschließlich über diese Funktionen zugegriffen werden. Der Generator muß dann nur noch die Ein- und Ausgabeparameter, d.h. die Aufrufschnittstellen der Funktionen kennen. Das Wissen über die Organisation der Daten wird in den Funktionen versteckt:

> [Stroustrup87, 133-167] *Data hiding (data abstraction): "... access to data is restricted to a specific set of access functions."*

Im nächsten Schritt müssen die Funktionen identifiziert werden, die ein Schnittstellenprogramm für die Übersetzung der Datenformate benötigt. Um die Allgemeingültigkeit dieser Schnittstellenfunktionen zu betonen, bezeichnen wir das DHE als *Ziel* und die Importdaten als *Quelle*.

Zunächst muß die Quelle in ihre Elemente zerlegt werden. Die Schnittstelle muß dazu das Format, d.h. die Begrenzungen (Begrenzungssymbole, Längenbegrenzungen) der Quellenelemente kennen. Außerdem muß sich die Schnittstelle den für die eindeutige Interpretation der Elemente nötigen Kontext (Interpretationskontext) merken. Enthält die Quelle zum Beispiel ein Element *Patient* und ein Unterelement *Name*, so besitzt das

Name-Element den Interpretationskontext *Patient*. Das Herauslösen von Elementen aus einer Struktur soll im folgenden als (Quell-) *Extraktion* bezeichnet werden.

Darüber hinaus können Granularitätsunterschiede zwischen der Quellstruktur und der Zielstruktur auftreten. Beispielsweise werden *Strasse* und *Ort* in der BDT-Nachricht zu einem Element *Anschrift* zusammengefaßt, während die DHE-Struktur eine Trennung der *Anschrift* in die Elemente *Strasse* und *Ort* vorsieht. Das DHE definiert an dieser Stelle eine präzisere Struktur, die entsprechend präzise Anfragen an das DHE ermöglicht. Derartige strukturelle Anpassungen wollen wir im folgenden als (Ziel-) *Granulierung* bezeichnen.

Werden Inhalte in der Quelle anders codiert als im Ziel, so müssen Umcodierungen vorgenommen werden. Das weibliche Geschlecht muß zum Beispiel von der BDT-Codierung *W* in die DHE-Codierung *F* konvertiert werden. Im folgenden soll von (Ziel-) *Codierung* gesprochen werden. Die Zusammenfassung aus Granulierung und Codierung nennen wir *Konversion*.

Aus den Elementen der Quellstruktur wird nach entsprechender Granulierung und Codierung die Zielstruktur aufgebaut. Das Einfügen von Daten in eine Struktur soll im folgenden (Ziel-) *Insertion* genannt werden.

Die Intelligenz einer Schnittstelle steckt in dem Zusammenspiel bzw. der *Steuerung* der übrigen Funktionen (Extraktion, Konversion, Insertion). Die Schnittstelle kann zum Beispiel quellorientiert (Quelle bestimmt Insertion) oder zielorientiert (Ziel bestimmt Extraktion) vorgehen, oder auch mit einer Zwischenstruktur arbeiten. Eine Abhängigkeit zwischen Insertion und Extraktion entsteht zum Beispiel auch dadurch, daß erst alle Eingabeparameter einer API-Funktion aus der Quelle herausgelöst werden müssen (Extraktion) bevor die API-Funktion aufgerufen werden kann (Insertion). Solche wechselseitigen Abhängigkeiten der Funktionen führen häufig zu einer monolithischen Schnittstellen-Logik, mit dem Nachteil, daß jede Veränderung an der Schnittstelle einen nicht unerheblichen Einarbeitungsaufwand erfordert.

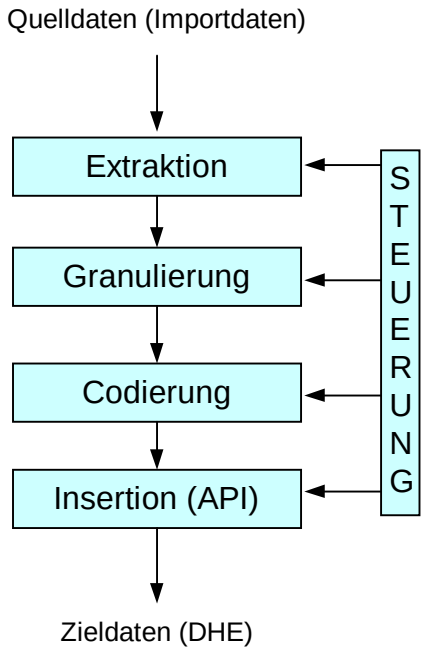

Abbildung 22: Funktionen einer Schnittstelle. In Klammern ist die angestrebte Anwendung dieses Verfahrens abzulesen. Die zentrale Funktion ist die Steuerung (abwechselnde Aktivierung) der übrigen Funktionen. Insgesamt bewirkt die Schnittstelle damit eine Umstrukturierung des Inhalts von den Quelldaten in die Zieldaten.

4.7 Verfeinerung des Generatorkonzepts

Um eine getrennte Entwicklung der Schnittstellenfunktionen zu ermöglichen, werden die Schnittstellenprogramme in einen *Controller* (Steuerung), einen *Parser* (Extraktion), einen *Converter* (Granulierung und Codierung) und das *API* (Insertion) modularisiert. Parser, Converter und API verstecken die Organisation der Daten gegenüber dem Controller, bewirken also die Datenkapselung. Der Generator kann somit den Controller erzeugen, wenn er die Aufrufschnittstellen dieser "Datenkapseln" kennt. Die Aufrufschnittstellen von Parser, Converter und API erhält der Generator wie folgt. Die Parametrisierung der Konversionsfunktionen kann der Generator der

Map entnehmen. Die Aufrufschnittstellen der API-Funktionen entnimmt der Generator entsprechenden DHE-Dateien, welche die Datentypen der einzelnen Daten definieren (Zeichenfolge aus 30 Zeichen, ganze Zahl etc.). Die Aufrufschnittstelle des Parsers wird wie folgt festgelegt: Der Parser zerlegt die Importdaten in eine Wertefolge, erzeugt zu den Werten kontextunabhängige Bezeichner und gibt bei jedem Aufruf das nächste (Bezeichner,Wert)-Paar zurück.

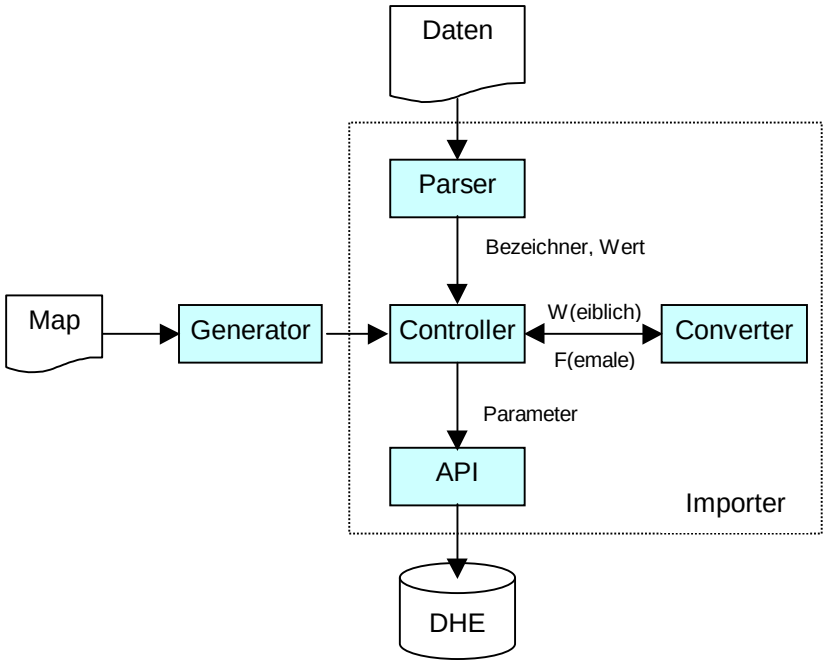

Abbildung 23: Verfeinertes Generatorkonzept. Die Importschnittstelle (Importer) zerfällt in mehrere Module. Der Parser zerlegt die Importdaten in (Bezeichner,Wert)-Paare und liefert bei jedem Aufruf das nächste Paar. Der Converter paßt die Daten an das DHE-Format an. Mit dem API werden die Daten im DHE bearbeitet. Parser, Converter und API kapseln die Einzelheiten über die Daten gegenüber dem Controller ab. Der Generator kann also den Controller für beliebige Datenformate erzeugen.

Die nächsten beiden Abschnitte werden das vorgestellte Verfahren am Beispiel der DHE-Konfiguration und der Generierung der BDT-Schnittstelle illustrieren. Die letzten beiden Abschnitte liefern eine Detailbeschreibung des Controllers und des Generators.

4.8 Konfiguration des DHE

Die Konfigurationsaufgabe für die BDT-Schnittstelle zum DHE kann wie folgt beschrieben werden: Die BDT-Feldbeschreibungen (siehe 4.3) sind auf den DHE-Medizindatentyp (siehe 3.1.1) abzubilden. Der Vollständigkeit halber werden die zu importierenden BDT-Feldbeschreibungen und der relevante DHE-Modellausschnitt noch einmal dargestellt.

7709 Diagnosetext des Tumors
7715 Freitext Lokalisation
7725 Freitext Histologie
7753 TNM T-Kategorie (Basisdok. 2.14.2)
7757 TNM N-Kategorie (Basisdok. 2.14.2)
7760 TNM M-Kategorie (Basisdok. 2.14.2)

Abbildung 24: BDT-Feldbeschreibungen.

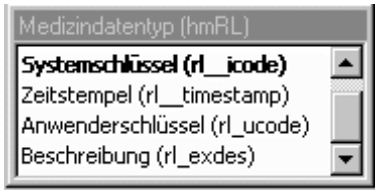

Abbildung 25: DHE-Medizindatentyp.

Wir wollen die BDT-Feldkennung auf den DHE-Anwenderschlüssel und die BDT-Feldbeschreibung auf die DHE-Beschreibung abbilden. Die Bezeichner für die DHE-Konzepte entnimmt man dem DHE-Modell: *hmRL.rl_ucode, hmRL.rl_exdes*. Für die BDT-Konzepte wählen wir die Bezeichner *kennung* und *beschreibung*. Damit kann nun die Map für die DHE-Konfiguration formuliert werden:

```
kennung(hmRL.rl_ucode)
beschreibung(hmRL.rl_exdes)
```

Abbildung 26: Map zur BDT-Konfiguration des DHE.

Jetzt muß nur noch ein Parser geschrieben werden, der die BDT-Feldbeschreibungen in (Bezeichner,Wert)-Paare zerlegt. Die Bezeichner sind *kennung* und *beschreibung* und die Werte kommen aus der Datei in Abbildung 24. In der Programmiersprache C++ kann der Parser wie folgt codiert werden. Der Parser-Code soll nur einen Eindruck vom Aufwand vermitteln.

```
void zerlege(char * bezeichner, char * wert) {
        static int beschreibung = 0;
        static char zeile[ANZAHL_ZEICHEN];
        if (beschreibung)
                { strcpy(bezeichner, “beschreibung“); strcpy(wert, zeile + 5); }
        else { // kennung
                strcpy(bezeichner, “kennung“);
                fgets(zeile, ANZAHL_ZEICHEN, stdin);
                zeile[4] = ‘\0‘;
                strcpy(wert, zeile);
        }
        beschreibung = (beschreibung + 1) % 2;
}
```

Abbildung 27: C++ Parser zur Zerlegung der BDT-Feldbeschreibungen in (Bezeichner,Wert)-Paare.

Der Parser zerlegt die BDT-Feldbeschreibungen in eine Folge von (Bezeichner,Wert)-Paare:

Bezeichner	Wert
kennung	7709
beschreibung	Diagnosetext des Tumors
kennung	7715
beschreibung	Freitext Lokalisation
kennung	7725
beschreibung	Freitext Histologie
etc.	

Abbildung 28: Der Parser erzeugt eine Folge von (Bezeichner,Wert)-Paaren.

Der zu treibende Aufwand ergibt sich also zu 2 Zeilen für die Map und 13 Zeilen für den Parser. Aus der Map und dem Parser kann der Generator dann ein entsprechendes Importprogramm erzeugen, mit dem die BDT-Feldbeschreibungen in das DHE geladen werden können. Angesichts der 6 Zeilen, die hiermit in das DHE importiert werden, mag dieser Aufwand übertrieben erscheinen. Diese 6 Zeilen hätte man auch manuell eingeben können. Bedenkt man aber, daß mit demselben Aufwand ebenso gut tausende von Zeilen importiert werden können, wird der Vorteil sehr schnell deutlich.

4.9 Generierung der BDT-Schnittstelle

Nach Durchführung der DHE-Konfiguration sind die BDT-Konzepte *Tumordiagnose*, *Histologie*, *TNM* etc. im DHE als Medizindatentypen repräsentiert. Analog zur Konfiguration des DHE verfahren wir nun bei der Generierung der BDT-Schnittstelle zum DHE. Die Importaufgabe besteht jetzt in der Abbildung einer BDT-Nachricht (siehe 4.2 und 4.3) auf die DHE-Entitäten *Patient* und *Medizindatum* (siehe 3.1.1). Wieder werden die Importdaten und der relevante DHE-Modellausschnitt dargestellt:

```
01380006100
01330001242
0163101Alphorn
0143102Heidi
017310328101933
0263106Frankfurt am Main
0243107Auf der Wiese 5
0103110W
01380006201
01330001242
0107701
017770120031967
0347709Mamma-Ca. intraduktal re.
0277709unt. äuß. Quadrant
0397709Befall ax. LK Stadium T2 N1 M0
01380006201
01330001242
0107702
0247709Mamma-Ca. links
```

Abbildung 29: BDT-Nachricht. Feldkennungen sind zur besseren Lesbarkeit unterstrichen.

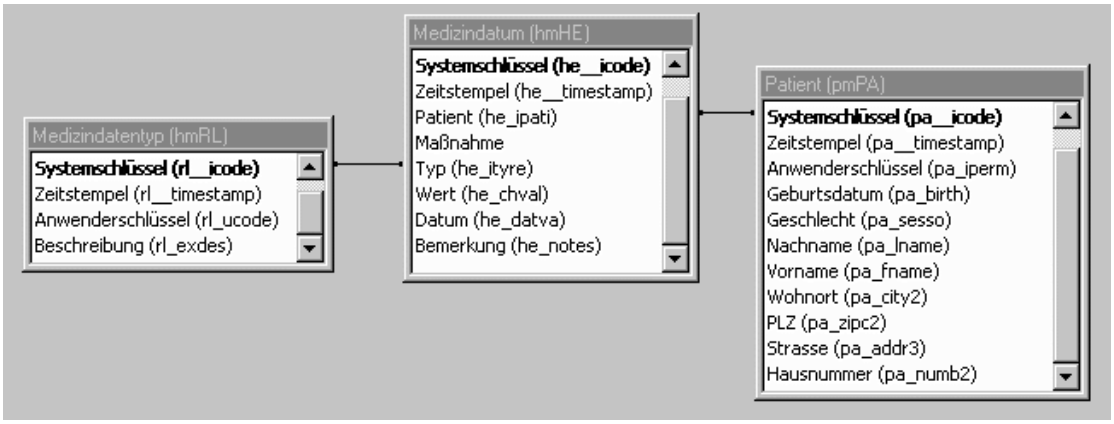

Abbildung 30: Für den Import einer BDT-Nachricht relevanter DHE-Modellausschnitt.

Die Konzepte in der BDT-Nachricht werden durch die BDT-Feldkennungen identifiziert. Die Bezeichner der DHE-Konzepte entnimmt man wieder dem dargestellten Modellausschnitt. Damit kann die Abbildung zwischen BDT und DHE formuliert werden:

```
// INHALTLICHE SCHLÜSSEL
pmPA[pa_iperm]                          // Aktualisierungsschlüssel
hmRL[rl_ucode)                          // Verweisschlüssel
hmHE[he_ipati, he_notes]

// ZUORDNUNGEN
3000(pmPA.pa_iperm)
3101(pmPA.pa_lname)
3102(pmPA.pa_fname)
3103(pmPA.pa_birth=datum(3103))
3106(pmPA.pa_zipc2=plz(3106), pmPA.pa_city2=ort(3106))      // Granulierung
3107(pmPA.pa_addr3=strasse(3107), pmPA.pa_numb2=haus(3107))
3110(pmPA.pa_sesso=geschlecht(3110))                         // Codierung
3203(blob(3201, 3203, pmPA))        // Hauptversicherter (Name, Geburt)
7709(hmHE.he_ipati=pmPA,            // Verweis auf Patient
      hmRL.rl_ucode="7709",         // Konstante in Anführungszeichen
      hmHE.he_ityre=hmRL,           // Typ Tumordiagnose
      hmHE.he_notes=7700,           // Tumoridentifikation
      hmHE.he_datva=datum(7701),    // Konversion Tumordiagnosedatum
      hmHE.he_chval=7709)           // Diagnosetext
```

Abbildung 31: Rudimentäre BDT-Map. Kommentare sind durch // und das Zeilenende begrenzt.

Die BDT-Map soll in Auszügen beschrieben werden. Die zu importierende BDT-Nachricht enthält die Tumordiagnosen der Patientin "Heidi Alphorn". Im DHE soll der Patient *pmPA* über das Attribut *pa_iperm* identifiziert werden. Zu diesem Zweck definiert man einen entsprechenden Schlüssel: *pmPA[pa_iperm]*. Durch die Zuordnung *3000(pmPA.pa_iperm)* wird die Verbindung zwischen BDT und DHE hergestellt, wobei *3000* die Patientennummer in der BDT-Nachricht kennzeichnet. Insgesamt wird der Patient also über die in der BDT-Nachricht enthaltene Patientennummer identifiziert. Da wir einen Aktualisierungsschlüssel definiert haben, werden die Patientendaten aus der BDT-Nachricht aktualisiert: Existiert der Patient *1242* (Inhalt von Feld *3000*) noch nicht, so wird für ihn ein neuer Stamm-Datensatz im DHE angelegt. Andernfalls werden die Stammdaten des Patienten aktualisiert.

Im BDT-Modell gibt es außerdem das Konzept des „Hauptversicherten“, das durch die Feldkennungen *3201* (Name des Hauptversicherten) und *3203* (Geburtsdatum des Hauptversicherten) vertreten ist. Da das Konzept der Mitversicherung eine Eigenheit des deutschen Gesundheitswesens ist, gibt es keine Entsprechung im DHE-Modell. Aus diesem Grund haben wir eine Konversionsfunktion *blob* entwickelt, welche die

Daten des Hauptversicherten im DHE als Extended data (siehe 3.1.1) zu der Patient-Entität abspeichert.

Die BDT-Tumordiagnose *7709* wird auf das DHE-Medizindatum *hmHE* abgebildet. Der Diagnosetext selbst wird in dem Attribut *hmHE.he_chval* untergebracht. Der Patientenbezug wird über den Ausdruck *hmHE.he_ipati=pmPA* hergestellt (die Attribute der Patient-Entität sind bereits aus den BDT-Feldern *3000-3110* bestimmt). Der Verweis des DHE-Medizindatums auf den entsprechenden DHE-Medizindatentyp erfolgt in zwei Schritten. Gemäß Verweisschlüssel *hmRL[rl_ucode)* wird der DHE-Medizindatentyp *hmRL* über das Attribut *rl_ucode* identifiziert. Die Identifikation des Datentyps erfolgt in unserem Fall also durch die Zuweisung *hmRL.rl_ucode="7709"* (Schlüsselausprägung). Gemäß Konfiguration handelt es sich dabei um den Medizindatentyp „Diagnosetext des Tumors“. Anschließend kann der Bezug des Medizindatums zum Datentyp mit dem Ausdruck *hmHE.he_ityre=hmRL* hergestellt werden. Aufgrund dieser Zuordnungen weiß nun das DHE, daß der aus der BDT-Nachricht stammende Text "Mamma-Ca. intraduktal re., unt. äuß. Quadrant, Befall ax. LK Stadium T2 N1 M0" (der BDT-Parser faßt aufeinanderfolgende Diagnose-Textzeilen zusammen) zum Patienten "Heidi Alphorn" gehört und daß es sich dabei um eine Tumordiagnose handelt. Das Datum der Tumordiagnose *7701* wird nach Anpassung an das DHE-Datumsformat in dem Attribut *he_datva*, und die Tumor-Identifikation *7700* in dem Attribut *he_notes* abgelegt.

Aus der BDT-Map erzeugt der Generator eine BDT-Importschnittstelle zum DHE. Mit dieser Schnittstelle können beliebig viele BDT-Nachrichten in das DHE importiert werden. Auf eine Darstellung des Parser und der Konversionsfunktionen *datum* (Umcodierung vom BDT-Datumformat in das DHE-Datumformat), *plz*, *ort*, *strasse*, *haus* (Zerlegung der BDT-Anschrift in PLZ, Ort, Straße und Hausnummer des DHE) und *geschlecht* (Umcodierung des weiblichen Geschlechts von der BDT-Codierung *W* in die DHE-Codierung *F*) wird an dieser Stelle verzichtet.

4.10 Der Controller: Ein generierbares Modul

Gemäß Abbildung 23 in Abschnitt 4.6 erzeugt der Generator den Controller, der in diesem Abschnitt genauer beschrieben wird. Die Controller-Logik wird vom Generator gut strukturiert, um eine Veränderung der Logik durch den Schnittstellen-Entwickler zu ermöglichen. Solche Veränderungen waren z.B. während der Testphase des Generators sehr hilfreich (Addition von Testlogik zur der vom Generator erzeugten Logik). Die Controller-Logik zerfällt im wesentlichen in drei Funktionsgruppen:

Entity-Funktionen leisten die logische Umsetzung der in Abschnitt 4.5.1 beschriebenen inhaltlichen Schlüssel für DHE-Entitäten. Der Aktualisierungsschlüssel *pmPA[pa_lname, pa_birth]* implementiert zum Beispiel die Identifikation des Patienten *pmPA* anhand seines Namens *pa_lname* und seines Geburtsdatums *pa_birth*. Die Ausprägung des Schlüssels ist z.B. *[Müller, 1976/07/21]* und erfolgt aus den Importdaten. Die entsprechende Entity-Funktion muß dazu prüfen, ob bereits ein Patient mit den angegebenen Daten in der DHE-Datenbank existiert. Ist das nicht der Fall, so wird für den Patienten ein neuer Datensatz angelegt. Andernfalls werden die Daten des Patienten aktualisiert. Entity-Funktionen fassen also mehrere API-Aufrufe (siehe 3.1.4) zu mächtigeren Funktionen zusammen und verbergen damit die API-Programmierung (Allokation von Speicher, Initialisierungen, Fehlerkontrolle etc.). Eine Entity-Funktion gibt stets den Systemschlüssel der bearbeiteten Entität zurück.

Für jeden Bezeichner (Tag) des Importmodells erzeugt der Generator außerdem eine Tag-Funktion, die immer dann aufgerufen wird, wenn der Parser ein entsprechendes (Bezeichner,Wert)-Paar liefert. *Tag-Funktionen* setzen die in der Map formulierten Beziehungen zwischen den Modell-Konzepten um. Die Umsetzung erfolgt durch Zuordnungen (Importmodell zu DHE-Modell), Verweise (innerhalb des DHE-Modells) und Konversionen, wie sie in der Map angegeben sind. Darüber hinaus enthalten Tag-Funktionen aber auch logische Details, die nicht in der Map zu finden sind. Tag-Funktionen entscheiden insbesondere darüber, wann eine Interaktion mit dem DHE durchzuführen ist. Die Interaktion mit dem DHE erfolgt dabei durch den Aufruf von Entity-Funktionen. Tag-Funktionen sorgen z.B. dafür, daß die in das DHE zu importierenden Daten nicht zu früh (das DHE fordert z.B. die Mindest-Eingabe von

Name, Geschlecht und Geburtsdatum des Patienten) im DHE gespeichert werden und verwalten den dazu nötigen Zwischenspeicher.

Protokoll-Funktionen zeichnen den Importverlauf für den Schnittstellen-Entwickler auf. Aufgezeichnet werden z.B. Fehler während des Imports und die Dauer des Imports.

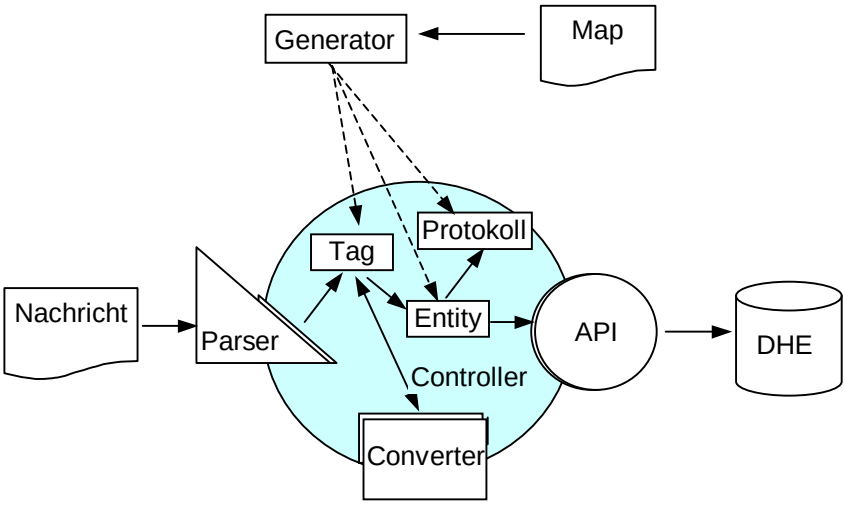

Abbildung 32: Generatorkonzept mit Detaildarstellung des Controllers. Der Controller zerfällt in Entity-, Tag- und Protokoll-Funktionen. Entity-Funktionen implementieren die inhaltlichen Schlüssel in der Map. Tag-Funktionen implementieren die Beziehungen zwischen den Modellkonzepten in der Map. Protokoll-Funktionen zeichnen den Importverlauf auf.

4.11 Der Generator: Von der Map zum Controller

In diesem Abschnitt wird die Wirkungsweise des Generators beschrieben, der die Map in den Controller überführt. Der Generator ist im wesentlichen ein Compiler und besteht aus zwei Phasen: *Analyse* und *Synthese* (*Analyse-Synthese-Modell* der Compilierung gemäß [Aho88]). Die Analyse zerlegt die Map in ihre Bestandteile. Die Synthese konstruiert daraus den entsprechenden Controller. Die Analyse wird weiter in eine lexikalische, syntaktische und semantische Analyse aufgeteilt. Die lexikalische Analyse (engl. scanning) bildet aus dem Zeichenstrom einzelne Worte, wie sie von der Syntaxanalyse gebraucht werden. Die Syntaxanalyse (engl. parsing) prüft, ob der Wortstrom gewissen Regeln, d.h. der formalen Sprache in 4.5.2 folgt. Diese formale Sprache soll im folgenden als Map-Sprache bezeichnet werden. Die semantische Analyse des Generators überprüft, ob die in der Map angegebenen Entitäten und Attribute im DHE existieren. Insgesamt ergibt sich also folgendes gegenüber [Aho88, 12] abgewandelte Phasen-Modell (Zwischencode-Erzeugung und Code-Optimierung entfallen):

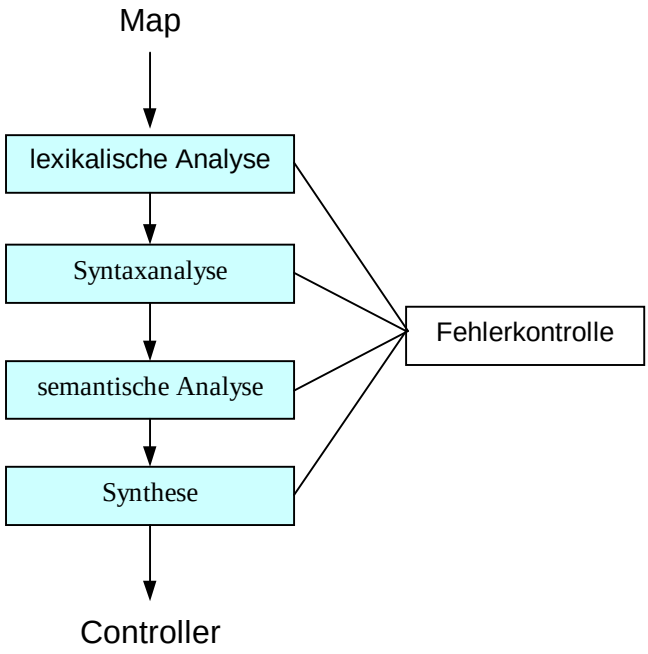

Abbildung 33: Phasen des Generators nach [Aho88]. Der Generator besteht aus vier ineinandergreifende Phasen: lexikalische Analyse, syntaktische Analyse, semantische Analyse und Synthese. Die lexikalische Analyse erzeugt aus Zeichen Worte und die Syntaxanalyse prüft, ob die Worte den Regeln der Map-Sprache folgen. Die semantische Analyse prüft, ob die in der Map angegebene Entitäten und Attribute im DHE existieren. Die Synthese erzeugt den Controller und ist im wesentlichen an die Syntaxanalyse gekoppelt. Die Fehlerkontrolle steht in Wechselbeziehung zu den vier Phasen.

Die Syntaxanalyse wird aus der Map-Sprache wie folgt implementiert. Jedes Nichtterminal links von dem Symbol *::=* entspricht einem gleichnamigen Unterprogramm. Die Unterprogrammlogik reflektiert im wesentlichen die zugehörige rechte Seite. Die Unterprogramme rufen zu diesem Zweck den Scanner (lexikalische Analyse) auf und entscheiden anhand der gelesenen Worte, welche Regel anzuwenden ist. Ein Nichtterminal rechts von dem Symbol *::=* entspricht einem Unterprogramm-Aufruf. Endständige Rekursionen (ein Unterprogramm ruft sich am Ende selbst wieder auf) können dabei in Iterationen (Schleifen) umgewandelt werden.

Das folgende Struktogramm zeigt einen kleinen Ausschnitt aus der Synthese-Logik des Generators. Das Ergebnis der Synthese sind Tag-Funktionen des Controllers. Gemäß 4.10 entscheiden Tag-Funktionen darüber, wann eine Interaktion mit der DHE-Datenbank durchzuführen ist. Der Generator addiert diese DHE-Interaktionslogik zu den Anweisungen in der Map (Zuordnungen, Verweise, Konversionen) hinzu:

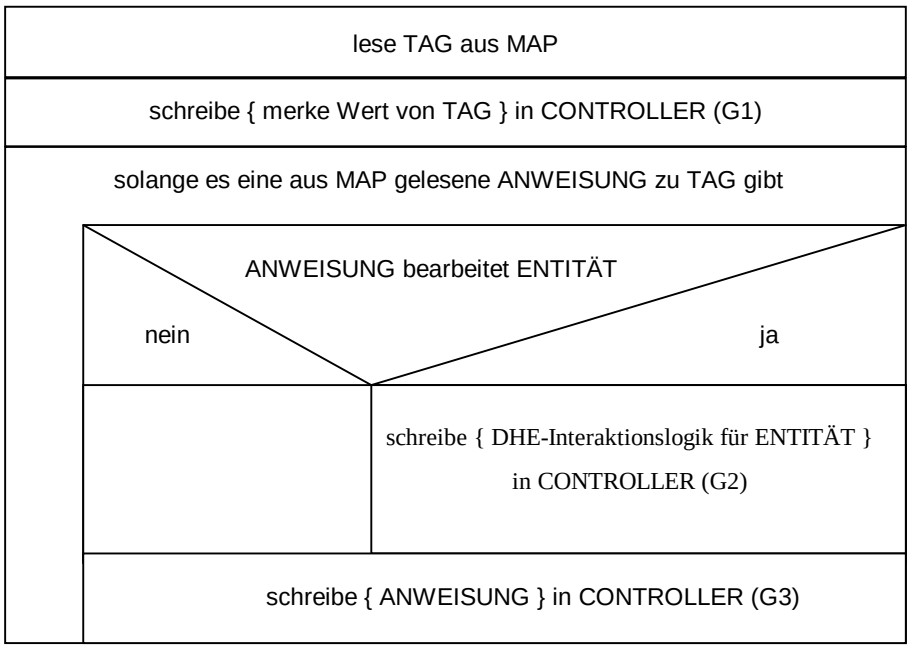

Abbildung 34: Struktogramm zur Synthese der Tag-Funktionen eines Controllers. Der Generator erzeugt insbesondere die DHE-Interaktionslogik (G2).

Abschließend soll die DHE-Interaktionslogik noch an der folgenden Map illustriert werden. Die Bedeutung der Bezeichner *7709*, *hmHE* etc. ist an dieser Stelle nicht relevant und kann gegebenenfalls in Abschnitt 4.9 nachgelesen werden.

```
7709(hmHE.he_ipati=pmPA,
    hmRL.rl_ucode="Tumordiagnose",
    hmHE.he_ityre=hmRL,
    hmHE.he_notes=7700,
    hmHE.he_datva=datum(7701),
    hmHE.he_chval)
```

Der Generator liest die Map und erzeugt für das Tag *7709* des Importmodells eine Tag-Funktion des Controllers. Die Tag-Funktion ist eine Folge von Anweisungen, die zum Teil aus der Map gelesen (G3) und zum Teil vom Generator hinzugefügt (G1, G2) werden. Am Ende einer Controller-Anweisung wird angegeben, welche Generator-Anweisung aus Abbildung 34 die Controller-Anweisung erzeugt hat. Die Controller-Anweisung *lege* bewirkt die temporäre Speicherung in einem Zwischenspeicher, die Anweisung *sichere* die permanente Speicherung im DHE.

```
merke Wert von 7709 (G1)
ermittle Systemschlüssel von pmPA (G2)
sichere hmHE, falls hmHE.he_ipati schon belegt (G2)
lege pmPA-Systemschlüssel in hmHE.he_ipati (G3)
sichere hmRL, falls hmRL.rl_ucode schon belegt (G2)
lege „Tumordiagnose“ in hmRL.rl_ucode (G3)
ermittle Systemschlüssel von hmRL (G2)
sichere hmHE, falls hmHE.he_ityre schon belegt (G2)
lege hmRL-Systemschlüssel in hmHE.he_ityre (G3)
sichere hmHE, falls hmHE.he_notes schon belegt (G2)
lege 7700-Wert in hmHE.he_notes (G3)
sichere hmHE, falls hmHE.he_datva schon belegt (G2)
konvertiere Datumsformat von 7701-Wert und lege Resultat in hmHE.he_datva (G3)
sichere hmHE, falls hmHE.he_chval schon belegt (G2)
lege 7709-Wert in hmHE.he_chval (G3)
```

Die Controller-Logik ist immer noch stark vereinfacht. Zum Beispiel muß der Controller zu jeder Entität eine Liste mit den bereits belegten Attributen führen, um Kollisionen bei der Wertzuweisung und somit Wiederholungen in den Importdaten erkennen zu können.

5 Ergebnisse

Ziel dieser Arbeit war die Erprobung des DHE. Zu diesem Zweck wurde das GTDS mit dem DHE über eine BDT-Nachrichtenschnittstelle verbunden. Damit konnte sowohl die Anbindung einer existenten Anwendung an das DHE als auch die Integration zweier unabhängiger Anwendungen (GTDS - Stationsarbeitsplatz) mittels DHE demonstriert werden. Im folgenden werden unsere Erfahrungen mit dem DHE dokumentiert. Darüber hinaus lieferte der von uns entwickelte Generator einige überraschende Ergebnisse.

5.1 Unsere Erfahrungen mit dem DHE

5.1.1 Integration mittels DHE

Als Middleware ist es die Aufgabe des DHE, die Integration von Anwendungssystemen zu unterstützen (siehe 3.1.2). Im Rahmen dieser Arbeit wurde zur Erprobung des DHE das Gießender Tumordokumentationssystem (GTDS) mit dem DHE verbunden. Unseren Erfahrungen zufolge war die Unterstützung von seiten des DHE begrenzt.

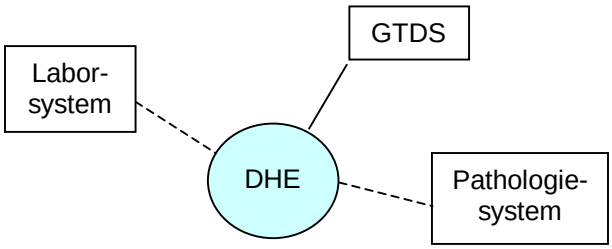

Abbildung 35: Das DHE unterstützt die Integration von Anwendungssystemen. Aber inwieweit?

Um den Grad der DHE-Unterstützung bei der Anwendungsintegration prägnant formulieren zu können, wird der Integrationsbegriff zunächst verfeinert. Abhängig davon, ob die Integrationsabsicht bereits vor oder erst nach der Implementierung einer Anwendung berücksichtigt wird, soll zwischen *Pre- und Postintegration* unterschieden

werden. Preintegrierte Anwendungen verfügen also bereits über Schnittstellen zu anderen Systemen. Eine Preintegration mit dem DHE erfordert zum Beispiel die Berücksichtigung der DHE-Programmierschnittstelle - dem API - bereits bei der Entwicklung der Anwendung. Ferner muß differenziert werden, was eine Anwendung von einer anderen Anwendung einbezieht, d.h. integriert: Sind es die Daten oder ist es die Funktionalität? Mit Funktionalität sind in diesem Zusammenhang aufrufbare Softwarefunktionen gemeint, die auch ohne Benutzer-Interaktion ausgeführt werden können (entfernter Funktionsaufruf). Gegenstand der Integration können also *Anwendungsdaten* oder *Anwendungsfunktionen* sein.

Bei der DHE-Integration des GTDS handelt es sich um eine Postintegration, da weder das Modell noch das API des DHE bei der GTDS-Entwicklung berücksichtigt wurden. Es mußte folglich eine Lösung entwickelt werden, welche die Informationen vom GTDS zum DHE überträgt. Gekennzeichnet war diese Lösung vor allem durch die *Replikation* der GTDS-Daten in der DHE-Datenbank. Die Replikation der Anwendungsdaten umfaßte Tätigkeiten wie die Abbildung der Datenmodelle (GTDS-BDT-DHE), die Erstellung der Datenredundanz (Import von BDT-Dateien in die DHE-Datenbank) und die Redundanzkontrolle (Sendung von BDT-Nachrichten bei Veränderungen der GTDS-Datenbank). Vereinfacht wurde unsere DHE-Demonstration durch die Begrenzung der Modellausschnitte sowie die Annahme, daß die GTDS-relevanten Daten im DHE nicht verändert werden. Die Datenredundanz zwischen Anwendung und DHE wird im allgemeinen den Import großer Datenmengen in die DHE-Datenbank notwendig machen. Das DHE unterstützt diesen Datentransfer nicht. Im Rahmen dieser Arbeit wurde daher ein Verfahren entwickelt, das den Datentransfer zwischen Anwendung und DHE unterstützt (siehe 4.4, 5.3).

Darüber hinaus ist es für eine andere DHE-basierte Anwendung nach wie vor nicht möglich, Funktionen des GTDS in Anspruch zu nehmen. Das GTDS wäre zm Beispiel funktional mit einem Arbeitsplatzsystem integriert, wenn der Arzt nach Bereitstellung der notwendigen Parameter (Gewicht des Patienten etc.) vom GTDS einen Plan für die Zytostatika-Medikation berechnen lassen könnte. Das GTDS wurde im Rahmen dieser Arbeit nur datenintegriert. Die funktionale Integration von Anwendungen wird durch das DHE nicht unterstützt.

Ergebnis: Die Post-Integration von Anwendungssystemen mittels DHE erfordert die Replikation der Anwendungsdaten. Die Integration von Anwendungsfunktionen wird nicht unterstützt.

5.1.2 Das Modell

Im großen und ganzen konnten die Informationen des BDT-Tumordatensatzes im DHE untergebracht werden. Als flexibel erwies sich in diesem Zusammenhang die *Konfigurierbarkeit des DHE*. Beispielsweise waren die BDT-Konzepte Tumordiagnose, Histologie und TNM-Klassifikation im DHE nicht vorgesehen. Das DHE verfügte jedoch über ein Konzept "Medizindatentyp", das die Beschreibung der BDT-Konzepte im DHE erlaubte. Das DHE kann somit an individuelle Erfordernisse angepaßt werden. Entsprechend können auch die Leistungsstellen eines Klinikums (Pathologie, Labor, Radiologie, ...) oder das Leistungsangebot einer Leistungsstelle (Computertomografie, Röntgenuntersuchung in der Radiologie) beschrieben werden. Zwei Probleme sind bei der Abbildung des BDT-Modells auf das DHE-Modell dennoch aufgetreten:

Das BDT-Konzept des *Hauptversicherten* war im DHE-Modell nicht repräsentiert. In solche Fällen bietet das DHE die Möglichkeit, sogenannte BLOBs (Binary Large Objects, 3.1.1) mit einer im DHE vorhandenen Entität wie dem Patienten zu verbinden. Das Format der BLOB-Daten wird aber nicht mehr durch das DHE-Modell festgelegt und erfordert Verabredungen zwischen den DHE-Anwendungsentwicklern.

Das DHE unterstützte darüber nur bedingt die *Zusammenfassung und Wiederholung von klinischen Daten*. Das DHE-Modell bietet die Möglichkeit, zu einem Patienten klinische Daten (Health data, Medizindaten) abzuspeichern. Auf diese Weise konnten die Tumordaten aus der BDT-Nachricht im DHE repräsentiert werden. Ein BDT-Tumor war dabei in mehrere Elemente wie zum Beispiel Diagnose, Histologie, Lokalisation und TNM-Klassifikation strukturiert. Das DHE erlaubte es jedoch nicht, klinische Datenelemente zusammenzufassen. Bei mehr als einer Tumordiagnose ergab sich somit das Problem, daß die Zusammengehörigkeit der Daten nicht mehr aus dem

DHE rekonstruiert werden konnte. Das folgende Beispiel verdeutlicht das Strukturproblem, wobei eine Zugehörigkeit durch eine Einrückung gekennzeichnet ist:

Patient
  Mamma-Ca. links *(Diagnose-2)*
  Mamma-Ca. intraduktal re. *(Diagnose-1)*
  T2 N1 M0 *(TNM-1)*
  T1 N0 M0 *(TNM-2)*

Abbildung 36: Welche TNM-Klassifikation gehört zu welcher Diagnose?

Pragmatisch wurde das Problem dadurch gelöst, daß zu der Diagnose, der TNM-Klassifikation und den anderen Datenelementen die vom GTDS gelieferte Tumoridentifikation mit abgespeichert wurde. Über die Tumoridentikfikation konnten zusammengehörige Daten dann wieder zusammengeführt werden. Der Pragmatismus dieser Lösung äußert sich in drei Schwierigkeiten: Erstens war die Tumoridentifikation nicht im DHE vorgesehen und wurde daher als Notiz zu den klinischen Daten abgespeichert. Diese Notiz könnte von einer anderen DHE-basierten Anwendung leicht verändert werden. Zweitens erlaubt die GTDS-Tumoridentifikation keine GTDS-übergreifende Zusammenführung von Tumordaten. Drittens ergibt sich dasselbe Problem wieder, wenn zum Beispiel der Histologiebefund noch weiter in einen Histologietext und einen Histologieschlüssel strukturiert werden soll.

<u>Ergebnis</u>: Das DHE-Modell ist zwar umfassend und konfigurierbar, bestimmte Konzepte wie der Hauptversicherte sind aber nicht repräsentiert. Die Strukturierung klinischer Daten ist eingeschränkt.

5.1.3 Die Programmierung

Gemäß 3.1.4 gibt es zu jeder Entität im DHE-Modell drei API-Funktionen (Maintain, List, Row) für die Eingabe der Daten in die DHE-Datenbank und die Selektion der Daten aus der DHE-Datenbank. Bei über 200 DHE-Entitäten kommt man damit auf mehr als 600 API-Funktionen. Das API ermöglicht den Zugriff auf die entfernte DHE-Datenbank, ohne daß sich der Entwickler mit Belangen der Datenübertragung beschäftigen muß. Unseren Erfahrungen zufolge war die API-Programmierung dennoch aufwendig. Die folgenden Beispiele sollen das illustrieren.

Bei der Programmierung muß jeder API-Aufruf vor- und nachbereitet werden. Zur Vorbereitung gehört die Initialisierung einer Datenstruktur mit den Attributen der bearbeiteten Entität (3.1.1, 3.1.4). Der Entwickler mußte zu diesem Zweck jedes einzelne Attribut mit einem, eventuell leeren Wert belegen. Da die Entitäten zum Teil fünfzig und mehr Attribute haben, von denen häufig nur wenige benötigt werden, wurde dieses Initialisierungsverfahren sehr schnell aufwendig. Zur Vorbereitung zählt neben der Initialisierung auch die Speicherallokation für die Datenstrukturen. Die Nachbereitung eines API-Aufrufes besteht vor allem in der Fehlerkontrolle.

Die Eingabe großer Datenmengen in die DHE-Datenbank wurde von seiten des DHE nicht unterstützt. Zu diesem Zweck mußten häufig Importprogramme auf API-Basis entwickelt werden. Im Rahmen dieser Arbeit wurde ein generisches Verfahren vorgestellt (siehe 4.4), mit dem dieser Aufwand deutlich reduziert werden kann. Der Entwickler formuliert nur noch die Abbildung des Importmodells auf das DHE-Modell und überläßt die Erstellung des Importprogramms einem Generatorprogramm.

Auch die Selektion von Daten aus dem DHE war aufwendig, wie das folgende Beispiel verdeutlicht: Gemäß 3.1.4 stellt das DHE zwei Sorten von Wiedergewinnungs-funktionen zur Verfügung: *List* und *Row*. Mit List erhält man alle Entitäten (z.B. Patienten) zu gegebenen Attributen (z.B. Geburt in 1976). Mit Row erhält man alle Attribute (Anschrift, Geburtsdatum, ...) einer bestimmten Entität. Sucht man die Anschriften der Patienten mit Geburt in 1976, so müssen mit der Funktion List-Patient zunächst die Systemschlüssel der Patienten mit Geburt in 1976 ermittelt werden. Für

jeden ermittelten Patient-Systemschlüssel wird dann die Funktion Row-Patient aufgerufen, die unter anderem die Anschrift des Patienten liefert. Der folgende Pseudocode skizziert die erforderliche Anwendungslogik unter Verwendung der in 3.1.1 und 3.1.4 beschriebenen Nomenklatur der Entitäten, Attribute, Funktionen und Datenstrukturen. Der Übersicht halber wird auf die volle Vor- und Nachbereitung der API-Aufrufe verzichtet. Die DHE-Bezeichner sind zur besseren Lesbarkeit fett hervorgehoben:

```
pmPA__IL.pa_birth = "1976"
pmPA__L(pmPA__IL, pmPA__OL)
index = 0
Solange pmPA__OL[index] definiert ist
        pmPA__IR.pa__icode = pmPA__OL[index].pa__icode
        pmPA__R(pmPA__IR, pmPA__OR)
        Ausgabe pmPA__OR.pa_addr
        erhöhe index um 1
```

Deklarative Sprachen können viele Details der prozeduralen Programmierung verbergen, ohne die Ausdrucksfähigkeit wesentlich einzuschränken. Mit einer Datenbankabfrage-Sprache wie SQL (Structured Query Language) kann dieselbe Selektion in einer einzigen Anweisung formuliert werden:

```
select pa_addr from pmPA where pa_birth="1976"
```

Ergebnis: Die prozedurale API-Programmierung ist relativ aufwendig. Deklarative Sprachen und entsprechende Prozessoren können die Eingabe (Map, Generator) und die Selektion (SQL, SQL-Prozessor) von DHE-Daten stark vereinfachen.

5.1.4 Zusammenfassung der Erfahrungen

Die in den vorausgegangenen Abschnitten geschilderten Erfahrungen mit dem DHE werden der Übersichtlichkeit halber noch einmal in einer Tabelle zusammengefaßt:

	Positiv	**Negativ**
Anwendungsintegration mittels DHE	Das DHE unterstützt die Preintegration von Anwendungsdaten.	Die Postintegration von Anwendungen erfordert die Replikation von Anwendungsdaten. Die Integration von Anwendungsfunktionen wird nicht unterstützt.
DHE-Modell	Das DHE-Modell ist umfassend und konfigurierbar.	Bestimmte Konzepte wie der Hauptversicherte sind nicht repräsentiert. Die Strukturierung klinischer Daten ist eingeschränkt.
DHE-Programmierung	Die Übertragung der Daten zwischen Anwendung und Datenbank sowie die Datenbank selbst sind transparent für den Anwendungsentwickler.	Überwiegend primitive API-Funktionen machen die Programmierung aufwendig.

Tabelle 5: Übersicht über unsere DHE-Erfahrungen.

5.2 Verbesserungen des DHE aufgrund dieser Arbeit

Unsere Erfahrungen mit dem DHE wurden teilweise dazu genutzt, um das DHE und den entprechenden HISA-Standard zu verbessern. Folgende Verbesserungen sind auf diese Arbeit zurückzuführen:

In das DHE-Modell wurde eine rekursive Beziehung zwischen Medizindaten (Health data) eingeführt, die in dem nachfolgenden DHE-Modellausschnitt durch die Erweiterung der Attribut-Liste um das Attribut *gehört_zu* angedeutet wird. Damit ist es möglich, Medizindaten zu gruppieren und zu wiederholen (siehe 5.1.2).

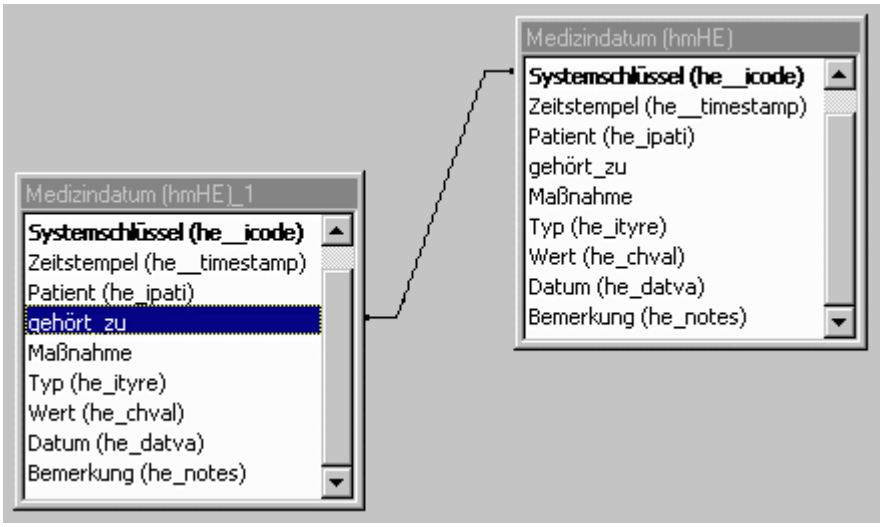

Abbildung 37: Strukturierung der Medizindaten.

Das folgende Beispiel verdeutlicht den Unterschied noch einmal. Ein Patient habe zwei Tumordiagnosen Diagnose-1 und Diagnose-2. Zu jeder Diagnose gibt es einen Histologiebefund und eine TNM-Klassifikation. Eine Zugehörigkeitsbeziehung wird durch eine Einrückung gekennzeichnet:

VORHER	NACHHER
Patient	Patient
Diagnose-1	Diagnose-1
Histologie-1	Histologie-1
TNM-1	TNM-1
Diagnose-2	Diagnose-2
Histologie-2	Histologie-2
TNM-2	TNM-2

Abbildung 38: Der Vorher-Nachher-Vergleich verdeutlicht die Modellverbesserung. Bei der rechten Strukturierung ist eindeutig, daß Histologie-1 und TNM-1 zu Diagnose-1 gehören.

Da die Erfahrungen mit dem DHE auch in den HISA-Standard [CEN97] eingebracht wurden, wirkte sich die Modellverbesserung auch auf den Standard aus. Während im

DHE-Modell [Ferrara96i] die Strukturierung klinischer Daten noch eingeschränkt ist, wird sie im HISA-Modell (siehe Anhang 10.5) ausdrücklich gefordert:

> *Health characteristics* (klinische Daten) *may be either elementary, for example relating to one single value, or structured for example comprising other elementary or, recursively, structured health characteristics.*

Neben dem DHE-Modell wurde auch die DHE-Programmierung verbessert (siehe 5.1.3). Um den Programmieraufwand bei der Initialisierung großer Datenstrukturen zu verringern, haben die DHE-Ingenieure Initialisierungsfunktionen in das API eingeführt. Diese Initialisierungsfunktionen belegen die einzelnen Attribute einer Datenstruktur mit leeren Werten vor, sodaß der Entwickler nur noch diejenigen Attribute mit Werten belgen muß, die im Rahmen der Anwendungsentwicklung interessieren. Darüber hinaus wurde das API um eine SQL-Funktion bereichert, womit beliebige Suchanfragen an das DHE schnell formuliert werden können. Schließlich wurde zur Verbesserung der Dateneingabe auch der Generator an die DHE-Ingenieure weitergegeben.

> *Ergebnis: Das DHE und der HISA-Standard wurden aufgrund unserer Erfahrungen und Entwicklungen verbessert.*

5.3 Der Generator und sein Nutzen

5.3.1 Einsatz durch andere HANSA-Partner

Kurz vor Ende des HANSA-Projektes wurde der Generator vom niederländischen Partner HISCOM angefordert, um große Datenmengen in das DHE zu importieren. Obgleich der Generator ursprünglich nur für die interne Benutzung konzipiert und überdies zu diesem Zeitpunkt nur marginal dokumentiert war, konnte HISCOM ohne weitere Unterstützung die entsprechenden Importprogramme erzeugen. Ähnliche Erfahrungen hat auch der portugiesische HANSA-Partner mit dem Generator gemacht.

Der Generator wurde außerdem an den HANSA-Partner Magdeburg weitergegeben. Magdeburg entwickelte auf dem DHE eine einfache Anwendung, die dem Arzt

Informationen von Tumorpatienten präsentierte. Die Informationen sollten aus dem in Magdeburg installierten GTDS kommen. Die Weitergabe des Gießener BDT-Importprogramms wäre in diesem Fall problematisch gewesen, da Magdeburg mit einer anderen DHE-Version (nicht kompatible Datentypen) gearbeitet hat. Stattdessen wurden der Generator und die BDT-Map weitergegeben. Der Generator erzeugte dann ein entsprechendes BDT-Importprogramm unter Verwendung der gegebenen DHE-Version. Mit diesem Importprogramm konnte dann eine Verbindung des Magdeburger GTDS zum Magdeburger DHE hergestellt werden.

5.3.2 Verwendung für die Entwicklung einer HL7-Schnittstelle zum DHE

Existente Anwendungssysteme verfügen häufig über die übliche Kommunikation, Informationen in Form von Nachrichten auszutauschen (siehe auch 4.2). Der Nachrichtenstandard *HL7* (Health Level Seven) zum Beispiel wird von zahlreichen Anwendungen unterstützt. Im Unterschied zum deutschen BDT-Standard handelt es sich bei HL7 um einen internationalen Standard, der 1987 seine Anfänge in den USA fand [Dudeck95]. Der HL7-Standard modelliert Transaktionen des klinischen Alltags wie die Patientenaufnahme in ein Krankenhaus (ADT: Admission, Discharge, Transfer), die Laboranfrage oder die Laborbefundung (Query/Result). Um die Anbindung von HL7-Anwendungen an das DHE zu ermöglichen, mußte eine HL7-Schnittstelle zum DHE entwickelt werden. In diesem Abschnitt wird beschrieben, wie der Generator zur Entwicklung dieser Schnittstelle verwendet wurde.

Analog zur BDT-Schnittstelle (siehe 4.9) mußte wieder ein *Parser* geschrieben werden, der die HL7-Nachrichten in eine Folge von (Bezeichner,Wert)-Paare zerlegt. Um den Aufbau der verwendeten HL7-Bezeichner verstehen zu können, muß zunächst das HL7-Format betrachtet werden.

HL7-Nachrichten sind in *Segmente*, *Felder* und *Komponenten* strukturiert, wobei ein Feld zu einem Segment, und eine Komponente zu einem Feld gehört. Die folgende HL7-Nachricht könnte bei der Aufnahme des Patienten "William Jones" vom Stationssystem an das DHE geschickt werden:

```
MSH|^~\&|ADT1|MCM|LABADT|MCM|198808181126||ADT^A01|MSG00001|P|2.3<cr>
EVN|A01|198808181123<cr>
PID|||PATID1234^5^M11||JONES^WILLIAM^A^III||19610615|M||C|1200 N
        ELMSTREET^^GREENSBORO^NC^27401-1020||(919)379-1212|(919)271-
        3434||S||PATID12345001^2^M10|123456789|<cr>
PV1|1|I|2000^2012^01||||004777^LEBAUER^SIDNEY^J.|||SUR||||ADM|A0<cr>
```

Abbildung 39: Beispiel einer ADT-Nachricht (HL7 Version 2.3).

Segmente, Felder und Komponenten werden durch besondere Zeichen (Carriage Return <cr>, Balken | und Dach ^), sogenannte Delimiter, begrenzt. Segmente haben darüber hinaus einen Bezeichner wie z.B. PID für Patient Identification. Die Bedeutung der restlichen HL7-Elemente wird über die Position in der Nachricht bestimmt, wobei der Segment-Bezeichner dem Feld 0 entspricht. Eine Ausnahme bildet das Segment MSH (message header), in dem die Numerierung der Felder bei eins beginnt. Die leeren Felder sind für die Abzählung der Position wichtig. In der Beispielnachricht findet sich der Vorname des Patienten in Segment *PID*, Feld *5*, Komponente *2*. Der Patient heißt demzufolge *William*. Die zweite Komponente des neunten Feldes des MSH-Segments enthält den Ereignistyp. Das Ereignis *A01* codiert die Aufnahme des Patienten in das Krankenhaus.

Der Parser erzeugt nun HL7-Bezeichner der Gestalt *Segment_Feld_Komponente*, wobei die Angabe der Komponente optional ist. Damit sind die Konzepte von Nachrichtenmodell und DHE-Modell bezeichnet (siehe 3.1.1) und können in einer ADT-Map (vgl. 4.5) aufeinander abgebildet werden. Im wesentlichen entsprechen die HL7-Segmente *PID* (Patient identification) und *PV1* (Patient visit) den DHE-Entitäten *pmPA* (Patient) und *pmTT* (Contact of patient with healthcare institution). Am rechten Rand ist angegeben, welche Information abgebildet wurde.

```
pmPA[pa_lname, pa_birth, pa_sesso]                  // Identifikation Patient
pmTT[tt_icont]

EVN_2(
  event(MSH_9_2,
    &pmTT.tt_stada, &pmTT.tt_stahh,                 // Aufnahmedatum/-zeit
    &pmTT.tt_endda, &pmTT.tt_endhh))                // Entlassungsdatum/-zeit
PID_3_1(pmPA.pa_iperm)                              // Pat-Id
PID_5_1(pmPA.pa_lname)                              // Nachname
PID_5_2(pmPA.pa_fname)                              // Vorname
PID_7(pmPA.pa_birth=date2dhe())                     // Geburtsdatum
PID_8(pmPA.pa_sesso)                                // Geschlecht
PID_11_1(pmPA.pa_addr3)                             // Strasse
PID_11_3(pmPA.pa_city2)                             // Ort
PID_11_5(pmPA.pa_zipc2)                             // PLZ
PID_13(pmPA.pa_phon1)                               // Telefonnr.
PV1_1(pmTT.tt_iperm=pmPA, pmTT.tt_icont)            // Patient-Kontakt
PV1_3(pmTT.tt_letto)                                // Bett
```

Abbildung 40: Vereinfachte ADT-Map.

Neben dem Parser mußten noch zwei Funktionen für die Konversion des Datumformats (*date2dhe*) und für die Unterscheidung zwischen Aufnahme und Entlassung (*event*) implementiert werden. Damit konnte der Generator aus der Map eine HL7-Importschnittstelle für ADT-Nachrichten erzeugen. Der Parser und die Datumskonversion können dabei für beliebige HL7-Nachrichten wiederverwendet werden.

5.3.3 Aufwandseinsparung in Zahlen

In diesem Abschnitt wird dargelegt, in welchem Umfang durch das Generatorkonzept (4.4) die Effizienz der Schnittstellenentwicklung verbessert wurde. Der Gewinn liegt in der Aufwandseinsparung bei der Schnittstellenentwicklung. Wie kann man den Aufwand messen? In [Gumm98, 601-603] werden zwei Maße für den Aufwand einer Softwareentwicklung vorgestellt, die Anzahl der Textzeilen (man spricht auch von *LOC* = Lines of Code) und die Personenjahre (*PJ*). Ein PJ ist der Softwareumfang, den eine Person in einem Jahr erstellen kann. Als grobe Näherung wird 1 PJ = 5000 LOC angegeben. Zugleich wird aber darauf hingewiesen, daß beide Maße sehr ungenau sind. Es gibt z.B. mehr oder weniger kompakten Programmtext und mehr oder weniger produktive Software-Entwickler. Als günstig erwies sich allerdings die Tatsache, daß alle Entwicklungstätigkeiten von einer Person durchgeführt wurden. Die verschiedenen Entwicklungsarbeiten sind damit zumindest bedingt vergleichbar.

Für die Aufwandsmessung wurde die Zeilenmetrik gewählt, da die Textzeilen im Unterschied zu den PJ gezählt werden können. Um die Aufwandseinsparung durch den Generator ermitteln zu können, müssen die Verfahrensweisen "mit Generator" und "ohne Generator" verglichen werden. Grundlage für diesen Vergleich ist das verfeinerte Generatorkonzept (4.7). Zunächst werden die Textzeilen der Map (Eingabe des Generators) und des zugehörigen Controllers (Ausgabe des Generators) ermittelt. Die absolute Aufwandseinsparung ergibt sich dann aus den Controller-Zeilen abzüglich der Map-Zeilen. Stellt man die absolute, d.h. zeilenmäßige Aufwandseinsparung noch in das Verhältnis zu den Textzeilen des gesamten Importprogramms (Parser, Converter und Controller), so erhält man die relative bzw. prozentuale Aufwandseinsparung durch den Generator. Diese prozentuale Aufwandseinsparung betrug im Fall der BDT-Schnittstelle ca. 80 % und im Fall der HL7-Schnittstelle knapp 90 %. Dieser Unterschied ist darauf zurückzuführen, daß im Falle der BDT-Schnittstelle nur 17 Konzepte, im Falle der HL7-Schnittstelle dagegen 26 Konzepte in das DHE importiert wurden (17 versus 26 Tag-Funktionen gemäß 4.10). Der Nutzen wächst also mit der Anzahl der aus dem Importmodell importierten Konzepte.

Es konnte außerdem festgestellt werden, daß die absolute Aufwandseinsparung sowohl bei der BDT-Schnittstelle als auch bei der HL7-Schnittstelle zum DHE stets größer war als der zeilenmäßige Entwicklungsaufwand für den Generator. Das Generator-Verfahren hat sich nach der Zeilenmetrik also bereits für die BDT-Schnittstelle rentiert. Dies spricht dafür, daß der Generator aus wenigen Textzeilen in der Map viele Programmzeilen erzeugt, die nicht mehr vom Entwickler erstellt werden müssen.

Die Arbeitseinsparung ist aber nicht nur in Textzeilen zu messen. Ein wesentlicher Vorteil ist darin zu sehen, daß der Entwickler das Schnittstellenmodell (Map) *vor* dem Schnittstellenprogramm fertigstellen muß. Auf diese Weise werden Abbildungsprobleme sehr früh erkannt (siehe 5.1.2). Die Früherkennung von unüberwindbaren Modellunterschieden kann zu Entscheidungen führen (z.B. Wahl eines anderen Kommunikationsstandards für die Übertragung), die jede Programmierung überflüssig machen. Es muß außerdem berücksichtigt werden, daß viele manuell erstellte Zeilen tendenziell auch viele Fehler enthalten. Der Generator verkürzt folglich die Testphase bei der Schnittstellen-Entwicklung. Schließlich lassen sich wenige Textzeilen leichter pflegen als viele Textzeilen.

Die Zahlen werden i.a. nicht auf andere Bereiche übertragbar sein. Das Generatorprinzip hingegen schon. Bei sich wiederholenden Entwicklungstätigkeiten wie in unserem Fall die Entwicklung mehrerer Import- bzw. Schnittstellenprogramme zum DHE können Generatoren den Entwicklungsaufwand wesentlich reduzieren. Dabei ist im einzelnen zu prüfen, welche Tätigkeiten vom Entwickler zum Generator verlagert werden können (siehe 4.6). In dieser Arbeit wurde das Generatorprinzip auf die Entwicklung von Datenschnittstellen zum DHE angewandt, mit dem folgenden Erfolgsrezept:

> *Ergebnis: Die Trennung von Schnittstellen-Modellierung (Map) und Schnittstellen-Programmierung (Parser, Converter, Controller) spart Entwicklungsarbeit.*

Die angeführten Zahlen belegen, daß die Optimallösung, nämlich die vollständige Generierung der Schnittstellenprogramme aus dem Schnittstellenmodell (Map),

annähernd erreicht wurde. Parser und Converter müssen zwar nach wie vor entwickelt werden, verursachen aber im Vergleich zum Controller nur unwesentlichen Entwicklungsaufwand.

6 Diskussion

Ziel dieser Arbeit war die Demonstration und Erprobung der DHE-Middleware im Rahmen des europäischen Projektes HANSA. Zu diesem Zweck wurde das Gießener Tumordokumentationssystem (GTDS) mit dem DHE über eine BDT-Nachrichtenschnittstelle verbunden. Damit konnte sowohl die Anbindung einer existenten Anwendung an das DHE als auch die Integration zweier unabhängiger Anwendungen (GTDS und Anwendung für den Arbeitsplatz des Arztes) mittels DHE demonstriert werden. Unsere Erfahrungen haben gezeigt, daß die Unterstützung des DHE bei der Anwendungsintegration eher begrenzt ist. Auf der anderen Seite hat sich unsere Methodik, nämlich die Modellierung und Programmierung von Schnittstellen zu trennen, bewährt. Die Diskussion umfaßt folglich zwei Abschnitte, die jeweils das DHE und unsere Methodik in den Mittelpunkt stellen.

6.1 Hat das DHE eine Zukunft?

Als „Healthcare Middleware“ ist es die Aufgabe des DHE, die Integration von Anwendungen im Gesundheitswesen zu unterstützen. In diesem Abschnitt werden unsere Integrationserfahrungen mit dem DHE (siehe 5.1) diskutiert. Zu diesem Zweck werden die Erfahrungen der anderen Projekt-Partner den eigenen Erfahrungen gegenüber gestellt und es werden alternative Integrationsansätze zum DHE betrachtet. Abschließend wird auf der Basis der Erfahrungen und Alternativen die zukünftige Rolle und Funktion des DHE eingeschätzt.

Abhängig davon, ob die Integrationsabsicht bereits vor oder erst nach der Entwicklung einer Anwendung berücksichtigt wird, haben wir zwischen *Pre- und Postintegration* der Anwendung unterschieden. Preintegrierte Anwendungen verfügen bereits über Schnittstellen zu anderen Systemen. Eine Preintegration mit dem DHE erfordert insbesondere die Berücksichtigung der DHE-Programmierschnittstelle - dem API - bereits bei der Anwendungsentwicklung. Ergebnis 5.1.1 zufolge beschränkt sich die integrationsunterstützende Wirkung, d.h. die Middleware-Funktion des DHE auf die Preintegration von Anwendungsdaten. Die Postintegration der Anwendungsdaten und

der Anwendungsfunktionen des GTDS wurde folglich durch das DHE nicht unterstützt.

Der Bielefelder HANSA-Partner entwickelte zum Beispiel zwei einfache Anwendungen für die Anforderung und Befundung von EKG-Untersuchungen. Beide Anwendungen schreiben von vornherein ihre Daten in die DHE-Datenbank, womit die Integration mittels DHE schon demonstriert ist. Anwendungen, die das DHE für die Verwaltung von anwendungsübergreifenden Daten (Daten, die für mehr als eine Anwendung von Interesse sind) benutzen, sind über das DHE ohne weiteres datenintegriert. Privighitorita stellt daher grundsätzlich fest, daß das DHE für die Neuentwicklung eines datenintegrierten Informationssystems verwendet werden kann. Zugleich wird darauf hingewiesen, daß die Postintegration, d.h. die nachträgliche Integration von Anwendungen mittels DHE aufwendiger ist:

> [Privighitorita97] *„Present results of the described validation show that DHE can be used as a basis for developing HIS* (Healthcare Information System). ... *Integration of existing applications that are not DHE-based is basically possible, however more expensive.“*

Die Preintegration von Anwendungssystemen mittels DHE entspricht durchaus dem langfristigen Ziel von RICHE (siehe 3.1.6), wonach der Markt eine Fülle von DHE-basierten Anwendungen anbieten wird. Kurz-, mittel- und auch langfristig wird man aber nicht umhin können, auch DHE-fremde Anwendungen wie das GTDS in das Informationssystem zu integrieren. Die Problematik bei der DHE-Postintegration von Anwendungen besteht darin, daß die Anwendungsdaten im DHE repliziert werden müssen. Ähnliche Erfahrungen haben auch andere HANSA-Partner gemacht:

> [Endsleff97] *„Already developed systems cannot be used directly with the DHE, but will have to be migrated or replicated to some extent to the DHE.“*

Der Replikationsaufwand bestand vor allem darin, daß man große Datenmengen von der Anwendung in das DHE übertragen (Redundanzerstellung) und die entstehende

Datenredundanz kontrollieren (Redundanzkontrolle) mußte. Die DHE-Ingenieure entwickelten daraufhin einen Lösungsvorschlag für die Redundanzkontrolle zwischen Anwendung und DHE. Der Ansatz in [Ferrara97] verfolgt das Ziel, die Programmierschnittstelle des DHE – das API - nachträglich in die existenten Anwendungen - man spricht in diesem Zusammenhang auch von *Legacy systems* - einzubauen. Diese „anwendungsinterne“ Lösung erzeugt jedoch bei proprietären Systemen, deren Anwendungslogik nur vom Hersteller eingesehen werden darf, entsprechende Abhängigkeiten. Außerdem muß sich der Entwickler bei größeren Systemen wie dem GTDS in viele für die Schnittstelle zum DHE irrelevante Details der Anwendungslogik einarbeiten. Eine „anwendungsexterne“ Lösung der Replikationskontrolle, wie sie im Rahmen dieser Arbeit entwickelt wurde (siehe 4.3), kommt hingegen mit einem beschränkten Systemzugang aus und verursacht überdies sehr viel weniger Einarbeitungsaufwand.

Abbildung 41: Interne (links) und externe (rechts) Kontrolle der Datenreplikation zwischen Anwendung und DHE.

Andere HANSA-Partner vertreten ebenso die Meinung, daß man Eingriffe in gewachsene Anwendungssysteme aus Gründen der Machbarkeit und Wirtschaftlichkeit vermeiden sollte:

> [Weiler98] *"For economic and management reasons, it will be tried to avoid modifications of the source code of existing applications."*

Das Problem der Redundanzerstellung ist damit aber nach wie vor offen und wurde im Rahmen dieser Arbeit durch die automatische Generierung von DHE-Ladeprogrammen pragmatisch gelöst. Die Problematik wurde zudem sehr häufig umgangen, indem

entweder neue Anwendungen auf dem DHE entwickelt wurden oder die DHE-Demonstration auf wenige Anwendungsdaten beschränkt wurde. Tatsächlich muß die Rolle des DHE als zentrale und einzige Integrationsplattform überdacht werden. Für die Integration von Anwendungssystemen gibt es *Alternativen zum DHE*.

Das *Synapses*-Projekt [Grimson97] verfolgt z.B. das Konzept einer föderierten elektronischen Krankenakte (*Federated Healthcare Record*). Ein sogenannter Synapses-Server legt dazu eine globale Sicht über die in den Anwendungssystemen vorhandenen Datenquellen. Statt von Anwendungssytemen spricht man im Synapses-Umfeld auch von *Feeder-Systemen*. Mögliche Feeder-Systeme sind das Stationssystem, das Laborsystem etc. Eine Anfrage gegen die föderierte Datenbank wird vom Synapses-Server in Anfragen gegen die lokalen Datenbestände zerlegt. Die Feeder-Systeme müssen in der Lage sein, solche Anfragen zu beantworten. Die Ergebnisse der lokalen Anfragen werden vom Synapses-Server dann wieder zu einem Ergebnis zusammengefaßt. Welchen Vorteil hat Synapses gegenüber dem DHE? Gemäß [Duden88, 137-144] besteht eine Datenbank aus einem Datenspeicher und einem Verwaltungsprogramm. Der Synapses-Ansatz vermeidet einen zentralen Datenspeicher und damit auch die Datenredundanz. Der Föderationsansatz eignet sich somit sehr viel besser für die Postintegration von Anwendungsdaten als der zentrale DHE-Ansatz.

Noch weiter geht der *CORBA*-Ansatz (siehe 3.1.7). Während DHE und Synapses die anwendungsübergreifenden Daten zentral verwalten, ist bei CORBA auch die Datenverwaltung auf die Anwendungssysteme verteilt. Jedes System bzw. *Server object* definiert zu diesem Zweck seine eigenen Zugriffsmethoden in einer standardisierten Schnittstellenbeschreibungssprache, der *IDL*. Die anderen Systeme können diese Methoden über ein standardisiertes und für den Anwendungsentwickler transparentes Frage/Antwort-Protokoll, das *GIOP*, zugreifen. Welche Vorteile bietet der CORBA-Ansatz? Während in Synapses die Schnittstellen der Systeme festgelegt sind, können sie in CORBA frei definiert werden. Die einzelnen Systeme können also selbst entscheiden, wieviel Zugang sie zu ihren Daten gewähren wollen. Darüber hinaus muß das anfragende System erst zur Laufzeit die Schnittstellenbeschreibung des Zielsystems kennen, d.h. die Schnittstellen der Systeme können sehr einfach geändert

und angepaßt werden. Schließlich können mit demselben Prinzip nicht nur die Daten, d.h. die Datenverwaltungsprogramme sondern auch beliebige Funktionen der Anwendungssysteme integriert werden. CORBA unterstützt im Unterschied zum DHE also auch die Integration von Anwendungsfunktionen.

Die *Internet*-Technologie (siehe 3.1.7) geht analog zum CORBA-Ansatz von einer Verteilung der *Ressourcen*, d.h. der Daten und Programme aus. Die Unterstützung des Internet bei der Integration von Ressourcen geht allerdings weiter als bei den anderen Ansätzen. Beispielsweise können innerhalb des Web-Protokolls HTTP viele verschiedene Datenformate (Bilder, Video, Audio etc.) gekennzeichnet und somit auf Empfängerseite präsentiert werden. Das Internet entwickelt sich daher mehr und mehr zur zentralen Integrationsplattform für Informationssysteme, auch im Gesundheitswesen. Integrationstechnologien wie CORBA konvergieren zunehmend mit dem Internet [IW98].

Als vierte und letzte Integrationsalternative zum DHE sei noch der einfache Nachrichtenaustausch zwischen den Anwendungssystemen genannt. Betrachtet man die Integrationsansätze DHE, Synapses, CORBA und Internet, so stellt man fest, daß die Verteilung der Daten und Programme und damit auch die Unabhängigkeit der Systeme zunimmt. In diesem Sinne liefert der Nachrichtenaustausch die flexibelste Lösung. Es wurde aber schon in der Einführung darauf hingewiesen, daß die Integration der Systeme durch den Austausch von Nachrichten einigen Entwicklungsaufwand verursacht. Und genau darum geht es bei den Middleware-Ansätzen, den Integrationsaufwand zu reduzieren. Der Aufwand besteht häufig darin, daß viele einzelne Nachrichten verschickt werden müssen und die Nachrichtenformate nicht transparent sind. Die anderen Integrationsansätze kommen wegen der Verteiltheit der Systeme auch nicht umhin, Nachrichten zu übertragen. Allerdings ist der Nachrichtenaustausch für den Entwickler transparent. In CORBA werden zum Beispiel Frage- und Antwortnachrichten durch einfache Methodenaufrufe ersetzt. Ein Synapses-Server kann bei Veränderung der Patientendaten durch eine Anwendung automatisch Update-Nachrichten an die anderen Anwendungen verschicken und damit

eine eventuelle Datenredundanz zwischen den Anwendungssystemen kontrollieren (Replikationstransparenz nach [Tanenbaum95, 474-476]).

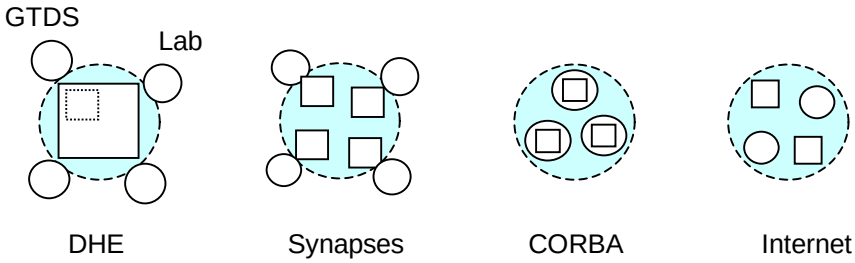

Abbildung 42: Integrationsansätze im Überblick. Rechteck und Kreis repräsentieren Programm (Anwendungen wie das GTDS) und Datenspeicher, der gestrichelte Kreis begrenzt den Integrationsbereich. Von links nach rechts ist die Integration von zentralem Datenspeicher mit angedeuteter Vervielfältigung existenter Daten (DHE), verteilten Datenspeichern (Synapses), Objekten (CORBA) und Ressourcen (Internet) dargestellt. CORBA und Internet unterstützen im Unterschied zu DHE und Synapses auch die Integration von Anwendungsfunktionalität (Kreise im Integrationsbereich).

Alle vorgestellten Integrationsalternativen weisen Vorteile gegenüber dem DHE auf. Angesichts dieser Tatsache stellt sich die Frage, welche Rolle das DHE bei so viel Konkurrenz spielen kann. Die Rolle der zentralen Integrationsplattform wird das DHE wegen der genannten Nachteile nicht wahrnehmen können. Die Internet-Technologie liefert aufgrund der Vielfalt an standardisierten Formaten zur Beschreibung und Übertragung von Ressourcen (Daten, Programme) die besten Voraussetzungen für eine Integrationsplattform. Auf der anderen Seite werden gerade in großen Systemen Komponenten benötigt, die Informationen (Adressen, Beschreibungen etc.) über andere Komponenten vorhalten. In einem CORBA-System gibt es zum Beispiel sogenannte Namensserver, bei denen sich die Server-Objekte mit Name und Adresse registrieren und über welche die Client-Objekte die Verbindung zu den Server-Objekten herstellen. In einem Klinikum werden Verzeichnisse benötigt, welche die Adressen, das Leistungsangebot und andere Informationen über die vorhandenen Leistungsstellen bereitstellen. Solche Informationen können in den Anwendungssystemen häufig nicht untergebracht werden, aus Gründen der

Zuständigkeit (die Adressvermittlung ist nicht die Funktion der Anwendung), des Datenmodells (im Anwendungsmodell sind diese Informationen nicht vorgesehen) und des Speicherplatzes. Das DHE kann alle diese Dinge liefern. In einem Synapses-System wird das DHE zum Beispiel als ein mögliches Feeder-System diskutiert. Es sind also die Verzeichnisdienste, die das Überleben des DHE als Middleware sichern können. Auf der anderen Seite kann das DHE auch als Componentware eine Zukunft haben. Beispielsweise muß jede Leistungsstelle intern mehr oder weniger Ressourcen wie Personal, Medizin Technik und Materialien verwalten. Die Resource-Komponente des DHE kann an dieser Stelle nützliche Dienste leisten.

6.2 Diskussion des Generatorkonzepts

In diesem Abschnitt wird unsere Methodik der Schnittstellen-Generierung zur Diskussion gestellt. Die Schnittstellenfunktion war in diesem Zusammenhang die Umwandlung der Datenrepräsentation eines gegebenen Inhalts. Es werden vergleichbare Lösungen vorgestellt und gegen die eigene Lösung abgegrenzt. Ferner wird überlegt, ob man weitere Anwendungsfelder für das entwickelte Verfahren erschließen kann.

Das Schlüsselkonzept dieser Arbeit war die Generierung von Datenschnittstellen zum DHE (siehe 4.4.). Motiviert wurde diese Methodik durch die Tatsache, daß mehrere solche Schnittstellen benötigt wurden und die Entwicklung dieser Schnittstellen mehr Finger- als Kopfarbeit erforderte. Daraus erwuchs die Idee, Finger- und Kopfarbeit voneinander zu trennen. Die "Kopfarbeit" bestand vor allem in der Abbildung der Datenmodelle. Konkret stellte sich zum Beispiel die Frage, ob und wie die Histologiebefunde einer BDT-Nachricht im DHE-Modell repräsentiert sind. Die "Fingerarbeit" bestand überwiegend in der Implementierung der gefundenen Modellabbildung, d.h. der Steuerung des Datenflusses von der Datendatei in die DHE-Datenbank. Das Erfolgsrezept dieser Trennung war: Der Entwickler konnte sich auf die inhaltliche Arbeit konzentrieren, während die Fingerarbeit von einem Programm - dem *Generator* - durchgeführt wurde. Die Modellabbildung wurde zu diesem Zweck von dem Entwickler in einer eigens dafür definierten Schnittstellenbeschreibung - der

Map - formuliert, woraus der Generator dann eine entsprechende Datenschnittstelle erzeugte.

Eine vergleichbare Lösung zur Reduzierung des Schnittstellenaufwands wurde im Umfeld des pro7-Kommunikationsservers entwickelt [Gesell97]. Kommunikationsserver leisten unter anderem die Übersetzung zwischen verschiedenen Nachrichtenprotokollen (siehe 3.1.7). Zur Vereinfachung dieser Übersetzung wurde eine Schnittstellenbeschreibungssprache, die Protocol Definition Language (PDL), definiert. Die *PDL* ermöglicht die Abbildung von beliebigen und unter Umständen proprietären Nachrichtenprotokollen auf das pro7-Protokoll, das eine Erweiterung des HL7-Standards (5.3.2, deutsche Anpassungen, Übertragung von Röntgenbilder) darstellt. Aus der PDL-Beschreibung werden dann automatisch entsprechende Protokoll-Converter - zu und von pro7 - generiert.

Eine weitere Lösung zur Verringerung des Schnittstellenaufwands stellt die standardisierte Sprache *XSLT* (eXtensible Styling Language for Transformation) dar [W3C99]. Mit XSLT wird die Umwandlung sogenannter XML-Dokumente [W3C98] in beliebige Datenformate beschrieben. Ein XML-Dokument ist eine Textdatei, in der einzelne Elemente durch spitz geklammerte Markierungen begrenzt sind. Die Markierungen sind in XML (eXtensible Markup Language) frei definierbar. Das folgende XML-Dokument enthält z.B. eine Markierung <befund>, die durch ein code-Attribut näher beschrieben wird. Durch die Unterscheidung einer Anfangs- und Endmarkierung (<bericht> und </bericht>) können Elemente beliebig verschachtelt werden:

```
<bericht>
        Der Patient <patient>Maier</patient> leidet unter einem Melanom
        im Bereich der <lokalisation>rechten Stirn</lokalisation> mit der
        Morphologie <befund code="ICD-O">M-8091/3</befund> und der
        TNM-Klassifikation <befund code="TNM">pT1N0M0</befund>.
</bericht>
```

XSLT definiert nun seinerseits XML-Markierungen, mit denen man die Umwandlung beliebiger XML-Dokumente beschreiben kann. Beispielsweise definiert XSLT eine Markierung <value-of>, die den Wert eines Elementes aus dem Quelldokument

ausgibt. Die Markierung <for-each> erlaubt die Bearbeitung der in dem select-Attribut identifizierten Menge von Elementen. Die folgende XSLT-Anweisung gibt z.B. die Texte aller TNM-codierten Befunde aus:

```
<for-each select="befund[@code='TNM']">
        <value-of select="text()"/>
</for-each>
```

Wie können die vorgestellten Lösungsansätze für die Reduzierung des Schnittstellenaufwands nun gegeneinander abgegrenzt werden? Das Prinzip ist jedesmal dasselbe und besteht in der Schaffung einer Schnittstellenbeschreibungssprache (Map, PDL, XSLT), die viele Details einer Programmiersprache verbirgt. Die Programmiersprache umfaßt prodezurale Elemente wie Schleifen zur Handhabung von Wiederholungen (mehrere Tumordiagnosen in einer BDT-Nachricht), bedingte Anweisungen zur Handhabung von Fallunterscheidungen (Interpretation einer ADT-Nachricht als Aufnahme oder Entlassung), Zwischenspeicher etc. Eine deklarative Beschreibungssprache kann viele prozedurale Konstrukte zu mächtigeren Konstrukten zusammenfassen, was die Schnittstellenimplementierung vereinfacht.

Im Unterschied zur Map umfassen die Schnittstellenbeschreibungen PDL und XSLT aber nach wie vor prozedurale Elemente, d.h. die Trennung zwischen Modellierung und Programmierung wurde in diesen Ansätzen nicht ganz vollzogen. Die Map hingegen enthält ausschließlich Elemente, die für die Modellabbildung notwendig sind: Konzeptbezeichner und Konstrukte zur Ausbildung von Konzeptbeziehungen. Welche Vorteile ergeben sich nun aus der konsequenten Trennung von Schnittstellen-modellierung und Schnittstellenprogrammierung? Zum einen wird die Schnittstellen-beschreibungssprache und damit auch die Implementierung des Generators einfacher. Der Entwurf der Map-Sprache war tatsächlich pragmatisch motiviert. Entscheidend ist aber die Tatsache, daß eine Kombinationsmöglichkeit zwischen der Beschreibungssprache (Map) und der Programmiersprache (C++) geschaffen wurde. Da die Modelle nicht immer eins zu eins aufeinander abgebildet werden können, wurden Funktionen eingeführt, die in einer Programmiersprache implementiert und in der Map für die Abbildung benutzt werden können. Solche Funktionen können

Fallunterscheidungen, Datenkonversionen und andere Einzelheiten transparent machen. Auf diese Weise werden die Vorteile einer Beschreibungssprache (prägnante Formulierung) mit den Vorteilen einer Programmiersprache (Vielfalt der Datenverarbeitung: Speicherung in einer Datenbank, weitere Zerlegung von Texten) kombiniert.

Das erarbeitete Schnittstellenverfahren kann nun wie folgt generalisiert werden (Abbildung 43). M1, M2 bezeichnen zwei Datenmodelle und D1, D2 entsprechende Daten (Modellinstanzen). Der Zugang zu den Daten erfolgt durch gegebene Programme, welche die Organisation der Daten verbergen (Datenabstraktion). Um D1 und D2 ineinander zu übersetzen, erstellt der Entwickler ein Schnittstellenmodell SM, indem die Konzepte von M1 und M2 aufeinander abgebildet sind. Der Schnittstellengenerator SG erzeugt aus dem Schnittstellenmodell ein Schnittstellenprogramm SP, das beliebige Modellinstanzen ineinander überführt. Die Generierbarkeit der Schnittstellenprogramme ergibt sich also vor allem durch die Trennung von Programm- und Modellebene.

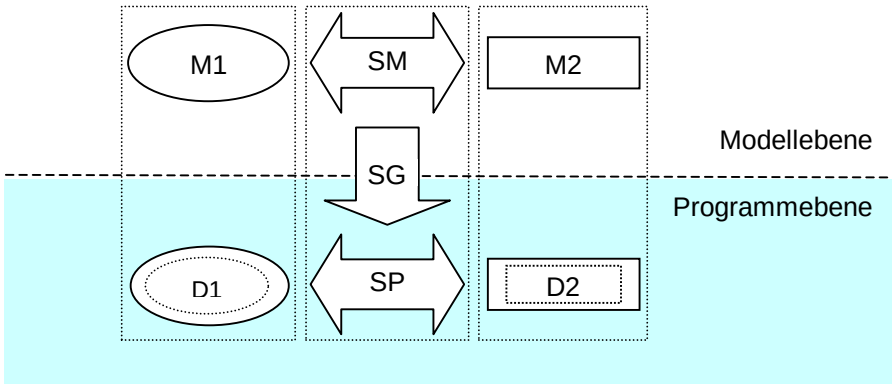

Abbildung 43: Trennung von Modell- und Programmebene bei der Schnittstellenentwicklung. Die Organisation der Daten D1, D2 (gestrichelte Linie) ist durch entsprechende Zugriffsprogramme (durchgezogene Linie) gekapselt. Der Schnittstellengenerator SG erzeugt aus dem Schnittstellenmodell SM (Abbildung der Modelle M1, M2) ein Schnittstellenprogramm SP, das beliebige Modellinstanzen bzw. Daten D1, D2 ineinander übersetzt.

Für das generalisierte Verfahren lassen sich jetzt neue Anwendungsfelder erschließen: D1 sei eine relationale Datenbank, auf die mittels JDBC (Java Data Base Connectivity) zugegriffen wird. D2 sei eine XML-Datei, die mit einem Java-Parser bearbeitet wird. Ein Generator SG kann dann aus dem Schnittstellenmodell SM entsprechende Java-Austauschprogramme SP zwischen der Datenbank und den XML-Nachrichten erzeugen.

Die Auftrennung der Schnittstellenentwicklung in inhaltliche (Semantic mapping) und nicht-inhaltliche (Syntactic mapping) Aufgaben wird in [Bürkle99] als eine sinnvolle und allgemeine Verfahrensweise beschrieben und am Beispiel der GTDS-DHE-Verbindung illustriert (Abbildung 44). Unter Semantik werden in diesem Zusammenhang die Beziehungen zwischen den Modellkonzepten (innerhalb eines Modells, zwischen verschiedenen Modellen) verstanden. Alle anderen Aufgaben fallen in den Bereich Syntax. Unter EPR (Electronic Patient Record) ist ein Datenmodell für

die elektronische Krankenakte zu verstehen. Die Verbindung der EPR-Modelle GTDS und DHE erfolgt über das BDT-Nachrichtenmodell.

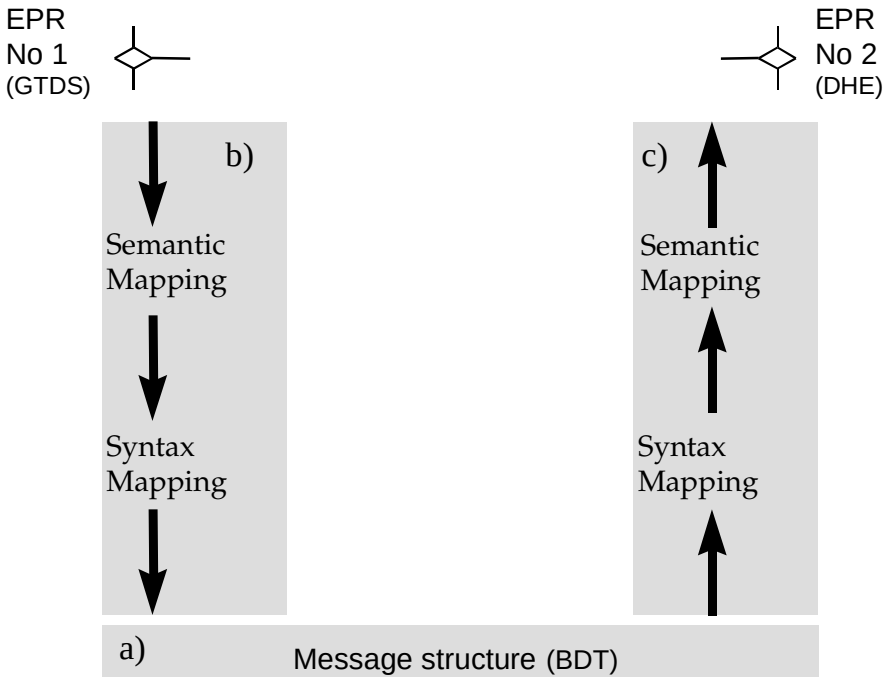

Abbildung 44: Die Entwicklung der BDT-Exportschnittstelle auf seiten des GTDS (links) und die Entwicklung der BDT-Importschnittstelle auf seiten des DHE (rechts) werden jeweils in inhaltliche (semantic mapping) und nicht-inhaltliche (syntactic mapping) Tätigkeiten aufgetrennt. Entnommen aus [Bürkle99].

Zwei Vorteile ergeben sich aus der Trennung von Syntax und Semantik. Zum einen werden Abbildungsprobleme zwischen den verschiedenen Modellen (GTDS-BDT, BDT-DHE, siehe 5.1.2) frühzeitiger erkannt, wenn die inhaltlichen Aufgaben abgeschlossen sind bevor die nicht-inhaltlichen Aufgaben in Angriff genommen werden. Inhaltliche Probleme können weitreichende Konsequenzen haben bis hin zur Ersetzung des Nachrichtenmodells, das die Inhalte nicht transportieren kann. Zum anderen entsteht der Wunsch, die nicht-inhaltlichen Aufgaben transparent zu machen. Ein möglicher Weg zu mehr Transparenz wurde im Rahmen dieser Arbeit demonstriert und besteht in der automatischen Generierung von DHE-Importschnittstellen (Syntactic mapping) aus einer Map (Semantic mapping). Ebenso gibt es auf seiten des GTDS Bestrebungen zu mehr Transparenz. Hier wurde zum Beispiel ein API

(Application Programming Interface) zum GTDS entwickelt, das dem Schnittstellenentwickler beim Zugriff auf das GTDS die Navigation durch die GTDS-Datenbank (Existenzprüfungen, Zusammenführung von Daten) erspart.

7 Zusammenfassung

Aufgabe des europäischen Projektes HANSA (1996-1998) war die Erprobung und Demonstration der DHE-Middleware mit dem Ziel, eine standardisierte Middleware-Architektur für Informationssysteme im Gesundheitswesen zu schaffen. Als Middleware hat das DHE (Distributed Healthcare Environment) die Funktion, die Integration von Anwendungssytemen (Station, Labor, Apotheke, klinisches Krebsregister etc.) zu unterstützen. Diese Arbeit entstand im Rahmen des HANSA-Projektes mit der Zielsetzung, das Gießener Tumordokumentationssystem (GTDS) an das DHE anzubinden. Andere DHE-basierte Anwendungen wie das Arbeitsplatzsystem des Arztes auf der Station können somit auf die Informationen im GTDS zugreifen. Das GTDS unterstützt die Dokumentation von Tumorkrankheiten in klinischen Krebsregistern und umfaßt Befunde (Labor, Histologie, CT), Diagnosen, Behandlungen (Operation, Chemotherapie, Bestrahlung), Arztbriefe und andere Informationen. Das GTDS befindet sich heute in über 30 deutschen Tumorzentren im Einsatz.

Das DHE stellt im wesentlichen eine Datenbank für anwendungsübergreifende Daten dar, auf die der Entwickler in einer transparenten Art und Weise (der Entwickler muß z.B. den Ort der Datenbank nicht kennen) zugreifen kann. Das Datenmodell des DHE zeichnet sich durch ein Maßnahmekonzept aus, wobei eine Maßnahme (z.B. eine Untersuchung) i.a. durch verschiedene Personen angefordert, eingeplant, durchgeführt und befundet wird. Bei der Verbindung von GTDS und DHE wurden folgende DHE-Schwachstellen identifiziert:

1) Die Daten des GTDS mußten im DHE vervielfältigt werden. Die Übertragung der Daten vom GTDS zum DHE erforderte die Entwicklung mehrerer Schnittstellen (Austauschprogramme).

2) Das Datenmodell des DHE erlaubte keine angemessene Strukturierung der Tumordaten. Beispielsweise können aus einem Tumor mehrere Metastasen hervorgehen, wobei zu jeder Metastase mehrere Beurteilungen zu speichern sind. Die Zusammengehörigkeit der Konzepte Tumor, Metastase und Beurteilung der

Metastase konnte im DHE-Modell nicht repräsentiert werden. Darüber hinaus war das Konzept des Hauptversicherten im DHE-Modell nicht vorgesehen.

3) Der Zugriff auf das DHE erforderte vom Entwickler die Navigation durch ein über 200 Konzepte (Patient, Maßnahme, klinische Daten etc.) und zahlreiche Konzeptbeziehungen umfassendes Modell mit überwiegend primitiven Funktionen. Die Anwendungsentwicklung auf dem DHE war damit relativ aufwendig.

Die Erfahrungen aus dieser Arbeit konnten für die Verbesserung des DHE sowie des entsprechenden Middleware-Standards HISA (Healthcare Information System Architecture, CEN/TC251/PT013) genutzt werden. Im einzelnen wurden folgende Verbesserungen erzielt:

1) Es wurde ein Verfahren zur automatischen Generierung von DHE-Schnittstellen entwickelt. Mit diesem Verfahren ist es möglich, gegebene Daten sehr rasch in das DHE zu importieren. Das Verfahren reduzierte den Aufwand bei der GTDS-Anbindung um mehr als 80%, wurde von anderen HANSA-Partnern eingesetzt und ist mittlerweile zusammen mit dem DHE verfügbar. Das Verfahren kann generalisiert und auf andere Bereiche übertragen werden.

2) Die Datenmodelle von DHE und HISA wurden jeweils so erweitert, daß beliebige Beziehungen zwischen klinischen Daten formuliert werden können. Damit genügen DHE und HISA nun auch den Anforderungen eines Tumordokumentationssystems.

3) Die Entwicklerschnittstelle des DHE wurde um einige Funktionen (Initialisierungsfunktionen, SQL-Funktion) bereichert, welche die Entwicklungen auf dem DHE beschleunigen. Fallunterscheidende Funktionen, die z.B. bei der Existenz eines Patienten im DHE den Datensatz aktualisieren und andernfalls einen neuen Datensatz anlegen, sind darüber hinaus wünschenswert.

Wie kann nun die Rolle des DHE bei der Anwendungsintegration definiert werden? Die Integrationsaufgabe in einem Informationssystem besteht häufig darin, gegebene Ressourcen, d.h. Daten und Programme zusammenzuführen (Postintegration). Eine

Vervielfältigung der Ressourcen sollte aus Aufwandsgründen vermieden werden. Andere Integrationsansätze wie CORBA (Common Object Request Broker Architecture) oder das Internet, die von einer Verteilung der Ressourcen ausgehen, sind ggfs. gegenüber dem DHE zu bevorzugen. Mittels Internet-Technologie können zum Beispiel nicht nur die Daten, sondern auch die Funktionen des GTDS (Erzeugung von Übersichtsberichten etc.) am Arbeitsplatz des Arztes relativ einfach zugänglich gemacht werden.

Das DHE kann hingegen all diejenigen Daten verwalten, die in den existenten Anwendungen bisher nicht repräsentiert sind. Beispiele für solche Daten sind Adressverzeichnisse (Adressen der Leistungsstellen eines Klinikums) oder Codeverzeichnisse (ICD, TNM). Das Maßnahmekonzept des DHE kann für die Implementierung eines noch nicht vorhandenen Anforderungssystems verwendet werden. Beim Aufbau eines neuen Informationssystems könnte das DHE sogar die komplette Datenverwaltung übernehmen. Das DHE kann also die Lücken in existenten Informationssystemen schließen und somit eine begrenzte Middleware-Funktion erfüllen.

8 Literatur

[Aho88] Aho AV, Sethi R, Ullman JD: Compilerbau Teil 1. Addison-Wesley Verlag Bonn 1988.

[Altmann95] Altmann U, Katz FR, Haeberlin V, Willems C, Dudeck J: GTDS – An Oncology Workstation. Prokosch HU, Dudeck J (Hrsg) Hospital Information Systems: Design and Development Characteristics; Impact and Future Architecture. Elsevier Amsterdam 1995, 117-129.

[Altmann98] Altmann U, Wächter W, Müller A, Funken O, Sembritzki J, Lichtner F, Gehlen E, Dudeck J: Datenaustausch in der Onkologie mittels BDT. Greiser E, Wischnewsky (Hrsg) Methoden der Medizinischen Informatik, Biometrie und Epidemiologie in der modernen Informationsgesellschaft. MMV Medien & Medizin Verlag München 1998, 247-250.

[Bangemann94] Bangemann M, da Fonseca EC, Davis P et al: Europe and the global information society, Recommendations to the European Council. Brussels May 1994.

[Bürkle99] Bürkle T, Schweiger R, Altmann U, Holena M, Blobel B, Dudeck J: Transferring Data from One EPR to Another: Content - Syntax - Semantic. Methods of Information in Medicine Vol 38 No 4-5, Dec 1999, 321-325.

[CEN95] CEN Comité Européen de Normalisation: Healthcare Information Framework (HIF). CEN/TC251/PT010, 1995.

[CEN97] CEN Comité Européen de Normalisation: Healthcare Information Systems Architecture Part 1 (HISA) - Healthcare Middleware Layer. CEN/TC251/PT013, 1997.

[DeJesus96] DeJesus EX: The Middleware Riddle. Byte Magazine, Apr 1996, www.byte.com.

[Dolgicer99] Dolgicer M: Building a Middleware Platform. Component Strategies, March 1999, 40-44.

[Dudeck95] Dudeck J: Communication Standards in Healthcare. Prokosch HU, Dudeck J (Hrsg) Hospital Information Systems: Design and Development Characteristics; Impact and Future Architecture. Elsevier Amsterdam 1995, 17-36.

[Duden88] Lektorat des B.I.-Wissenschaftsverlags, Engesser H, Claus V, Schwill A: Duden Informatik, Ein Sachlexikon für Studium und Praxis. Dudenverlag Mannheim 1988.

[Endsleff97] Endsleff F, Per Loubjerg E: System integrational and migrational concepts and methods within Healthcare. Pappas C, Maglaveras N, Scherrer JR (Hrsg) Medical Informatics Europe '97. IOS Press 1997, 1-5.

[Ferrara95] Ferrara FM: Healthcare Advanced Networked System Architecture (HANSA). Commission of the European Communities, Project Number HC 1019 HC, 1995.

[Ferrara96a] Ferrara FM, Sottile PA: DHE middleware ver 1.0 API of the services. Gestione Sistemi per l'Informatica (GESI) Rome 1996.

[Ferrara96f] Ferrara FM, Sottile PA: The DHE middleware Functional view. Gestione Sistemi per l'Informatica (GESI) Rome 1996.

[Ferrara96i] Ferrara FM, Sottile PA: The DHE middleware Information view. Gestione Sistemi per l'Informatica (GESI) Rome 1996.

[Ferrara97] Ferrara FM: Healthcare Information Systems Architecture. Dudeck J, Blobel B, Lordieck W, Bürkle T (Hrsg) New Technologies in Hospital Information Systems. IOS Press Berlin 1997, 1-9.

[Fowler97] Fowler M, Kendall S: UML Distilled, Applying the Standard Object Modeling Language. Addison-Wesley Object Technology Series 1997.

[Frandji95] Frandji B: RICHE Architecture Overview. Groupe RICHE Technical Committee, Utrecht Nov 1995.

[Gesell97] Gesell B, Schmal T: Das pro7-System als Basis für den Aufbau eines verteilten krankenhausweiten Informationssystems. Hasselbring W (Hrsg) Erfolgsfaktor Softwaretechnik für die Entwicklung von Krankenhausinformationssystemen. Krehl Verlag Münster 1997, 37-44.

[Grimson97] Grimson W, Sottile PA: Synapses in the Context of Healthcare Information Systems. Dudeck J, Blobel B, Lordieck W, Bürkle T (Hrsg) New Technologies in Hospital Information Systems. IOS Press Berlin 1997, 30-39.

[Gumm98] Gumm HP, Sommer M: Einführung in die Informatik. Oldenbourg Verlag München Wien 1998.

[Hannah95] Hannah KJ, Edwards MJ: Design Issues for Nursing Information Systems. Prokosch HU, Dudeck J (Hrsg) Hospital Information Systems: Design and

Development Characteristics; Impact and Future Architecture. Elsevier Amsterdam 1995, 131-151.

[Health-Comm97] Health-Comm GmbH, Software für das Gesundheitswesen: Cloverleaf ® Leistungsbeschreibung. Sep 1997.

[Heitmann99] Heitmann KU, Blobel B, Dudeck J: HL7 Kommunikationsstandard in der Medizin, Kurzeinführung und Information. Alexander Mönch Verlag Köln 1999.

[Hripcsak97] Hripcsak G: IAIMS Architecture. Journal of the American Medical Informatics Association, Vol 4, No 2, Mar/Apr Suppl 1997, 20-30.

[IEEE97] IEEE Institute of Electrical and Electronics Engineers: Draft Standard for Information Technology - Software Reuse Life Cycle Processes. Document P1517/Draft 5, Jul 1997, rsc.asset.com.

[IW98] IW Information Week: Corba und Java im Schulterschluß. Information Week, 24/98, 48-48.

[Masys98] Masys DR, Baker DB, Barnhart R, Buss T: PCASSO: A Secure Architecture for Access to Clinical Data via the Internet. Cesnik B, McCray AT, Scherrer JR (Hrsg) Proceedings of the 9th World Congress on Medical Informatics. IOS Press Amsterdam 1998, 1130-1134.

[OMG98] OMG Object Management Group: The CORBAmed Roadmap. Feb 1998.

[Privighitorita97] Privighitorita R, Sobottka HG, Lordieck W: Experiences with DHE: An Order Entry and Result Reporting Case Study. Dudeck J, Blobel B, Lordieck W, Bürkle T (Hrsg) New Technologies in Hospital Information Systems. IOS Press Berlin 1997, 74-79.

[Prokosch94] Prokosch HU: Ein Referenzmodell zur Charakterisierung informations- und wissensverarbeitender Funktionen innerhalb von Krankenhaus-Informationssystemen. Kunath H, Lochmann U, Straube R, Jöckel KH, Köhler CO (Hrsg) Medizin und Information. MMV Medien & Medizin Verlag München 1994, 44-48.

[ROKD98] ROKD, Produkte Gesundheitswesen: Leistungsbeschreibung des Kommunikationsservers pro7. Stand July 1998.

[Roundtable98] Broy M, Deimel A, Henn J, Koskimies K, Plasil F, Pomberger G, Pree W, Stal M, Szyperski C: What characterizes a (software) component? Virtual

roundtable in a theme issue of Software – Concepts & Tools on componentware (guest editor: W. Pree), Springer-Verlag, Heidelberg/New York, June 1998, www.informatik.uni-konstanz.de/~pree/SelectedPublications.html.

[Ruan98] Ruan W, Bürkle T, Dudeck J: Context Sensitive Information at the Clinical Workstation - the Role of the Medical Data Dictionary. Greiser E, Wischnewsky (Hrsg) Methoden der Medizinischen Informatik, Biometrie und Epidemiologie in der modernen Informationsgesellschaft. MMV Medien & Medizin Verlag München 1998, 214-217.

[Rymer96] Rymer JR: The Muddle in the Middle. Byte Magazine, Apr 1996, 67-70.

[Schweiger97] Schweiger R, Bürkle T, Dudeck J: Postintegration of a tumor documentation system into a HIS via middleware. Pappas C, Maglaveras N, Scherrer JR (Hrsg) Medical Informatics Europe '97. IOS Press 1997, 6-9.

[Schweiger98] Schweiger R, Bürkle T, Dudeck J: Testing a healthcare specific middleware. Greiser E, Wischnewsky (Hrsg) Methoden der Medizinischen Informatik, Biometrie und Epidemiologie in der modernen Informationsgesellschaft. MMV Medien & Medizin Verlag München 1998, 185-188.

[Siegel96] Siegel J: CORBA Fundamentals and Programming. John Wiley & Sons 1996.

[Spahni98] Spahni S, Scherrer JR, Sauquet D, Sottile PA: Middleware for Healthcare Information Systems. Cesnik B, McCray AT, Scherrer JR (Hrsg) Proceedings of the 9th World Congress on Medical Informatics. IOS Press Amsterdam 1998. 212-216.

[Spector98] Spector A: Brücken bauen statt neu programmieren. Information Week, 24/98, 44-47.

[Stroustrup87] Stroustrup B: The C++ Programming Language. Addison-Wesley Amsterdam 1987.

[Tanenbaum95] Tanenbaum AS: Moderne Betriebssysteme. Hanser Verlag München 1995.

[W3C98] W3C World Wide Web Consotium: Extensible Markup Language (XML) 1.0. W3C Recommendation 10 Feb 1998.

[W3C99] W3C World Wide Web Consotium: XSL Transformations (XSLT) Version 1.0. W3C Recommendation 16 Nov 1999.

[Weiler98] Weiler F, Jora S, Wiktor O: DIFF: an electronic patient record for the Hospital Princesse Marie-Astrid in Differdange. Workshop on Standard Hospitals Information Systems Architecture (HISA) Results of the HANSA Project. Barcelona Feb 1998.

[Wilkes99] Wilkes L: Legacy Componentization and Wrapping. Component Strategies, Feb 1999, 50-57.

[Williamson97] Williamson M: Component-Speak: A Glossary. CIO Magazine, Mar 1, 1997, www.cio.org.

[Wingert79] Wingert F: Medizinische Informatik. Richter L, Stucky W (Hrsg) Leitfäden der angewandten Informatik. Teubner Stuttgart 1979.

9 Veröffentlichungen

1) Schweiger R, Bürkle T, Dudeck J: Postintegration of a tumor documentation system into a HIS via middleware. Pappas C, Maglaveras N, Scherrer JR (eds) Medical Informatics Europe '97. IOS Press 1997, 6-9.

2) Schweiger R, Bürkle T, Dudeck J: Plug and Play Integration into a Healthcare Information System. Dudeck J, Blobel B, Lordieck W, Bürkle T (eds) New Technologies in Hospital Information Systems. IOS Press Berlin 1997, 90-94.

3) Schweiger R, Bürkle T, Hölzer S, Tafazzoli AG, Dudeck J: Plug and Play - Fiction or Reality? Cesnik B, McCray AT, Scherrer JR (eds) Proceedings of the Ninth World Congress on Medical Informatics. IOS Press Amsterdam 1998, 999-1001.

4) Schweiger R, Bürkle T, Dudeck J: Testing a healthcare specific middleware. Greiser E, Wischnewsky (Hrsg) Methoden der Medizinischen Informatik, Biometrie und Epidemiologie in der modernen Informationsgesellschaft. MMV Medien & Medizin Verlag München 1998, 185-188.

5) Schweiger R, Bürkle T, Ruan W, Dudeck J: XML: Evolution towards a structured electronic patient record. Moorman PW, van der Lei J, Musen MA (eds) Electronic Patient Records in Medical Practice, Proceedings of IMIA Working Group 17, Rotterdam 8-10 October 1998, 264-267.

6) Holena M, Schweiger R, Blobel B, Bürkle T, Altmann U, Dudeck J: Communication between a clinical front end and an oncological EHR using a middleware based on a European prestandard. Moorman PW, van der Lei J, Musen MA (eds) Electronic Patient Records in Medical Practice, Proceedings of IMIA Working Group 17, Rotterdam 8-10 October 1998, 93-96.

7) Hölzer S, Hausen ME, Schweiger R, Dudeck J: Daten zum Anaplastischen Schilddrüsenkarzinom in Deutschland. Strahlentherapie und Onkologie 1998, 174:43-43.

8) Schweiger R, Bürkle T, Köhn F, Dudeck J: How to structure clinical information: A practical example. Kokol P, Zupan B, Stare J, Premik M, Engelbrecht R (eds) Medical Informatics Europe '99. IOS Press Amsterdam 1999, 20-24.

9) Bürkle T, Schweiger R, Altmann U, Holena M, Blobel B, Dudeck J: Transferring Data from One EPR to Another: Content - Syntax - Semantic. Methods of Information in Medicine Vol 38 No 4-5, Dec 1999, 321-325.

10) Altmann U, Wächter W, Tafazzoli AG, Katz FR, Schweiger R, Dudeck J: A model for integration and continuous development of standards for tumor documentation using relational database techniques and extensible markup language. Kokol P, Zupan B, Stare J, Premik M, Engelbrecht R (eds) Medical Informatics Europe '99. IOS Press Amsterdam 1999, 895-898.

11) Schweiger R, Bürkle T, Tafazzoli AG, Hölzer S, Dudeck J: Migration from legacy data to XML, Experiences with drug information sources at Gießen university. Lorenzi NM (ed) The Annual Symposium of the American Medical Informatics Association '99. Hanley & Belfus, Inc. Medical Publishers Philadelphia 1999, 1157-1157.

12) Altmann U, Tafazzoli AG, Noelle G, Huybrechts T, Schweiger R, Wächter W, Dudeck J: Combining dictionary techniques with eXtensible Markup Language (XML) - Requirements to a new approach towards flexible and standardized documentation. Lorenzi NM (ed) The Annual Symposium of the American Medical Informatics Association '99. Hanley & Belfus, Inc. Medical Publishers Philadelphia 1999, 12-16.

13) Bürkle T, Altmann U, Schweiger R, Tafazzoli A, Michel A, Ruan W, Joch J, Dudeck J: Intranetanwendungen am Klinikum der Justus-Liebig-Universität

Gießen. Jäckel A (Hrsg) Telemedizinführer Deutschland - Ausgabe 2000, Deutsches Medizin Forum, Bad Nauheim 1999, 310-318.

14) Schweiger R, Bürkle T, Dudeck J: XML: A way to structured information sources. GMDS 1999, 403-403.

15) Hölzer S, Tafazzoli AG, Schweiger R, Altmann U, Dudeck J: Practical aspects of clinical documentation as the basis for quality management and outcome analysis, GMDS 1999, 202-202.

16) Schweiger R, Tafazzoli AG, Dudeck J: Using XML for flexible data entry in healthcare, Example use for pathology. XML Europe 2000 Conference Proceedings, 357-362.

17) Schweiger R, Bürkle T, Hölzer S, Dudeck J: XML structured clinical information: A practical example. Medical Informatics Europe (MIE) 2000 in Hannover.

18) Hölzer S, Tafazzoli AG, Schweiger R, Altmann U, Dudeck J: Structured Clinical Documentation for the Assessment of Medical Care. MIE2000.

19) Schweiger R, Bürkle T, Hölzer S, Dudeck J: Management of clinical XML documents: A pragmatic approach. Annual Symposium of the American Medical Informatics Association 2000 in Los Angeles.

20) Hölzer S, Tafazzoli AG, Schweiger R, Altmann U, Dudeck J: Structured Clinical Documentation for the Assessment of Medical Care. MIE 2000.

10 Anhang

10.1 Mapsprache

Mapsprache in Backus-Naur-ähnlicher Form. Links von ::= befinden sich die Nichtterminale. Abgesehen vom senkrechten Strich, der alternative Produktionsregeln trennt, sind alle anderen Symbole Terminale oder Terminalklassen. Eine Konstante ist in doppelte Anführungszeichen zu setzen.

```
MAP ::= ENTITY-LIST TAG-LIST end-of-file
ENTITY-LIST ::= entity [ ATTRIBUTE-LIST ] ENTITY-LIST |
                        entity [ ATTRIBUTE-LIST ) ENTITY-LIST | nil
TAG-LIST ::= tag ( INSTRUCTION-LIST ) TAG-LIST | nil
ATTRIBUTE-LIST ::= attribute , ATTRIBUTE-LIST | attribute
INSTRUCTION-LIST ::= INSTRUCTION , INSTRUCTION-LIST | INSTRUCTION
INSTRUCTION ::= FUNCTION | ASSIGNMENT
FUNCTION ::= function ( ARGUMENT-LIST ) | function ( )
ASSIGNMENT ::= ATTRIBUTE = RIGHT-ARGUMENT | ATTRIBUTE
ARGUMENT-LIST ::= ARGUMENT , ARGUMENT-LIST | ARGUMENT
ATTRIBUTE ::= entity . attribute
RIGHT-ARGUMENT ::= FUNCTION | ATTRIBUTE | entity | constant | tag
ARGUMENT ::= RIGHT-ARGUMENT | & ATTRIBUTE
```

10.2 BDT-Map

```
pmPA[pa_iperm]
hmHE[he_ipati,he_ityre,he_notes]

3000(pmPA.pa_iperm)
3101(pmPA.pa_lname)
3102(pmPA.pa_fname)
3103(pmPA.pa_birth=bdt_date_dhe())
3106(bdt_locality_dhe(&pmPA.pa_zipc2,&pmPA.pa_city2))
3107(bdt_street_dhe(&pmPA.pa_addr3,&pmPA.pa_numb2))
3110(pmPA.pa_sesso=bdt_sex_dhe())
3201()
3203(mxdata(3201,pmPA))
7700()
7701()
7709(hmHE.he_chval,
     hmHE.he_datva=bdt_date_dhe(7701),
     hmHE.he_ipati=pmPA,
     hmHE.he_ityre="dhe__0000008",
     hmHE.he_notes=7700,
     hmHE.he_stare="C")
7715(hmHE.he_chval,
     hmHE.he_datva=bdt_date_dhe(7701),
     hmHE.he_ipati=pmPA,
     hmHE.he_ityre="dhe__000000E",
     hmHE.he_notes=7700,
     hmHE.he_stare="C")
7725(hmHE.he_chval,
     hmHE.he_datva=bdt_date_dhe(7701),
     hmHE.he_ipati=pmPA,
     hmHE.he_ityre="dhe__000000K",
     hmHE.he_notes=7700,
     hmHE.he_stare="C")
7753(hmHE.he_chval,
     hmHE.he_datva=bdt_date_dhe(7701),
     hmHE.he_ipati=pmPA,
     hmHE.he_ityre="dhe__000000V",
     hmHE.he_notes=7700,
     hmHE.he_stare="C")
7757(hmHE.he_chval,
     hmHE.he_datva=bdt_date_dhe(7701),
     hmHE.he_ipati=pmPA,
     hmHE.he_ityre="dhe__000000Z",
     hmHE.he_notes=7700,
     hmHE.he_stare="C")
7760(hmHE.he_chval,
     hmHE.he_datva=bdt_date_dhe(7701),
     hmHE.he_ipati=pmPA,
     hmHE.he_ityre="dhe__000000c",
     hmHE.he_notes=7700,
     hmHE.he_stare="C")
```

10.3 Entity-Funktion

Der Generator erzeugt aus der BDT-Map 10.2 unter anderem folgende Entity-Funktion (Health datum *hmHE*). Ein Aufruf des DHE-API (Application Programming Interface) ist fett hervorgehoben.

```
void hmHE__W(void)
{
   hmHE__IL is_select;
   long i_offset = 0, o_nelem;
   hmHE__OL * osl_list;

   hmHE__L("*INIT_ONLY*", & is_select, & i_offset, & o_nelem, & osl_list, o_servmsg);
   is_select.he_icont[0] = '%';
   is_select.he_ityre[0] = '%';
   is_select.he_ipati[0] = '%';
   is_select.he__icode[0] = '%';
   is_select.he_clty[0] = '%';
   is_select.hi_iperm[0] = '%';
   is_select.hi_icont[0] = '%';
   is_select.hi_iacta[0] = '%';
   is_select.rd_iserp[0] = '%';
   is_select.ha_iact[0] = '%';
   is_select.oh_comm[0] = '%';
   is_select.hd_diag[0] = '%';
   is_select.dh_pdcode[0] = '%';
   is_select.by_bdpart[0] = '%';
   is_select.hh_elcode[0] = '%';
   is_select.hh_stcode[0] = '%';
   if (12 < strlen(hmHE.is_instance.he_ipati))
      strcat(strncpy(is_select.he_ipati, hmHE.is_instance.he_ipati, 11), "%");
   else
      strcpy(is_select.he_ipati, hmHE.is_instance.he_ipati);
   if (12 < strlen(hmHE.is_instance.he_ityre))
      strcat(strncpy(is_select.he_ityre, hmHE.is_instance.he_ityre, 11), "%");
   else
      strcpy(is_select.he_ityre, hmHE.is_instance.he_ityre);
   if (24 < strlen(hmHE.is_instance.he_notes))
      strcat(strncpy(is_select.he_notes, hmHE.is_instance.he_notes, 23), "%");
   else
      strcpy(is_select.he_notes, hmHE.is_instance.he_notes);
   osl_list = (hmHE__OL *) malloc(sizeof(hmHE__OL) * _l_maxlen);
   if (osl_list == NULL) error("\a\nERROR ad/hmHE__W: malloc failed\n");
   time(& time_dhe);
   chkerr(hmHE__L(i_env, & is_select, & i_offset, & o_nelem, & osl_list, o_servmsg));
   passed_dhe += difftime(time(NULL), time_dhe);
   switch(o_nelem) {
      case 0:
         i_action = 'A';
         time(& time_dhe);
         chkerr(hmHE__M(i_env, & i_action, & hmHE.is_instance, & hmHE.os_iden,
                                                                o_servmsg));
         passed_dhe += difftime(time(NULL), time_dhe);
         fprintf(log, "\n\thmHE\t %s", hmHE.os_iden._icode);
         break;
      case 1:
         strcpy(syscod, "hmHE"); /* pmPAdhe__0000001, hmHEdhe__0000001 */
         if (slexist(syscods, strcat(syscod, osl_list[0].he__icode)))
                                                                /* update once */
            strcpy(hmHE.os_iden._icode, osl_list[0].he__icode);
         else {
            strcpy(hmHE.is_instance.he__icode, osl_list[0].he__icode);
            memcpy(hmHE.is_instance.he__timestamp, osl_list[0].he__timestamp,
                                                                sizeof(TIMESTAMP));
            i_action = 'M';
            time(& time_dhe);
            chkerr(hmHE__M(i_env, & i_action, & hmHE.is_instance, & hmHE.os_iden,
                                                                o_servmsg));
            passed_dhe += difftime(time(NULL), time_dhe);
            fprintf(log, "\n\thmHE\t %s", hmHE.os_iden._icode);
            strcpy(syscod, "hmHE");
            strcat(syscod, hmHE.os_iden._icode); /* SLPUT is macro! */
            SLPUT(& syscods, syscod);
```

```
         }
         break;
      default:
         error("\a\nERROR ad/hmHE__W: identification failed\n");
   }
   free(osl_list);
   hmHE__M("*INIT_ONLY*", & i_action, & hmHE.is_instance, & hmHE.os_iden, o_servmsg);
   lfree(hmHE.attributes);
   linit(& hmHE.attributes);
}
```

10.4 Tag-Funktion

Der Generator erzeugt aus der BDT-Map 10.2 unter anderem folgende Tag-Funktion. In der Liste *hmHE.attributes* sind alle Attribute der Entität *hmHE* (Health datum) verzeichnet, die bereits belegt sind. Dadurch können Zuweisungskollisionen (Konflikte) erkannt werden. Die Konfliktbehandlung erfolgt durch Aufruf von Entity-Funktionen (fett).

```
void process7709(void)
{
   strcpy(value7709, value);
   if (slexist(hmHE.attributes, "he_chval")) hmHE__W();
   SLPUT(& hmHE.attributes, "he_chval");
   strcpy(hmHE.is_instance.he_chval, value7709);
   if (slexist(hmHE.attributes, "he_datva")) hmHE__W();
   SLPUT(& hmHE.attributes, "he_datva");
   strcpy(hmHE.is_instance.he_datva, bdt_date_dhe(value7709, value7701));
   if (slexist(hmHE.attributes, "he_ipati")) hmHE__W();
   SLPUT(& hmHE.attributes, "he_ipati");
   if (! lempty(pmPA.attributes)) pmPA__W();
   strcpy(hmHE.is_instance.he_ipati, pmPA.os_iden._icode);
   if (slexist(hmHE.attributes, "he_ityre")) hmHE__W();
   SLPUT(& hmHE.attributes, "he_ityre");
   strcpy(hmHE.is_instance.he_ityre, "dhe__0000008");
   if (slexist(hmHE.attributes, "he_notes")) hmHE__W();
   SLPUT(& hmHE.attributes, "he_notes");
   strcpy(hmHE.is_instance.he_notes, value7700);
   if (slexist(hmHE.attributes, "he_stare")) hmHE__W();
   SLPUT(& hmHE.attributes, "he_stare");
   hmHE.is_instance.he_stare = 'C';
}
```

10.5 Auszug aus HISA [CEN97]

8. Health Characteristic Healthcare Common Services (HC-HCS)

8.1 Scope

This HCS groups the services responsible for the management of the health characteristics on the subjects of care of interest for the healthcare centre.

The information managed by the HC-HCS shall include the description and classification of the various types of health characteristics as well as the actual data relating to the values of such characteristics for the individual subject of care. Health characteristics may be either elementary, for example relating to one single value, or structured for example comprising other elementary or, recursively, structured health characteristics.

The classification criteria defined through the HC-HCS shall conform and refer to taxonomies and coding criteria defined for the individual specialities, according to applicable standards and regulations at national and international level.

The properties for example structure, values, of the individual health characteristics shall be defined in individual installations through the concepts managed by these Healthcare Common Services, on the basis of international standards, national regulations and specific needs of the individual users.

Individual health characteristics may also represent result of the execution of activities. As defined in the specification of the Activity Healthcare Common Services, one health characteristic may represent the result of one individual activity, as well as the result of a combined group of activities for example multiple report.

At least three levels of status, Preliminary, Validated, Annulled, shall be possible for each health characteristic:

Preliminary

representing preliminary information, being unofficial, and whose validity is both limited over time and locally within the organisational unit which is collecting or generating them;

Validated

official information;

Annulled

information replaced through a later version.

8.2 Structural view

Conceptual diagram

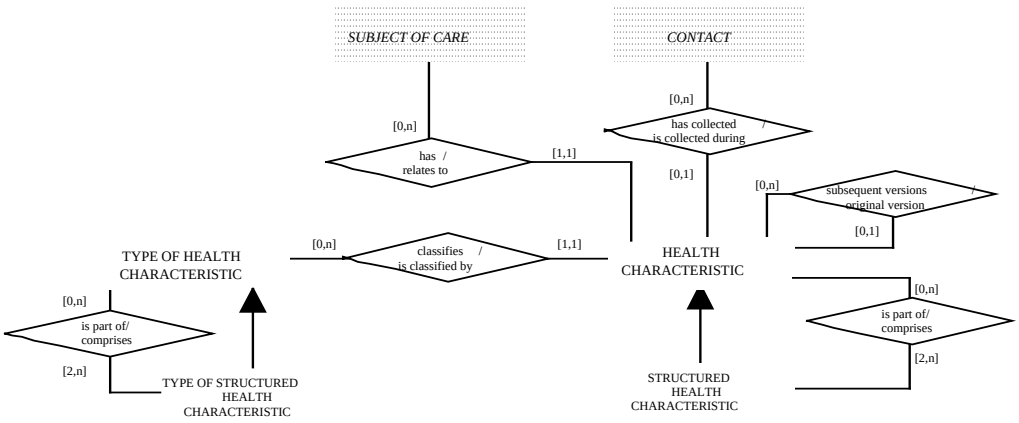

Figure 3:

Conceptual schema of the essential information managed by the Health Characteristic Healthcare Common Services

8.2.2 Specification of the individual entities

8.2.2.1 Entity: TYPE OF HEALTH CHARACTERISTIC

Definition

Concept used to classify the health characteristics which are managed by the healthcare organisation. Types of health characteristics may be either elementary information items or structured groups of other types of health characteristics.

Subsets	TYPE OF STRUCTURED HEALTH CHARACTERISTIC	Type of health characteristic consisting of a structured composition of other types of health characteristics

Essential attributes

Identifier of an overall class for types of health characteristics

Identifier of the type of health characteristic

Identifier whether the types of health characteristic is structured or not

Domain of the possible values which can be assumed

Unit of measure and coding criteria adopted in the registration of the actual values

Relationships with:

TYPE OF STRUCTURED HEALTH	**relationship:** is part of [0,n]

CHARACTERISTIC Defines the Types of Structured health characteristics in which this type (either being structured itself or not) may be comprised

attributes: rules (e.g. sequence position) specifying the role of the health characteristic in the structured type

8.2.2.2 Entity: HEALTH CHARACTERISTIC

Definition:

Element of information describing certain aspects of the health status of one SUBJECT OF CARE, that is considered relevant by the healthcare organisation and is registered in the Healthcare Information System

Subsets STRUCTURED HEALTH CHARACTERISTIC : Health characteristic consisting of a structured composition of other of health characteristics

Essential attributes

Identifier of the health characteristic
Status of the health characteristic i.e. Preliminary, Validated, Annulled
Date and time of the event through which the health characteristic has been generated or collected
Identification of the healthcare actor who has generated or collected the health characteristic
Date and time of the event through which the health characteristic has been finally validated
Identification of the healthcare actor who has finally validated the health characteristic
Version of the health characteristic (to permit the management of subsequent modifications)
Actual value of the health characteristic

Relationships with:

STRUCTURED HEALTH**relationship:** is part of [0,n]

CHARACTERISTIC Specifies the structured health data in which the health characteristic is comprised.

attributes: rules (e.g. sequence position) specifying the role of the health characteristic in the structured type

HEALTH CHARACTERISTIC **relationship:** original version [0,1]

Specifies the original health characteristic of which the instance is a subsequent version.

HEALTH CHARACTERISTIC **relationship:** subsequent versions [0,n]

Groups the sequence of subsequent versions of the health characteristic.

CONTACT **relationship:** is collected during [0,1]

Specifies the contact during which the health characteristic has been collected.

SUBJECT OF CARE **relationship:** relates to [1,1]
Specifies the subject of care to whom the health characteristic relates.

8.3 Functional view

For each entity defined in the structural view, the HC-HCS shall provide to the rest of the healthcare information system , at least, a set of services as defined in Clause 14.

14. Common functional characteristics of the Healthcare Common Services (HCS)

For each entity defined in the structural view of the various HCS, the healthcare middleware layer shall provide the rest of the healthcare information system with a set of services allowing, at least, the following functionalities:
retrieval of a list of instances of the entity, specified on the basis of different selection criteria defined through conditions on the attributes of the entity or on the relationships the entity has with other entities;

maintenance of one instance of the entity, i.e.
retrieval of the full set of information of one instance;
entering of a new instance;
modification of the attributes of one already existing instance;
deletion of one instance;

retrieval of the list of the instances related to one instance of the entity; for each relationship connected to the entity;

maintenance of one already existing instance for each relationship connected to the entity.

Tabellarischer Lebenslauf

Name	Ralf Schweiger
Geboren	am 19. November 1967 in Crailsheim/Württemberg
Nationalität	deutsch
Schulausbildung	Grundschule Stimpfach (1974-1978) Hauptschule Stimpfach (1978-1979) Realschule Crailsheim (1979-1985) Wirtschaftsgymnasium Crailsheim (1985-1988)
Staatsdienst	15 Monate Wehrdienst in Niederstetten/Württemberg (1988-1989)
Hauptstudium	Medizinische Informatik an der Universität Heidelberg (1989-1995), Thema der Diplomarbeit: Optimierung von Algorithmen zur Berechnung von Permutations-tests und Konfidenzintervallen
Nebenstudium	Mathematik an der Fernuniversität Hagen (seit 1992)
Wissenschaftliche Tätigkeit	Seit 1996 am Institut für Medizinische Informatik der Justus-Liebig-Universität Gießen. Mitarbeit an mehreren internationalen Projekten mit folgenden Tätigkeitsschwerpunkten: Integration klinischer Informationssysteme (Dissertation), Telemedizin, Anwendung von Internet-Technologien im medizinischen Bereich mit Schwerpunkt XML.

Danksagung

Mein Dank richtet sich vor allem an Prof. Dr. Joachim Dudeck für die Überlassung des Themas der Doktorarbeit und für das in mich gesetzte Vertrauen.

Weiterhin möchte ich mich bei meinen Arbeitskollegen, allen voran bei Dr. Thomas Bürkle, für die Unterstützung und die zahlreichen Diskussionen bedanken.