DIPLOMARBEIT

Konzeption, Entwurf, Implementierung und Validierung der OERR-Funktionalität in einer Standard KIS-Architektur mit Hilfe von DHE

Ruxandra Privighitorita

Diplomarbeit
am Fachbereich Informatik
der Universität Dortmund

21. Dezember 1997

Gutachter:

Prof. Dr. E.-E. Doberkat
Dipl.-Inform. H.-G. Sobottka

Teil I

Schriftliche Ausarbeitung

Zusammenfassung

Das im Rahmen des Projektes CEN/TC251 des „Comitee European de Normalisation“ (CEN) vorgeschlagene Basiskonzept eines Standards für KIS-Architekturen definiert eine Anzahl gesundheitsspezifischer Dienste, die in jeder KIS-Anwendung benutzt werden können, um die gegenseitige Zusammenarbeit der Anwendungen zu verwalten. Das von der Firma GESI in Rom entworfene und implementierte *Distributed Healthcare Environment* (DHE) bietet eine konkrete Implementierung dieser definierten Dienste an.

Die Aufgabe dieser Diplomarbeit ist die Entwicklung von charakteristischen Vorgänge eines *Anforderungssystems* (OERR - Order Entry and Result Reporting System) für Krankenhausstationen, mit Hilfe von geeigneten API-Komponenten, die von DHE zur Verfügung gestellt werden. Ein weiteres Ziel dieser Arbeit ist die Validierung der DHE-Software durch das implementierte Anforderungssystem. Das Ergebnis der Validierung soll zeigen, inwieweit die DHE-Software die von CEN vorgeschlagene Standardarchitektur implementiert, um damit moderne und offene Krankenhausinformationssysteme zu entwickeln, anzupassen, zu integrieren und erfolgreich und effizient zu betreiben. Diese Validierung wird anhand von festgelegten Erfolgsfaktoren durchgeführt.

Stichwörter

Distributed Healthcare Environment (DHE), Middleware, Anforderungssystem (OERR), Object Oriented Software Engineering (OOSE), Qualitätssicherung, Qualitätsmodell, Validierungskonzept.

Abstract

As a part of the CEN/TC251 project of the „Comitee European de Normalisation“ (CEN), the proposed basic concept of a standard for HIS-architectures defines a set of healthcare-specific services common to all healthcare information systems, useable by any application, to manage mutual interworking. The *Distributed Healthcare Environment* (DHE), which was designed and implemented by GESI company, Rome, offers concrete implementation of this defined services.
One of the tasks of this master thesis is the development of characteristic use cases of an *Order Entry and Result Reporting System* (OERR System) for hospital wards, using the API-components of DHE. A further aim of this thesis is the validation of the DHE-software, based on the implemented OERR-system, as a basis for an open standard for HIS-architectures. The validation will be performed based on a set of success quality factors.

Keywords

Distributed Healthcare Environment (DHE), Middleware, Order Entry and Result Reporting System (OERR), Object Oriented Software Engineering (OOSE), Quality Security, Quality Model, Validation Concept.

Inhaltsverzeichnis

Teil I Schriftliche Ausarbeitung

Teil II Entwicklungsdokument

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Krankenhäuser sind mehr als die Summe ihrer Kliniken, Institute und Fachabteilungen und Krankenhausinformationssysteme (KIS) sind mehr als die Summe der Abteilungsinformationssyteme. Gesundheitsstrukturgesetz (GSG), Kostendruck und die zunehmende Verzahnung von ambulanter und stationärer Betreuung bewirken einen Ausbau von Arbeitsteiligkeit, Spezialisierung und Dezentralisierung, die eine optimale Kommunikation und Kooperation zwischen den Krankenhausbereichen erfordern. Das gilt für die realen Versorgungsprozesse ebenso wie für die sie unterstützenden Informationssysteme. Damit kommt zukünftig der Integration dezentralisierter, heterogener, offener Anwendungssysteme im Krankenhaus eine entscheidende Bedeutung zu. KIS-Anwendungen erstrecken sich von der üblichen Bearbeitung der Patientenakten bis hin zum Behandlungsmanagement, Rechnungswesen und zur Erbringung von Leistungen außerhalb der Stationen.

Die Infrastruktur heute eingesetzter KIS ist durch den Einsatz einer Vielzahl unterschiedlicher Hard- und Software-Komponenten geprägt. Sie basieren auf veralterte EDV-Technologien und werden den modernen Anforderungen im System Krankenhaus nicht mehr gerecht.

Das im Rahmen des Projektes CEN/TC251 des „Comitee European de Normalisation" (CEN) vorgeschlagene Basiskonzept eines Standards für KIS-Architekturen, welches insbesondere die Integration von bestehenden Anwendungen unterstützt und den gemeinsamen Betrieb heterogener Anwendungen von verschiedenen Herstellern erlaubt, definiert eine Menge gesundheitswesenspezifischer Dienste, die in jeder KIS-Anwendung benutzt werden kann, um die gegenseitige Zusammenarbeit der Anwendungen zu unterstützen.

Das von der Firma GESI in Rom entworfene und implementierte Distributed Healthcare Environment (**DHE**) bietet eine konkrete Implementierung dieser definierten Dienste an. DHE wurde konzipiert und implementiert, um den Grundstein für einen offenen Standard für KIS-Architekturen in Europa zu legen. DHE soll u.a. die Entwicklung von neuen Anwendungen auf der Basis schon bestehender Systeme erlauben und die verschiedenen Anforderungen der Stationen berücksichtigen können. Dies soll zur Kostenreduktion für neue Systeme und zur Sicherung bereits getätigter Investitionen in bestehende Systeme beitragen.

Das von CEN vorgeschlagene Architekturgerüst eines KIS ist in drei Hauptebenen strukturiert (vgl. Abbildung 1). Auf der obersten Ebene werden Anwendungen spezifisch entworfen und an die Bedürfnisse jeder Station und Abteilung des Krankenhauses speziell angepaßt (*Anwendungsebene*). Die darunterliegende *DHE-Ebene* verwaltet die Dienste, die von den verschiedenen Anwendungen genutzt werden können. Die dritte und unterste Ebene (*Technologische Plattform*) ist für die transparente Integration der verteilten, heterogenen Umgebung, die im Krankenhaus vorhanden ist, zuständig. Diese Ebene stellt die Protokolle für die Kommunikation zwischen den vorhandenen Komponenten zur Verfügung.

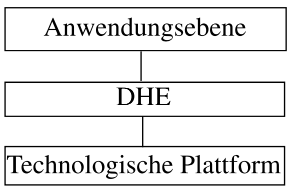

Abbildung 1: CEN-Architekturgerüst

1.1 Aufgabenstellung

Das Thema dieser Diplomarbeit hat sich durch externe Kontakte der Projektgruppe „PROMETHEUS - Spezifikation eines Systems zum Behandlungsmanagement im Krankenhaus" ([BCF+97]) zur ROKD GmbH in Bielefeld ergeben. Die Arbeit wurde deshalb in enger Zusammenarbeit mit dieser Firma durchgeführt.

Die ROKD GmbH (Rechenzentrum Ostwestfalen für Kirche und Diakonie) ist ein Anbieter von Lösungen im Krankenhausinformationsbereich. Es tritt je nach Kundenwunsch und Bedarf als Berater, Soft- und Hardwarelieferant oder auch als Rechenzentrum auf. Durch die Teilnahme des Unternehmens an dem EU-Projekt HANSA (Healthcare Advanced Networked System Architecture) (vgl. [Lor96]) ergab sich als Teilaufgabe die Implementierung von Teilen eines Anforderungssystems (**OERR** - Order Entry and Result Reporting) für Krankenhausstationen, mit Hilfe von geeigneten Application Programming

Interface (API) - Komponenten, die von DHE zur Verfügung gestellt werden. Dabei kann das Konzept der OERR-Komponenten des bestehenden Krankenhausinformationssystem proCOM der ROKD GmbH benutzt werden (vgl. [ROKD95]).

Das *Order Entry* - System als Teil eines Anforderungssystems unterstützt die Verwaltung und Durchführung von Anforderungen (wie z. B. Labor- und Röntgenanforderungen), die auf einer Krankenhausstation erforderlich sind und leitet diese an die verschiedenen Funktionsbereiche des Krankenhauses weiter (z. B. zum Labor oder zu der Röntgenabteilung). Nach der Durchführung der Anforderungen übermittelt der *Result Reporting* - Teil des Anforderungssystems die Anforderungsergebnisse von den Funktionsbereichen zu den Stationen. Die OERR-Anwendung entsteht demnach auf der Anwendungsebene des in Abbildung 1 dargestellten Architekturgerüstes.

Weiterhin wird ein Konzept aufgestellt, das als Ziel die Validierung der DHE-Software auf der Basis des implementierten Anforderungssystems hat. Diese Validierung wird anhand von in der Konzeptionsphase der Diplomarbeit festgelegten Erfolgsfaktoren durchgeführt. Sie soll zeigen, inwieweit die DHE-Software die von CEN vorgeschlagene Standardarchitektur implementiert, um damit moderne und offene Krankenhausinformationssysteme zu entwickeln, anzupassen, zu integrieren und erfolgreich und effizient zu betreiben.

DHE ist für die Entwicklung von Krankenhausinformationssytemen konzipiert. Ein KIS besteht aus verschiedenen Anwendungen, die die Bedürfnisse jeder Station und Abteilung eines Krankenhauses erfüllen. Diese Anwendungen erstrecken sich von der rechnergestützten Dokumentation von Pflegemaßnahmen und Diagnosen auf der Station bis hin zu der Generierung und Ergebnisdokumentation von Leistungen, sowie Rechnungswesen und Verwaltung von Patientenstammdaten. Das OERR-System unterstützt die Generierung und Ergebnisdokumentation von Leistungen auf der Station und in den Funktionsbereichen und stellt somit eine repräsentative Komponente eines KIS dar. Um festlegen zu können, inwieweit die Anforderungen von KIS-Anwendungen mit DHE erfüllt werden können, wird das OERR-System repräsentativ gestaltet: es werden charakteristische Vorgänge implementiert, deren Eigenschaften und Merkmale auf das gesamte KIS übertragbar sind. Die Auswahl dieser Vorgänge erfolgt anhand von vorgegebenen Erfolgsfaktoren. Es wird davon ausgegangen, daß wenn diese Vorgänge korrekt funktionieren, DHE für die Entwicklung eines gesamten KIS angewendet und somit Teil einer Standard-Architektur werden kann.

1.2 Inhalt

Diese Diplomarbeit besteht aus zwei Teilen. Der erste Teil („Schriftliche Ausarbeitung“) gliedert sich wie folgt: In den Kapiteln 2 und 3 werden zunächst wesentliche Grundlagen des Software-Entwicklungsprozesses im Bereich der Krankenhausinformationssysteme vermittelt, die zum Verständnis der Thematik und der damit verbundenen Problematik benötigt werden. Kapitel 4 enthält das Validierungskonzept, in dem spezielle kritische Erfolgsfaktoren, wie beispielsweise Performanz und Stabilität, und für diese Faktoren entsprechende Auswertungsmethoden festgelegt werden. Weiterhin werden in diesem Kapitel die charakteristischen OERR-Vorgänge anhand der festgelegten Erfolgsfaktoren identifiziert. In den Kapiteln 5 bis 7 wird das Validierungskonzept für jede der Software-Entwicklungsphasen des OERR-Systems durchgeführt. Kapitel 8 beschreibt die Auswertung und Interpretation der Ergebnisse der Validierung. Im Kapitel 9 werden Migrationsstrategien und Einsatzmöglichkeiten der DHE-Software vorgestellt. Eine Zusammenfassung dieser Arbeit und ein Ausblick folgen im Kapitel 10.

Der zweite Teil dieser Diplomarbeit ist das „Entwicklungsdokument“ und befindet sich in der Lehrstuhl-Bibliothek des Lehrstuhls 10 (Software-Technologie) der Universität Dortmund. Die Kapitel in diesem Dokument sind beginnend mit dem Buchstaben „E“ bezeichnet. Damit soll sichergestellt werden, daß die Verweise auf dieses Dokument nicht mit denen aus der „Schriftlichen Ausarbeitung“ verwechselt werden. So wird beispielsweise auf Kapitel 1 im Entwicklungsdokument in folgender weise verwiesen: vgl. Kapitel E1.

Das Entwicklungsdokument gliedert sich wie folgt: Kapitel E1 enthält die Sammlung von Auswertungsmethoden für die im Kapitel 4 identifizierten Validierungsziele. Im Kapitel E2 werden die Ergebnisse der Analysephase des OERR-Systems beschrieben. Kapitel E3 enthält die Ergebnisse der Konstruktionsphase und Kapitel E4 die der Testphase. Im Kapitel E5 werden Hinweise für die Installation und Start der OERR-Anwendung beschrieben.

2 Software-technologische Grundlagen

Dieses Kapitel dient der grundlegenden Beschreibung wesentlicher Merkmale der Software-Entwicklung. Die Begriffe in diesem Kapitel sind nicht vollständig beschrieben, sondern dienen nur als Grundlage für das bessere Verständnis der nachfolgenden Kapitel. Es werden klassische Ansätze erläutert, wie beispielsweise Phasenmodelle, da sie eine Reihe wichtiger Grundideen für die „maßgerechte“ Software-Entwicklung mit neuen Methoden und Programmiersprachen enthalten. Für ein spezielleres Interesse und ausführlichere Informationen zu den in diesem Kapitel angesprochenen Themengebiete wird an den entsprechenden Stellen auf die Literatur verwiesen.

Der Begriff Software-Engineering umfaßte in den letzten 25 Jahren u.a. die Entwicklung und Bereitstellung von Methoden, um Software zu entwickeln und zu warten. Bedingt durch den Einsatz immer komplexerer Software-Systeme in Bereichen, wo eine sehr hohe Zuverlässigkeit der Software gefordert wird, ist es heutzutage mehr und mehr das Ziel, qualitativ hochwertige, zuverlässige und wartbare Software zu erstellen (vgl. [Dum92]). Um dieses Ziel erreichen zu können, ist es sinnvoll, bei der Entwicklung eines Software-Systems sowohl konstruktive (entwickelnde) als auch analytische (prüfende) Elemente einzusetzen.

In dem Kapitel 2.1 wird das Phasenmodell als Grundlage für die Entwicklung von Software-Systemen, mit seinen Vor- und Nachteilen, sowie die Rolle der Software-Qualitätssicherung im Rahmen des Software-Engineerings vorgestellt. Im Mittelpunkt stehen dabei verschiedene Kategorien von Qualitätssicherungsmaßnahmen, wobei die konstruktive und analytische Qualitätssicherung eine bedeutende Rolle spielt. Kapitel 2.2 vertieft den Bereich der konstruktiven Qualitätssicherung. Ausgangspunkte sind Methoden, Formalismen und Werkzeuge. Diese konstruktiven Elemente werden durch ein für die Qualitätssicherung passendes Vorgehensmodell integriert und im Entwicklungsprozeß des OERR-Systems in den Kapiteln 5 bis 7 angewendet. Kapitel 2.3 beschreibt die Maßnahmen der analytischen Qualitätssicherung. Im Mittelpunkt stehen Qualitätsmodelle als umfassendes Hilfsmittel zur Qualitätsplanung und zur Qualitätsbewertung, sowie statische und dynamische Prüfungen. Die dort vorgestellten Prüfmethoden werden in dieser Arbeit als Grundlage für das Aufstellen des Validierungskonzeptes eingesetzt und stellen Werkzeuge für die qualitative Bewertung der DHE-Software dar.

2.1 Der Life-Cycle in der Software-Entwicklung

Das Konzept des Software-Lebenszyklus (*Life-Cycle*) ist der zentrale Gegenstand aller Software-Entwicklungsmethoden. Die Definition eines Software-Lebenszyklus soll helfen, den Software-Entwicklungsprozeß von der Spezifikation, über den Entwurf, der Kodierung bis hin zur Wartung von Software besser verstehen und kontrollieren zu können.

Um die Entwicklung und Pflege von Software-Systemen in geordneten Stufen abzuwickeln, wurden Phasenmodelle entwickelt. Eines der ersten Phasenmodelle mit Rückkopplungen zwischen den Phasen ist das in Abbildung 2 zu betrachtende Life-Cycle-Modell. Die einzelnen Phasen sind u.a. in [DF89] beschrieben und laufen prinzipiell sequentiell ab. Von jeder dieser Phasen besteht jedoch die Möglichkeit an den bezeichneten Stellen zurückzuspringen, um Änderungen oder Verbesserungen vorzunehmen (vgl. Abbildung 2).

Die Probleme dieses Life-Cycles sind seit längerem bekannt: bei Iterationen in diesem Modell kommt es oft zu erheblichen Qualitätsmängeln und je länger es dauert, bis der Auftraggeber bzw. die Benutzer ein ablauffähiges Modell der Applikation zur ersten Prüfung erhalten, um so größer ist das Risiko, die Anforderungen zu verfehlen.

Um eine höhere Qualität der Software-Systeme zu erzielen, ist im Rahmen des Software-Engineerings die Teildisziplin *Software-Qualitätssicherung* entstanden. Diese Teildisziplin versucht, „den Faktor Qualität durch eigene Hilfsmittel und durch geplante und systematische Anwendung des Software Engineering im Life-Cycle zu berücksichtigen” [Wal90].

Laut DIN 55350, Teil 11 ist *Qualität* „die Gesamtheit von Eigenschaften und Merkmalen eines Produktes oder einer Tätigkeit, die sich auf deren Eignung zur Erfüllung gegebener Erfordernisse bezieht”. *Eigenschaften*, die die Qualität konstituieren, sind jene, „die für den Produktbenutzer oder den Dienstleistungsnehmer relevant sind” ([Wal90]). Ein *Merkmal* ist nach DIN 55350, Teil 12, jene Eigenschaft, „die eine quantitative oder qualitative Unterscheidung eines Produktes oder einer Tätigkeit aus einer Gesamtheit ermöglicht”.

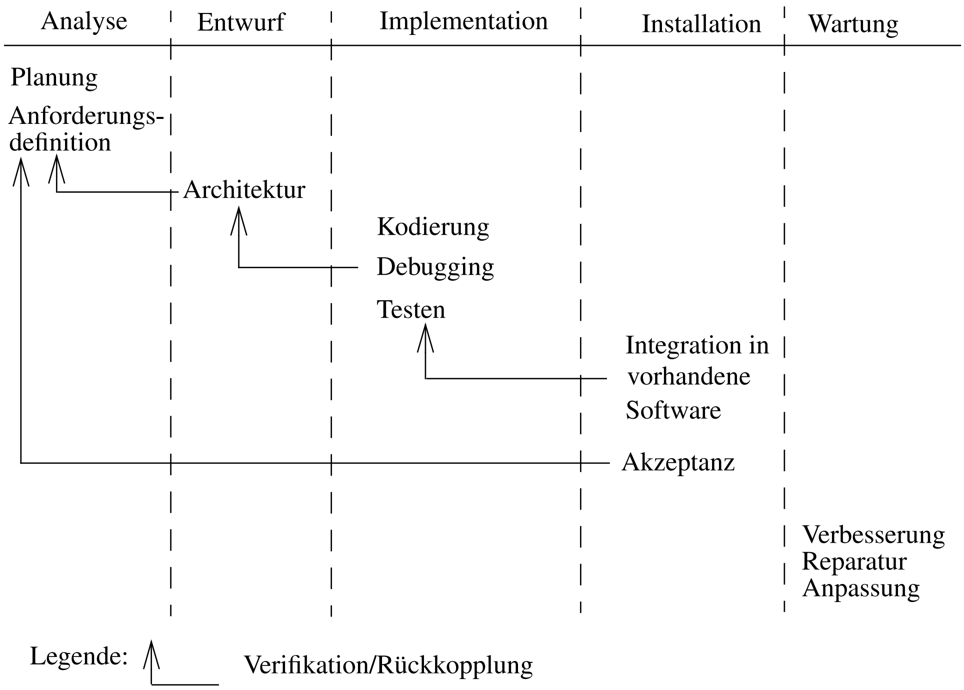

Abbildung 2: Das Life-Cycle-Modell eines Software-Systems ([DF89])

Damit nicht nur die Bewertung von Software-Produkten durch analytische Qualitätssicherungsmaßnahmen durchgeführt wird, sondern Qualität auch durch konstruktive Qualitätssicherungsmaßnahmen realisiert werden kann, kann ein Bewertungsansatz für den Entwicklungs- und Pflegeprozeß aufgestellt werden. Das bedeutet, daß Merkmale und Bewertungsmaßstäbe auch für Zwischen-/Endprodukte und für deren Entwicklungsaktivitäten in allen Phasen des Prozesses benötigt werden.

Qualität ist nichts Absolutes, sondern sollte immer relativ zu gegebenen Erfordernissen gesehen werden. Eine Qualitätsbewertung beinhaltet demnach immer einen Vergleich zwischen den aus den gegebenen Erfordernissen abgeleiteten Qualitätsvorgaben (Soll-Werte) und den tatsächlich erreichten Ausprägungen der Merkmale (Ist-Werte) (vgl. [Wal90]).

Die konstruktiven und analytischen Elemente des Software-Engineerings zur Qualitätssicherung werden in den folgenden Kapiteln 2.2 und 2.3 beschrieben.

2.2 Konstruktive Elemente des Software-Engineerings zur Qualitätssicherung

„Unter konstruktiven Qualitätssicherungsmaßnahmen werden all jene, die zur Qualitätsgestaltung dienen, verstanden” [Wal90]. Sie sind präventiv und sollen das Entstehen von Fehlern und Qualitätsmängeln von vornherein durch Vorgabe von geeigneten Methoden, Formalismen und Werkzeugen verhindern. Sie schließen auch alle Maßnahmen zur Fehlerbehebung ein.

Eine *Methode* wird in Anlehnung an [Wal90] als planmäßig angewandte Vorgehensanweisung zur Erreichung von festgelegten Zielen (z.B. bessere Qualität und standardisierte Ergebnisse) definiert. *Formalismen*, insbesondere Sprachen, werden auf den verschiedenen Abstraktionsebenen zur Ergebnisbeschreibung verwendet und ermöglichen die Darstellung von Zwischen- und Endergebnissen des methodischen Arbeitens. *Werkzeuge* unterstützen die Anwendung von Methoden und Formalismen. Das verbindende Element der drei beschriebenen Begriffe ist ein standardisiertes Vorgehen, das über Aktivitäts- und Ergebnistypen alle anderen Elemente integriert. Die Beschreibung des standardisierten Vorgehens wird als *Vorgehensmodell* bezeichnet (vgl. [Wal90]).

Im folgenden werden die aufgeführten konstruktiven Elemente näher betrachtet.

2.2.1 Das Vorgehensmodell

Um die Qualität der Ergebnisse einer jeden Entwicklungsphase durch eine zeitliche und inhaltliche Strukturierung des Entwicklungsprozesses sicherzustellen, wird das im Kapitel 2.1 beschriebene Phasenmodell verfeinert und zu einem Vorgehensmodell erweitert. „Ein Vorgehensmodell beschreibt modellhaft, d.h. idealisiert und abstrahierend, den Software-Entwicklungs- und -Pflegeprozeß" [Wal90]. Durch Vorgehensmodelle wird der Software-Entwicklungsprozeß in aufeinander abgestimmte Phasen zerlegt, und für jede Phase werden Tätigkeiten und Ergebnisse festgelegt. Die Tätigkeiten einer Phase werden unter Zuhilfenahme von konstruktiven Qualitätssicherungselementen, wie beispielsweise den Methoden, Formalismen und Werkzeugen, durchgeführt. Die Durchführung einer Tätigkeit setzt gewisse Informationen voraus (zum Beispiel Ergebnisse von anderen Tätigkeiten) und erzeugt Ergebnisse.

Besondere Bedeutung für die Qualitätssicherung haben *V-Modelle* erlangt. In dem V-Modell nach Boehm ([Boe79]) (Abbildung 3) sind nicht nur die Phasen des Life-Cycles (vgl. Kapitel 2.1) sehr deutlich herausgearbeitet. Durch die Gegenüberstellung korrespondierender Phasen wird auch deutlich, wie konstruktive (z.B. die Phase Komponentenentwurf) und analysierende Tätigkeiten (z.B. der Integrationstest) zusammenhängen.

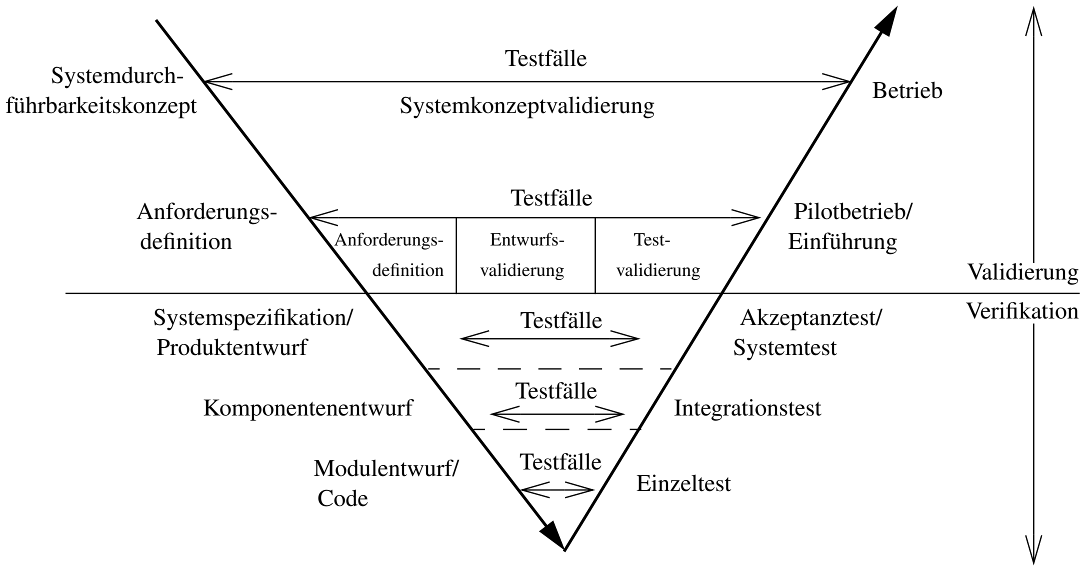

Abbildung 3: Das V-Modell

Die so entstandene Phasenzuordnung kann bildhaft als V dargestellt werden, wobei die linke Achse konstruktive Aktivitäten und die rechte Achse prüfende Aktivitäten enthält. Die prüfenden Tätigkeiten sind Validierungs- und Verifikationsaktivitäten und werden im Kapitel 2.3 erläutert.

Die Symmetrie dieses Modells kann folgendermaßen interpretiert werden: Fehler können am einfachsten auf derjenigen Abstraktionsstufe gefunden werden, auf der sie entstanden sind. Da bei einer Systementwicklung vom Ganzen (Aufgabenstellung) zum Detail (Code) fortgeschritten wird (dies entspricht der linken Achse im V-Modell) und anschließend vom Detail über Integrations- und Testschritte zum Ganzen (dies entspricht der rechten Achse im V-Modell), ist es sinnvoll, in den Phasen auf der rechten Seite des V nur jene Fehler zu suchen, die in der symmetrischen Phase auf der linken Achse des V entstanden sind. Konkret bedeutet dies, daß beispielsweise Testfälle in der Phase Modulentwurf aufgestellt werden und in der Phase Einzeltest zur Fehlerfindung ausgeführt werden. Je länger die Verweilzeit eines Fehlers im System ist, desto teurer ist seine Behebung. Durch entsprechende Prüfmaßnahmen in jeder konstruktiven Phase kann dieser Zeitabschnitt verkürzt werden.

Die Anwendung des V-Modells zeigt, wie wichtig ein ausbalanciertes Vorgehensmodell für die Wirksamkeit von Qualitätssicherungsmaßnahmen sein kann. Das V-Modell wird in dieser Arbeit bei der Entwicklung des OERR-Systems eingesetzt.

2.2.2 Die Methode

Methoden sind ein weiteres konstruktives Element für die Sicherstellung der Produktqualität. Die Anwendung von Methoden führt zu Arbeitsergebnissen, von denen angenommen wird, daß sie einer bestimmten Qualität bzw. einem Standard entsprechen. Ein Zweck des Einsatzes von Methoden aus der Sicht der Qualitätssicherung besteht in der systematischen und daher auch nachvollziehbaren Ergebnisermittlung und in der Aufstellung projektbegleitender Dokumentation.

Als Anforderungen an eine Methode aus der Sicht der Qualitätssicherung gelten eine Vereinheitlichung der Dokumentation der Arbeitsergebnisse, eine systematische und zielstrebige Vorgehensweise sowie eine begrenzte Anzahl planbarer Schritte, die zum Ergebnis der Aufgabe führen. Beispiele für Methoden die teilweise oder ganz diese Anforderungen erfüllen, sind: der Methodenrahmen STEPS (Software Technology for Evolutionary and Participative System Development) mit dem objekt-orientierten WAM (Werkzeug-Aspekt-Material)-Ansatz (vgl. [BGZ92]), MSA (Moderne Strukturierte Analyse) (vgl. [You89]), OOSE (Object-Oriented Software Engineering) (vgl. [JCJ+93]) etc.

Ähnliche Anforderungen an eine Methode wurden auch von der Projektgruppe PROMETHEUS in [BCF+97] aufgestellt. Dort werden die erwähnten Methoden einzeln analysiert. Diese Untersuchungen ergeben, daß für die erwähnten Anforderungen die OOSE-Methode am besten geeignet ist. Diese wurde auch im Rahmen der Projektgruppe erfolgreich eingesetzt und auch in dieser Arbeit bei der Entwicklung des OERR-Systems angewendet.

Grundsätzlich besteht OOSE aus drei Teilprozessen: *Analyse*, *Konstruktion* und *Test* (vgl. Abbildung 4).

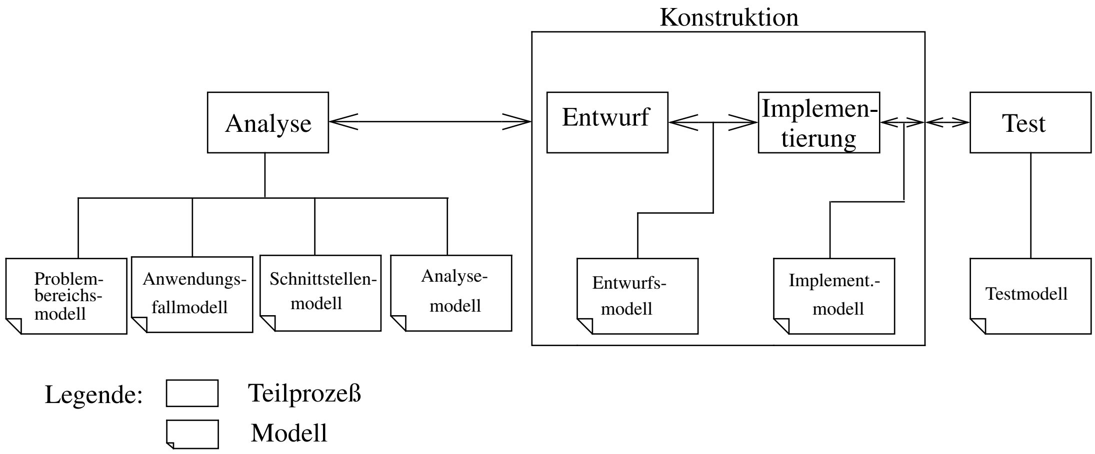

Abbildung 4: Teilprozesse und Modelle der OOSE-Methode (vgl. [JCJ+93])

Die daraus resultierenden Modelle werden im folgenden kurz beschrieben, um deren Anwendung in den nachfolgenden Kapiteln deutlich und verständlich zu machen. Für Details zu diesen Modellen wird jedoch an dieser Stelle auch auf [JCJ+93] verwiesen.

Das Problembereichsmodell

An der Analyse ist der zukünftige Anwender des zu entwickelnden Systems unmittelbar beteiligt und liefert die Grundlage für die Anwendungsfälle, sowie Erläuterungen fachspezifischer Begriffe. Letztere werden in einem Glossar, dem Problembereichsmodell (*domain object model*) gesammelt. Die fachspezifischen Begriffe werden in OOSE *Problembereichsobjekte* genannt. Die Problembereichsobjekte sind Objekte, die ein direktes Gegenstück in der Anwendungsumgebung besitzen. Durch diese gemeinsame Terminologie wird die Kommunikationsbasis über das System zwischen dem Anwender und dem Entwickler geschaffen.

Das Anwendungsfallmodell

Im Anwendungsfallmodell (*use case model*) werden die Anforderungen an das zu entwickelnde System in Form von *Anwendungsfällen* festgelegt. Anwendungsfälle beschreiben, wie ausgewählte Arbeitssituationen durch das zu entwickelnde System ausgeführt werden sollen. Die geforderte Funktionalität des Systems ergibt sich aus der Menge aller Anwendungsfälle. Zusätzlich wird im Anwendungsfall festgelegt, welche Akteure (z.B. Anwender, externe Systeme etc.) Aktionen des Systems veranlassen können.

Die Schnittstellenbeschreibung

Die Schnittstellenbeschreibung (*interface description*) spezifiziert wichtige Schnittstellen des Systems nach außen. Hierzu gehören Bildschirmmasken aber auch Protokolle zu anderen Systemen.

Das Analysemodell

Ziel bei der Konstruktion des Analysemodells ist der Entwurf einer änderungsfreundlichen und einfach erweiterbaren Systemstruktur (vgl. [JCJ+93], S.169). Dies soll dadurch erreicht werden, daß Änderungen oder Erweiterungen des Systems durch lokale Änderungen oder Erweiterungen einzelner Objekte vorgenommen werden können. Bei der Aufstellung des Analysemodells wird von idealen Umgebungsbedingungen ausgegangen. Das bedeutet, daß spezielle Erfordernisse und Einschränkungen, die sich durch die verwendete Software und Hardware ergeben, noch nicht beachtet werden. Im Analysemodell werden alle für das System benötigten Objekte und deren Beziehungen zueinander beschrieben. Im Gegensatz zum Problembereichsmodell werden auch Objekte verwendet, denen keine direkte Entsprechung aus dem Anwendungsbereich zugeordnet werden kann.

Im Analysemodell stehen drei grundsätzliche Objekttypen mit entsprechenden Attributen und Assoziationen zur Verfügung:

Schnittstellenobjekte realisieren die unmittelbare Interaktion mit den Akteuren und sind kurzlebig, d.h. sie existieren nur solange, wie der Anwendungsfall durchgeführt wird. Sie finden ihre Entsprechung insbesondere in der graphischen Benutzungsoberfläche.

Entitätsobjekte enthalten die Komponenten des Zustandes des Systems, die einzelne Anwendungsfälle überdauern oder auf die von mehr als einem Akteur zugegriffen wird. Sie dienen der Speicherung von Daten und werden weitgehend aus dem Problembereichsmodell übernommen.

Kontrollobjekte beschreiben verhältnismäßig kurzlebige, auf einen einzelnen oder wenige verbundene Anwendungsfälle beschränkte Komponenten des Systems, die hauptsächlich der Ablaufsteuerung dienen.

Neu gewonnene Kenntnisse bei der Erstellung des Analysemodells können eine Überarbeitung des Anforderungsmodells erfordern.

Das Entwurfsmodell

Das Entwurfsmodell (*design model*) ist das Resultat der Entwurfsphase als Teil der Konstruktion (vgl. Abbildung 4). Das zuvor aufgestellte Analysemodell wird hier im Hinblick auf technische Randbedingungen modifiziert (vgl. [JCJ+93], S.199). Das Entwurfsmodell beschreibt somit das zu entwickelnde System unter Berücksichtigung der verwendeten Hard- und Software. Die Strukturen des Analysemodells sollten allerdings, wegen der erzielten Robustheit, soweit wie möglich beibehalten werden.

Die in diesem Modell berücksichtigte Systemumgebung beinhaltet die Zielumgebung, die Programmiersprache sowie eventuell vorhandene Produkte, wie beispielsweise ein Datenbankmanagementsystem (DBMS), Netzwerke etc. Um die Reihenfolge der Ereignisse aus einem Anwendungsfall darzustellen, werden darüberhinaus im Entwurfsmodell noch *Interaktionsdiagrammen* aufgestellt (vgl. [JCJ+93]).

Analog zum Analysemodell, können auch nach der Entwicklung des Entwurfsmodells Änderungen im Anforderungsmodell oder im Analysemodell erforderlich sein.

Das Implementierungsmodell

Das Implementierungsmodell (*implementation model*) ist das Resultat der Implementierungsphase als zweitem Teil der Konstruktion (vgl. Abbildung 4). Es umfaßt die Beschreibung des erstellten Entwurfsmodells mit den Mitteln einer konkreten Programmiersprache, des zugrundeliegenden Betriebssystems etc.

Bei der Implementierung können Fehler im Entwurfsmodell deutlich werden. In diesem Fall müssen das Entwurfsmodell und eventuell auch vorhergehende Modelle verbessert werden.

Das Testmodell

Zum Test der Implementierung wird das Testmodell (*testing model*) aufgestellt. Dieses Modell besteht aus der Spezifikation der durchzuführenden Tests und aus den Testergebnissen. Abhängig vom Ausfall der Tests kann es nötig sein, früher erstellte Modelle zu überarbeiten.

Die Nachvollziehbarkeit (*traceability*) zwischen den verschiedenen Modellen ist eine der wichtigsten Eigenschaften der OOSE-Methode. Wenn Änderungen in der Anforderungsspezifikation vorzunehmen sind, so ist durch OOSE nachvollziehbar, an welcher Stelle gegebenfalls der Quellcode zu ändern ist. So können Konsistenzfehler im Entwicklungsprozeß leicht vermieden werden.

OOSE stellt also einen durchgängigen Prozeß mit Berücksichtigung der Qualitätssicherung dar (vgl. [JCJ+93], S.200). Dadurch unterscheidet es sich positiv von den meisten anderen z.B. in [BCF+97] betrachteten Methoden.

2.2.3 Die Formalismen

Sehr häufig sind Methoden direkt oder indirekt mit Formalismen oder formalen Sprachen verbunden. Ein Formalismus ist „eine spezielle textuelle oder graphische Notation. Er ist mehr oder minder formal durch ein System von Regeln festgelegt, die die Syntax und Semantik beschreiben“ [Wal90]. Der konstruktive Qualitätssicherungsaspekt von Formalismen und Sprachen besteht in der Unterstützung einer formalisierten Beschreibung der Ergebnisse, die Methoden liefern.

Die oben beschriebene OOSE-Methode empfiehlt für den Aufbau der Analyse- und Entwurfsmodelle als Formalismus verschiedene textuelle und graphische Notationen. Da die OOSE-Methode keinen Formalismus zur Konstruktion von Oberflächenprototypen zur Verfügung stellt, sollte für die Spezifikation und Dokumentation der Dialogabläufe des entwickelten Oberflächenprototyps ein weiterer Formalismus eingesetzt werden. Hier bieten sich zum Beispiel *Dialognetze* an.

Dialognetze wurden zur Beschreibung von Dialogabläufen in graphisch-interaktiven Systemen entwikkelt. Sie sind für die Beschreibung paralleler Teildialoge, wie sie in graphischen Benutzungsschnittstellen auftreten, besonders geeignet (vgl. [Jan93]).

Dialognetze basieren auf Petri-Netzen, die bereits im Grundsatz klar definierte Konzepte zur Beschreibung von Parallelität mitbringen. Dialognetze bestehen aus Stellen (Kreise), Transitionen (Rechtecke) und einer Flußrelation (Pfeile) zwischen Stellen und Transitionen oder umgekehrt.

Dialognetze stellen im Rahmen einer aufgaben- und benutzerorientierten Vorgehensweise schwerpunktmäßig ein Darstellungsmittel für Software-Entwickler dar. Dialognetze führen zu besser durchdachten Dialogen und sind ein Hilfsmittel für „eine bessere Integration der Entwicklung der Benutzungsschnittstelle mit software-technischen Methoden“ (vgl. [Jan93]).

2.2.4 Werkzeuge und Programmiersprache

Werkzeuge helfen die Qualität zu verbessern, indem sie einerseits das Entstehen von Fehlern vermeiden und andererseits die Anwendung der Methoden und Formalismen vereinfachen und unterstützen. Insbesondere bringt der Werkzeugeinsatz in den frühen Phasen erhebliche Qualitätsverbesserungen. Viele Werkzeuge besitzen Prüffunktionen, mit denen sich Ergebnisse qualitativ bewerten lassen.

Das Werkzeug, welches die vorher festgelegte favorisierte OOSE-Methode unterstützt und auch von der Projektgruppe *PROMETHEUS* erfolgreich angewendet wurde (vgl. [BCF+97]), ist Objectory SE. Objectory (<u>Object</u> fact<u>ory</u>) ist ein konfigurierbares System, d.h. daß dieses Werkzeug an der organisatorischen Struktur des Projektes und des Entwicklerteams angepaßt werden kann. Dieses Werkzeug unterstützt die Durchführung des Projektes in den Phasen Analyse und Entwurf und erzeugt für jede Entwicklungsphase verschiedene Dokumenttypen (vgl. Abbildung 4). Falls die Implementierung des Software-Systems in C++ oder Smalltalk erfolgt, so generiert Objectory aus dem Entwurfsmodell einen Coderahmen (*Outline*).

Aus der Arbeit der Projektgruppe *PROMETHEUS* hat sich herauskristallisiert, daß die Qualitäten des Werkzeuges Objectory SE für Einzelentwickler und für weniger umfangreichere Systementwicklungen nicht bedeutend relevant sind, sondern erst bei Teamarbeit und großen Systeme zum Vorschein kommen. Deshalb hat sich zwar in dieser Arbeit die Struktur der Dokumente, die von der OOSE-Methode gefordert werden, weitgehend an die Struktur der von Objectory SE erzeugten Dokumente orientiert, diese wurden jedoch mit Standard-Werkzeugen konstruiert. Außerdem ist am Lehrstuhl X des Fachbereiches Informatik die Objectory-Lizenz während der Durchführung dieser Arbeit ausgelaufen.

Die für diese Arbeit eingesetzte Programmiersprache wurde von der Firma ROKD GmbH vorgegeben und ist Visual Basic Version 3.0 (VB3) unter MS-Windows Version 3.1. Visual Basic ist eine ereignisorientierte Programmiersprache und beruht auf dem Prinzip der externen Kontrolle. Die externe Kontrolle unterstützt die Steuerung des Kontrollflusses einer Anwendung und hat mit der Entwicklung von graphischen Benutzungsschnittstellen enorm an Bedeutung gewonnen (vgl. [Püt95]). Bei der externen Kontrolle hat der Benutzer die Freiheit, die Reihenfolge seiner Aktionen selbst zu bestimmen. Er besitzt selbst die Kontrolle über das Programm und entscheidet, was als nächstes geschehen soll. Folgendes Beispiel stammt aus [Püt95] und soll den Begriff der externen Kontrolle praktisch verdeutlichen:

„Beispiel:

Ein Programm soll folgende Gleichung lösen:

$z=2x+y$

Der Benutzer legt durch seine Eingabe die Werte für x und y fest. Das Ergebnis wird schließlich berechnet und ausgegeben.

Bei einer internen Kontrolle wäre der Benutzer gezwungen, beispielsweise zuerst den Wert x und dann den Wert y zu belegen. Er hat nicht die Wahl zu bestimmen, welchen Wert er zuerst eingeben möchte und wird zu einer Reihenfolge der Eingabe, und damit der Bedienung des Programmes gezwungen.

Bei einer externen Kontrolle hat der Benutzer die Wahl, welchen Wert er zuerst eingeben möchte. Das Programm reagiert entsprechend auf die Eingaben des Benutzers.“

VB3 stellt verschiedene Komponenten zur Verfügung, mit denen die Implementierung einer Anwendung möglich ist. Sie werden in der Konstruktionsphase des OERR-Systems (vgl. Kapitel 6) eingesetzt. Für weitere ausführlichere Informationen wird auf [Püt95], [Sta94] und [MR93] verwiesen.

2.3 Analytische Elemente des Software-Engineerings zur Qualitätssicherung

Die prüfenden Aktivitäten des V-Modells nach Boehm (vgl. Abbildung 3) umfassen Validierungs- und Verifikationstätigkeiten und werden in diesem Kapitel vertieft. Um eine Basis für die weitere Diskussion zu schaffen, werden folgende Begriffe definiert:

Analytische Elemente sind „all jene Maßnahmen, die zur Erkennung und Lokalisierung von Mängeln und Fehlern im weitesten Sinne dienen, d.h. alle Maßnahmen, die zur Bewertung der Qualität dienen“ [Wal90].

Validierung ist die Prüfung und die Bewertung eines Software-Produktes am Ende des Entwicklungsprozesses, um die Übereinstimmung des Produktes mit den Produktanforderungen nachzuweisen (vgl. [Wal90]). Diese Prüfung wird mit Hilfe von analytischen Elementen durchgeführt.

Verifikation wird in jeder Phase und zwischen jeden Phasen der Software-Entwicklung durchgeführt. Die Verifikationstätigkeit bestimmt, ob jedes Phasenprodukt „korrekt, komplett und konsistent mit sich selbst und mit dem vorigen Produkt aus der vorigen Phase ist“ ([Wal90]).

Durch Validierung wird unter anderem versucht, folgende Fragen zu beantworten:

- Erfüllt das Produkt die Anforderungspezifikation?
- Kann das installierte Produkt einfach modifiziert und an die sich ändernden Benutzerbedürfnisse angepaßt werden?
- Kann das Produkt effizient in der Benutzerumgebung betrieben werden?

Durch Validierungs- und Verifikationsaktivitäten, die in einem Software-Validierungs- und -Verifikationskonzept festgehalten sind, wird versucht, Fehler so früh wie möglich im Life Cycle zu entdecken und zu beseitigen, Projektrisiken und -kosten zu reduzieren und die Produktqualität zu verbessern. Die Validierungsaktivitäten beginnen mit der Planung und Spezifikation von Qualitätsanforderungen. Ein Hilfsmittel dafür sind *Qualitätsmodelle*. Diese bilden das erste zu betrachtende analytische Element zur Qualitätssicherung. Nachdem der Entwickler festlegt, welche Qualitäten ein Software-Produkt haben soll, werden diese durch Prüfungen in weiteren Validierungs- und Verifikationstätigkeiten bewertet. Die zur Verfügung stehenden Qualitäts-Prüfmethoden lassen sich in zwei Kategorien einteilen, in *statische* und *dynamische* Prüfungen. Diese bilden die beiden weiteren, in diesem Kapitel betrachteten analytischen Elemente zur Qualitätssicherung.

2.3.1 Qualitätsmodelle

Zur Spezifikation, aber auch zur Prüfung des Erfüllungsgrades von Qualitätsanforderungen, werden Hilfsmittel benötigt, die sich einerseits auf den Entwicklungsprozeß, andererseits auf das Produkt beziehen. Differenzierte und den Erfordernissen von Software-Produkten angepaßte Möglichkeiten zur Spezifikation von Qualitätsanforderungen, die auch die Anforderungen an den Prozeß berücksichtigen, bieten *Qualitätsmodelle* (vgl. [Wal90]). Mit Hilfe eines Qualitätsmodells wird der allgemeine Qualitätsbegriff durch Ableiten von Unterbegriffen operationalisiert (vgl. Abbildung 5). Die einzelnen Unterbegriffe werden durch Festlegen von Indikatoren (Produkt- oder Prozeßkenngrößen) meß- und bewertbar gemacht. Diese Größen werden *Qualitätskenngrößen* genannt.

Ein wesentlicher Aspekt der Qualitätsmodelle ist die Zerlegungssystematik von Qualitätseigenschaften, die sich auf der untersten Ebene auf Qualitätskenngrößen abstützten. Jede Eigenschaft wird durch Merkmale näher bestimmt und für jedes Merkmal werden Qualitätskenngrößen zur quantitativen Bewertung angegeben.

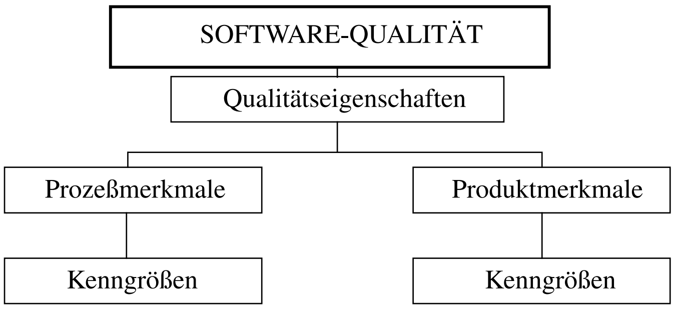

Abbildung 5: Struktur eines Qualitätsmodells

In der Vergangenheit wurden verschiedene Qualitätsmodelle entwickelt, wie beispielsweise von McCall ([MRW77]), Wallmüller ([Wal90]) und Dromey ([Kit96]). In Abhängigkeit von dem vorgegebenen Qualitätsmodell und den spezifischen Produktanforderungen wird für jedes Software-Produkt eine spezifische Menge von Eigenschaften, Merkmalen und Qualitätskenngrößen festgelegt, wobei deren Anzahl stark variieren kann. So hat beispielsweise McCall im Jahre 1977 in seinem Modell elf Eigenschaften vorgeschlagen (u.a. Effizienz, Integrität, Zuverlässigkeit) und aufgezeigt, wie er damit Software-Qualität bewertet. Das Modell liefert unter anderem eine Methode, die die Software-Qualität über den gesamten Life-Cycle des Produktes berücksichtigt. Die Nachteile dieses Modells liegen, so in [Wal90], in der teilweise unpräzisen und sich überschneidenden Definition der Eigenschaften und Merkmale. Für eine umfassende Beschreibung des McCall-Qualitätsmodells sei auf [MRW77] verwiesen.

Von einer anderen Einteilung der Qualitätseigenschaften wird im SPARDAT-Qualitätsmodell ausgegangen. SPARDAT hat als Basis das Modell des Softwaretest e.V. und wurde von E. Wallmüller im Jahre 1987 entwickelt (vgl. [Wal90]). Die Zielsetzung bei der Entwicklung des SPARDAT-Qualitätsmodells und insbesondere bei der Definition der Qualitätseigenschaften und Merkmale liegt darin, ein Bewertungssystem für Qualitätsaussagen zu schaffen, das in den einzelnen Phasen der Software-Entwicklung einsetzbar ist und es erlaubt, die sich entwickelnde Qualität abzuschätzen. Das SPARDAT-Qualitätsmodell dient als Hilfsmittel, um Qualitätsziele und Qualitätsanforderungen operativ auszuwählen und präzise und vollständig zu spezifizieren (vgl. [Wal90]). Das vorliegende Modell kann entsprechend den Bedürfnissen des Entwicklungsprozesses und der zu realisierenden Anwendungen angepaßt werden. Es kann in allen Entwicklungsphasen angewendet werden.

Ein neueres Qualitätsmodell wurde von G. Dromey im Jahr 1995 entwickelt und basiert nicht, so wie die Qualitätsmodelle von McCall und Wallmüller, auf Kenngrößen. Die Qualitätseigenschaften werden anhand einer systematischen Klassifizierung der möglichen auftretenden Qualitätsdefekte definiert. Für die Entdeckung dieser Qualitätsdefekte wurde von Dromey ein entsprechendes Werkzeug (vgl. [DR93]), sowie die Programmiersprache SAFE entwickelt (vgl. [OD94]). Das Dromey-Modell verlangt eine ausreichende Einarbeitungszeit und kann nur für die Entwicklungsphasen Entwurf und Implementierung angewendet werden. Für eine ausführlichere Beschreibung des Modells wird auf [Kit96] verwiesen.

Zusammenfassend läßt sich nachvollziehen, daß Qualitätsmodelle ein Hilfsmittel für die Verbesserung der Prozeß- und damit der Produktqualität darstellen. Ihr Einsatz führt zu einer Erhöhung der Fehlerentdeckungsrate beim Testen, zu einer Verbesserung der Produktivität (im Sinne von Leistung pro Zeiteinheit) und zu einer Kostenreduktion in der Wartung.

Qualitätsmodelle werden im Kapitel 4.1 bei der Identifizierung und Klassifizierung der Validierungsziele, als Grundlage des Validierungskonzeptes angewendet.

2.3.2 Statische Prüfungen

Bei der Betrachtung der beschriebenen Qualitätsmodelle kann festgestellt werden, daß es innerhalb einiger Qualitätsmodelle möglich ist, Kenngrößen in Form von bekannten Maßen anzugeben, wie beispielsweise das Maß von McCabe für die Bewertung der Komplexität eines Moduls. Bei anderen Eigenschaften, wie beispielsweise der Verknüpfbarkeit verschiedener Software-Systeme, sind nur Kenngrößen möglich, die sich aufgrund von Erfahrungswerten mit Checklisten auswerten lassen. Dabei werden die Ja-/Nein-Antworten mit Punkten gewichtet und anschließend die Punktesumme berechnet. Es gibt demzufolge unterschiedliche Arten von Methoden, mit denen Qualitätseigenschaften ausgewertet werden können. Diese Methoden werden unter dem Begriff *statische Prüfmethoden* zusammengefaßt.

Zu den statischen Prüfungen gehören u.a. Audits, Reviews (Inspektionen, Walkthrougs), die statische Analyse mit Software-Werkzeugen und Korrektheitsbeweise (mathematische Programmverifikation). Während *Audits* konkrete Problemsituationen (Abweichungen des Ist-Zustands vom Soll-Zustand) identifizieren, bieten *Reviews* für eine informelle Dokumentation die beste Möglichkeit, Mängel und Abweichungen von Qualitätsvorgaben festzustellen (vgl. [Wal90]). Mit den Einsatz von Software-Werkzeugen nimmt auch die Möglichkeit der werkzeuggestützten *statischen Analyse* von Prozeßergebnissen, wie beispielsweise von Anforderungs-, Entwurfsdokumenten oder dem Quellcode zu. Mit dieser Analysetechnik können alle Ergebnisse analysiert werden, die nach einem vorgegebenen Formalismus aufgebaut sind. Die Klasse der Dokumente, für die die Prüfmethode *Korrektheitsbeweise* anwendbar ist, beschränkt sich auf Aufgaben, die sich mit rein formalen Beschreibungsmitteln spezifizieren lassen.

Welche dieser Prüfmethoden wann eingesetzt werden, hängt einerseits von der Auswahl der Methoden und Werkzeugen ab, die für den Software-Entwicklungsprozeß eingesetzt werden und andererseits von den zu prüfenden Validierungszielen eines Software-Produktes. Für nähere Erläuterungen über statische Prüfmethoden und deren Einsatzgebiete sei auf [Wal90] verwiesen.

2.3.3 Dynamische Prüfungen

Die dynamischen Prüfmethoden sind Verifikationsaktivitäten und bilden den Abschluß der in dieser Arbeit betrachteten analytische Elemente zur Qualitätssicherung. Bei diesen Prüfungen wird, im Gegensatz zu den statischen Prüfmethoden, das Prüfobjekt ausgeführt. Ein Programm wird mit einer stichprobenartig ausgewählten Menge von Eingabewerten getestet. Dabei wird geprüft, ob sich das Programm so verhält, wie es in der Spezifikation gefordert wird. Wichtigstes dynamisches Prüfverfahren ist das *Testen* (vgl. Abbildung 3).

Allgemeines Ziel des *Testens* ist es, „die Qualität von Software-Systemen zu bestätigen durch systematisches Ausführen der Software unter sorgfältig kontrollierten Umständen“ ([Wal90]).

Debugging ist ein Prozeß, bei dem die Ursache eines Fehlers lokalisiert, dessen Korrektur überlegt, die Folgen der Korrektur geprüft und die Korrektur durchgeführt wird (vgl. [Wal90]).

Nach dem Entfernen eines Fehlers wird in der Regel der Test wiederholt, um zu prüfen, ob der Fehler auch tatsächlich korrigiert wurde und keine neuen Fehler eingeführt wurden. In der Literatur wird dafür der Begriff *Retesting* verwendet.

Es gibt verschiedene Testmethoden, wie beispielsweise die Methode der Funktionsabdeckung, die Äquivalenzklassenmethode usw. (vgl auch [Wal90]). Die dynamischen Prüfungen werden in der Testphase eines Entwicklungsprozesses angewendet. Ihre Auswahl hängt von der benutzten Methode und den Werkzeugen zur Software-Entwicklung, sowie von der angewendeten Programmiersprache ab.

3 Informationssysteme im Krankenhaus

Nachdem im Kapitel 2 die allgemeinen software-technologischen Grundlagen erläutert wurden, werden diese nun im Bereich des Gesundheitswesens, speziell beim Software-Entwicklungsprozeß von Krankenhausinformationssystemen, angewendet. Dabei handelt es sich um herkömmliche und modernere Ansätze für den Aufbau einer Standard-Architektur im KIS-Bereich, sowie um die Definition wesentlicher Merkmale eines Anforderungssystem.

3.1 Architekturen im KIS-Bereich

Um ein umfassendes Krankenhausinformationssytem zu realisieren ist es notwendig, die Systeme von verschiedensten Herstellern miteinander kommunizieren zu lassen. Daten, die in einem Subsystem eines KIS erzeugt werden, sollten direkt von einem anderen Subsystem übernommen werden, ohne daß diese Informationen manuell neu eingegeben werden müssen. Die aktuellen EDV-technischen Anforderungen (z.B. zur Kommunikation von Subsystemen) sind nur mittels einer durchgängigen EDV-technischen Unterstützung sowohl des medizinischen wie auch des administrativen Bereichs zu erfüllen. Demgegenüber steht jedoch die heutige Infrastruktur der genutzten EDV in Krankenhäusern mit ihrer großen Vielfalt von unterschiedlichen Hardware- und Softwarekomponenten, die aufgrund ihrer Eigenschaften oder ihrer proprietären Herkunft eine Integration miteinander stark erschweren. Viele derzeit verfügbare Systeme basieren zudem auf veralterten Technologien und werden modernen Anforderungen nicht mehr gerecht (vgl. [GS97]).

Um die schon getätigten Investitionen seitens der Krankenhäuser zu schützen und dennoch eine Migration zu modernen Applikationen zu ermöglichen, bietet sich die Entwicklung einer KIS-Architektur an, die eine Vernetzung und Integration existierender und zukünftiger Systeme gestattet.

Im folgenden werden konventionelle KIS-Architekturen betrachtet, so wie sie heute in den Krankenhäuser zu finden sind. Es folgt in den Kapiteln 3.1.2 und 3.1.3 ein Vorschlag für eine neue KIS-Architektur, die die wesentlichen Anforderungen eines modernen, leistungsfähigen krankenhausweiten Informationssystem in Betracht zieht und die sich als Standard in dem europäischen Krankenhausumfeld zu etablieren versucht.

3.1.1 Konventionelle KIS-Architekturen

Die ersten Abteilungen, die innerhalb der Krankenhäuser mit EDV-Lösungen ausgerüstet wurden, waren die Verwaltungen. Im weiteren Verlauf der Entwicklung wurden die Labore der Krankenhäuser durch EDV-Anlagen unterstützt. Durch die fortschreitende Entwicklung von Computersystemen für den medizinischen Bereich wurden im Laufe der Zeit zunehmend weitere Krankenhausabteilungen mit EDV ausgestattet. In der Regel konnte jede Station frei entscheiden, welches System installiert werden sollte. Eine Vorgabe für Kommunikationsschnittstellen zu anderen - bereits bestehenden - Systemen, gab es nicht. Die Notwendigkeit einer abteilungsübergreifenden Kommunikations- und Integrationslösung wurde nicht erkannt (vgl. [FFH+97]).

Diese Entwicklung hat dazu geführt, daß in fast allen Krankenhäusern Inselsysteme implementiert wurden, die für den jeweiligen Bereich zwar gute Lösungen darstellen, aber von einem integrierten Gesamtsystem, das die oben genannten EDV-technischen Anforderungen erfüllt, weit entfernt sind, da nur die wenigsten Systeme untereinander Daten austauschen können.

Konventionelle KIS-Architekturen basieren meistens auf monolitischen Modellen. Darauf beziehende Systeme zeichnen sich durch einen hohen Integrationsgrad ihrer Teilsysteme aus - sowohl hinsichtlich der Datenhaltung als auch funktional (vgl. [Blo97]). Jede Station bzw. jeder Funktionsbereich eines Krankenhauses verfügt in konventionellen KIS-Architekturen über eine eigene Datenbank, so daß ein Informationsaustausch auf elektronischem Wege zwischen den verschiedenen Datenbanken nicht möglich ist. Ein weiterer Nachteil solcher Insellösungen ist auch die doppelte Erfassung derselben Daten. Diese Redundanz enthält zwangsläufig die Gefahr der Inkonsistenz. Außerdem bedeuten diese Eingaben einen zusätzlichen Zeitaufwand für die Mitarbeiter des Krankenhauses und somit eine Erhöhung der Kosten. „Entwickelt, implementiert und integriert auf der Basis von Geschäftsprozeßmodellen, erfordern die einzelnen Anwendungssysteme im Krankenhaus jedoch soviel spezialisiertes Prozeßwissen, daß sich perspektivisch kein Allrounder am Markt halten kann" ([Blo97]). Dies sind nur ein paar der vielen Nachteile die eine solche konventionelle KIS-Architektur mit sich bringt. Damit kommt zukünftig der

Integration dezentralisierter, heterogener, offener Anwendungssysteme eine entscheidende Bedeutung zu.

Ein modernes und leistungsfähiges krankenhausweites Informationssystem wird daher in Zukunft folgende wesentlichen EDV-technische Anforderungen erfüllen müssen (vgl. auch [GS97]):

physikalische Integration: Schon existierende und zukünftige Anwendungen müssen auf der Grundlage von Netzwerken miteinander verbunden werden, um den effizienten Austausch von Daten zu ermöglichen.

logische Integration: Die in einer Anwendung enthaltenen Informationen müssen anderen Anwendungen zugänglich gemacht werden.

Ausfall- und Datensicherheit: Der zeitweilige Ausfall einer oder mehrerer Anwendungen in einer verteilten Umgebung darf nicht dazu führen, daß für die ausgefallenen Systeme relevante Daten verloren gehen.

Dies sind auch die zentralen Fragestellungen, mit denen sich das „Comitee European de Normalisation" (CEN) im Rahmen des Projektes CEN/TC251/PT010 beschäftigt hat. Ziel des Projektes war die Definition eines Standards für KIS-Architekturen, welcher eine einfache Integration nahezu beliebiger Anwendungen ermöglicht. Das Ergebnis ist ein Vorschlag für ein Basiskonzept eines Architekturgerüstes, das die oben genannten Anforderungen erfüllt. Dieses Architekturgerüst wird im folgenden beschrieben.

3.1.2 Das CEN-Architekturgerüst

In [CEN93] wird ein KIS wie folgt definiert:

„[...] a *federation of autonomous applications, which are individually optimised to the needs of the various units and users, and which rely on a set of common services, providing homogeneous and consistent mechanism for the management of the information relevant for the whole organisation and for the interworking of the various components*."

Das in [CEN93] vorgeschlagene Architekturgerüst für den medizinisch-technischen und klinisch-administrativen Krankenhausbereich läßt sich in drei Schichten strukturieren:

- die *Bit-Schicht* enthält die technologische Infrastruktur für die Netzwerkunterstützung;
- die *Middleware-Schicht* unterstützt die Kooperation verschiedener Anwendungen;
- die *Anwendungsschicht* unterstützt die spezifischen Anforderungen der Krankenhausabteilungen.

Abbildung 6 zeigt, wie die drei Schichten mit den Organisationsebenen und Notwendigkeiten des Krankenhauses in Zusammenhang stehen.

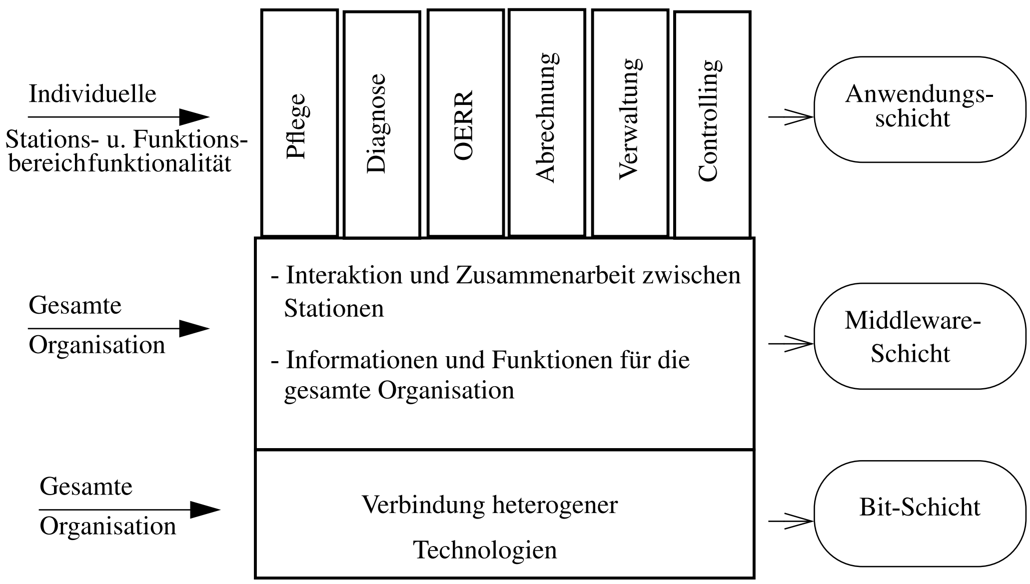

Abbildung 6: Der Zusammenhang zwischen den Organisationsebenen eines Krankenhauses und den Schichten des konzeptionellen Architekturgerüstes (vgl. [Fer95a])

So können beispielsweise in der Anwendungsschicht individuelle Stationsanwendungen implementiert werden (*Beispiel: Pflegesystem, OERR-System etc.*). Die Middleware-Schicht ist für die Interaktion und Zusammenarbeit zwischen den Stationen zuständig und stellt Informationen und Funktionen für die Kommunikation innerhalb der gesamten Organisation zur Verfügung. Die Bit-Schicht unterstützt die Netzwerkkommunikation in der gesamten Krankenhausorganisation.

Aufbauend auf diesem Architekturgerüst wurde im Rahmen der EU-Projekte RICHE und EDITH (vgl. [EDI93]) das *Distributed Healthcare Environment* (DHE) definiert, entworfen und implementiert. Diese Umgebung wird zur Zeit im Rahmen des EU-Projektes HANSA (Healthcare Advanced Networked System Architecture) (vgl. [Fer95]) in neun europäischen Länder, darunter auch in Deutschland durch die ROKD GmbH und an den Universitäten Gießen und Magdeburg, validiert.

DHE ist eine Integrationsplattform, die die Entwicklung und Implementierung neuer und vorhandener Anwendungen in verteilten, interagierenden Informationssystemen des Gesundheitswesens unterstützt (vgl. [Blo97]). Die Middleware-Schicht des beschriebenen Architekturgerüstes wird auf die DHE-Struktur abgebildet. Abbildung 7 dient dazu als Übersicht.

Die technologische Plattform NICE (Network Independent Communication Environment) entspricht der Bit-Schicht aus Abbildung 6 und ist für die transparente Integration der verteilten, heterogenen technologischen Umgebung zuständig, die im Krankenhaus vorhanden ist. Sie stellt die Protokolle für die Kommunikation zwischen den vorhandenen Komponenten eines KIS zur Verfügung (vgl. [Fer95a]). DHE entspricht der Middleware-Schicht und stellt Funktionen und Schnittstellen für die KIS-Anwendungen zur Verfügung. DHE liefert eine funktionelle Infrastruktur in Form domänenspezifischer Dienste (Services) mit ihren API. Diese Dienste unterstützen die Zusammenarbeit, Synchronisation und Konsistenz der Anwendungen. DHE wird im Kapitel 3.1.3 näher betrachtet. Die Anwendungsschicht ist für die Interaktion mit dem Benutzer zuständig und unterstützt die klinischen und verwaltungstechnischen Tätigkeiten eines Krankenhauses durch verschiedene Anwendungen. Diese Anwendungen werden spezifisch an die Bedürfnisse jeder Station und Funktionsbereich des Krankenhauses angepaßt.

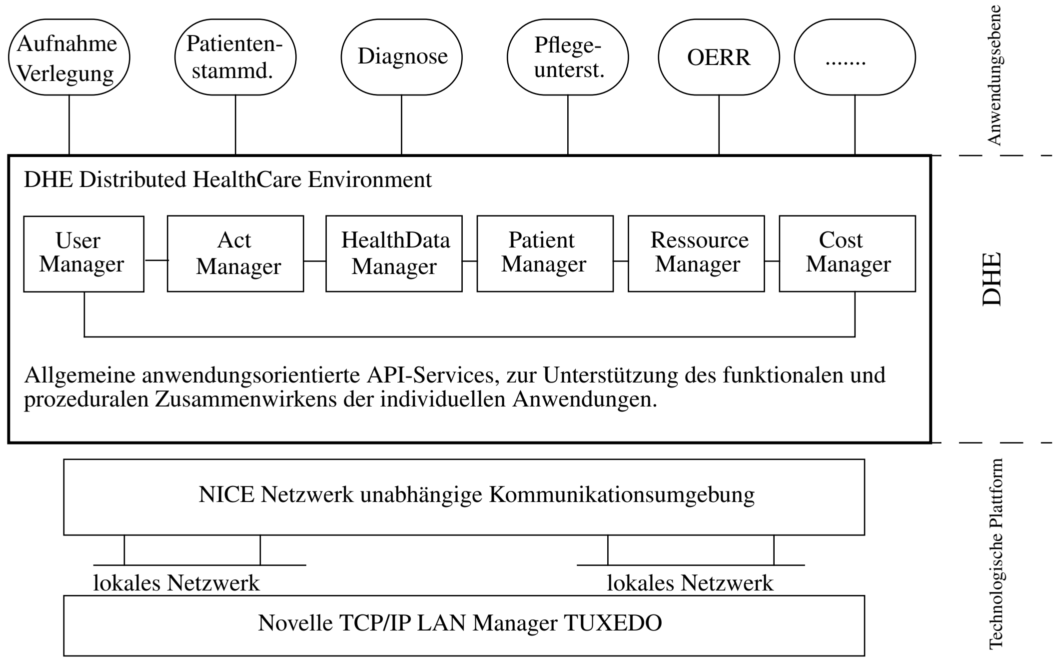

Abbildung 7: Die DHE-Struktur (vgl. [Fer95a])

Die Entwicklung und Integration einer neuen Anwendung in das in [CEN93] vorgeschlagene Architekturgerüst erfordert, im Unterschied zu konventionelle Architekturen, folgende Entwicklungsphasen (vgl. Abbildung 8):

- Entwurf und Implementierung des Client-Interaktionsmechanismuses,
- Entwurf und Implementierung der graphischen Benutzungsoberfläche der Anwendung.

Die Module DBMS, Serverfunktionen und -Interaktionsmechanismus sind in DHE integriert und stehen jeder Anwendung zur Verfügung.

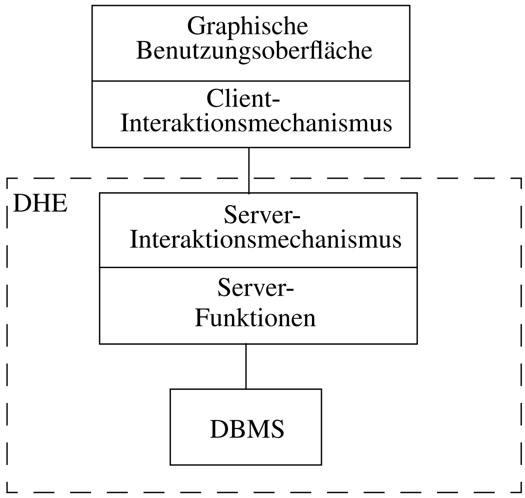

Abbildung 8: Entwicklungsmodule einer Anwendung in dem von CEN vorgeschlagenen Architekturgerüst (vgl. [Fer95a])

3.1.3 Beschreibung von DHE

DHE entspricht der Middleware-Schicht des von CEN vorgeschlagene Architekturgerüsts. DHE enthält etwa 1,5 Millionen Anweisungen, von denen 2/3 SQL- und 1/3 C-Anweisungen sind. Konzeptionell basiert DHE auf der Client-Server Technologie (vgl. [Fer95a]). Der DHE-Server befindet sich auf die DHE-Ebene und der DHE-Client auf der Anwendungsebene des CEN-Architekturgerüstes (vgl. Abbildung 7). Im folgenden wird die DHE-Architektur näher vorgestellt.

3.1.3.1 Der DHE-Server

Bestandteile des DHE-Servers sind eine relationale Datenbank und die Funktionen für die Verwaltung der Daten (vgl. Abbildung 9). Der DHE-Server kann in den Umgebungen SUN, Hewlett Packard, IBM oder Tandem installiert werden. Als DBMS unterstützt DHE Sybase, Informix und, seit Juli 1997, auch Oracle (vgl. [Fer95a]).

Der DHE-Server enthält die einzelnen Dienste, die entsprechend ihren Informationen und Funktionen zu „Managern" zusammengefaßt worden sind (vgl. Abbildung 7). In jedem Krankenhaus werden von verschiedenen Gruppen von Akteuren (Medizinisches Personal, Pflegepersonal, Verwaltungspersonal) diverse klinische und verwaltungstechnische Tätigkeiten durchgeführt. All diese Tätigkeiten werden in DHE durch den *Act Manager* verwaltet. Auf die Ressourcen, die jede Aktivität verwendet, kann über den *Ressource Manager* zugegriffen werden. Die Akteure werden in Benutzergruppen eingeteilt, die über unterschiedliche Authorisierungsprofile verfügen. Die Benutzerdaten werden durch den *User Manager* verwaltet. Alle patientenabhängigen Informationen werden von dem *Patient Manager* administriert. Die Ergebnisse einer durchgeführten Tätigkeit (seien es Pflegemaßnahmen, Therapiemaßnahmen oder Befundung einer Untersuchung) werden als Gesundheitsdaten bezeichnet und von dem *HealthData Manager* verwaltet. Die anfallenden Kosten einer jeden durchgeführten Tätigkeit werden von dem *Cost Manager* verwaltet.

Jeder dieser sechs Manager verfügt über eine Vielzahl von Funktionen (insgesamt 480), mit deren Hilfe auf Daten aus der Datenbank zugegriffen wird. Für die Verwaltung dieser Daten (Einfügen, Ändern, Löschen) werden die entsprechenden Manager aufgerufen. Dadurch bleibt die physikalische Struktur der Datenbank dem Entwickler einer auf DHE basierenden Anwendung verborgen (vgl. [Fer95]). Nutzer und Anwendungsentwickler werden auch von allen Fragen des Daten- und Transaktionsmanagements und der Dienstverteilung abgeschirmt.

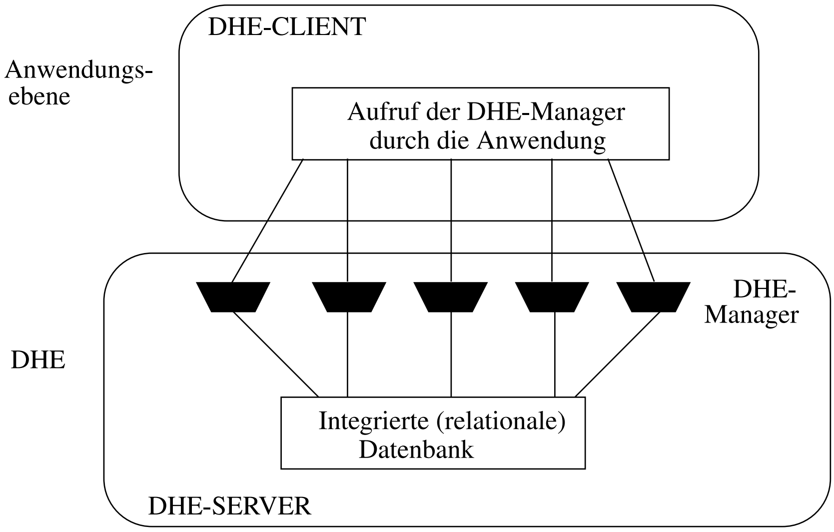

Abbildung 9: Der DHE-Server und der DHE-Client

3.1.3.2 Der DHE-Client

Die Client-Seite des DHE besteht aus PC's, die für die Interaktion mit dem Benutzer mittels einer graphischen Benutzungsoberfläche zuständig sind (vgl. [Fer95]). DHE stellt Bibliotheken für Windows 3.1, Unix und seit Mai 1997 auch für Windows 95 sowie Windows NT zur Verfügung. Die Anwendungen unter DHE können in den Programmiersprachen Visual Basic oder C implementiert werden.

3.1.3.3 Der Informationsaustausch zwischen dem DHE-Server und -Client

Jeder der sechs DHE-Manager verwaltet *deskriptive* und *dynamische* Informationen (vgl. [Fer96]). Die deskriptiven Informationen sind statisch und beschreiben die Struktur der Daten in der Datenbank und die organisatorische Anforderungen. Daten, welche die Benutzer bei ihrer täglichen Aktivitäten erzeugen, stellen dynamische Informationen dar. Sie beinhalten persönliche und medizinische Patientendaten, aktuell durchgeführte Tätigkeiten, medizinische Ressourcen etc.

Wenn der Benutzer dynamische Daten einfügen, ändern oder löschen möchte, so startet der DHE-Server eine Konsistenzprüfung dieser Daten mit dem vorhandenen deskriptiven Teil (vgl. Abbildung 10). Dadurch wird die Konsistenz der Daten innerhalb des Gesamtsystems sichergestellt.

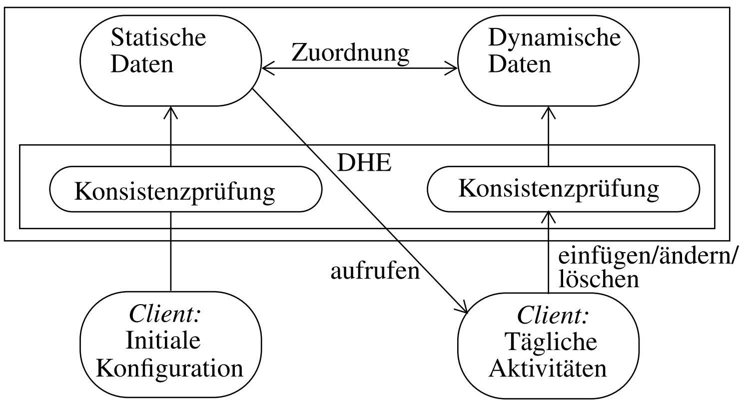

Abbildung 10: Die grundlegende Struktur aller DHE-Informationen (vgl. [Fer96])

In Anlehnung an [Fer95a] bietet DHE nicht nur eine Lösung für die Ist-Situation der EDV in Krankenhäuser, sondern, aufgrund der Konzeption als offene Client-Server-Architektur, eine plattformunabhängige und technologisch zukunftsorientierte Basis für den Aufbau eines verteilten krankenhausweiten Informationssystems. Inwieweit die DHE-Software Teile einer Standardarchitektur implementiert, um moderne und offene KIS zu entwickeln, anzupassen, zu integrieren und erfolgreich und effizient betreiben zu können, wird anhand von verschiedenen Erfolgsfaktoren in den folgenden Kapiteln der vorliegenden Arbeit validiert.

3.2 Das Order-Entry-and-Result-Reporting-System

In einem Krankenhaus existieren vielfältige Informationssysteme. Sie werden für die schnelle Bereitstellung und Verarbeitung aller anfallenden Daten benötigt. Die Gesamtheit dieser zentralen oder autonomen Einheiten wird als Krankenhausinformationssystem bezeichnet. Zu diesen Einheiten zählt auch das Anforderungssystem (Order-Entry-and-Result-Reporting-System).

In diesem Kapitel werden die allgemeinen Anforderungen an ein Order-Entry-and-Result-Reporting (OERR) -System festgelegt und das proCOM-OERR-System der Firma ROKD GmbH näher betrachtet.

3.2.1 Allgemeine Beschreibung

Ein OERR-System ist ein Anforderungssystem, das die Verwaltung und Durchführung von Anforderungen, die auf einer Krankenhausstation initiiert werden, unterstützt und diese an die verschiedenen Funktionsbereiche des Krankenhauses weiterleitet (*Order Entry*). Nach der Durchführung der Anforderungen übermittelt das OERR-System die Anforderungsergebnisse von den Funktionsbereichen zu den Stationen (*Result Reporting*).

Die folgende Beschreibung des Ist-Zustandes eines OERR-Vorganges orientiert sich an der im Rahmen der Projektgruppe PROMETHEUS durchgeführten Ist-Analyse im Mathias-Spital in Rheine (vgl. [BCF+97]) und wird deshalb nur kurz beschrieben. Da die deutsche Sprache keine gefällige geschlechtsneutrale Ausdrucksweise zuläßt, wird in dieser Arbeit, aus Rücksicht auf die Lesegewohnheiten der Leserinnen und Leser, vorwiegend die männliche Form benutzt. Infolgedessen werden die üblichen Schreibweisen wie bspw. Stationsarzt verwendet, wohingegen der Ausdruck Stationsschwester auch die männlichen Stationspfleger miteinbezieht.

Wird für einen Patienten eine Untersuchung, die nicht auf der Station durchgeführt werden kann, von dem Stationsarzt angeordnet, so muß die Stationsschwester ein entsprechendes Formular ausfüllen und telefonisch einen Termin bei dem betroffenen Funktionsbereich anfordern. Das ausgefüllte Formular wird anschließend von dem Stationsarzt unterzeichnet. Der Patient wird dann mit dem Formular, abhängig von dem Zustand in der er sich befindet, zu diesem Funktionsbereich geschickt oder gebracht. Nachdem der Patient untersucht wurde, wird dasselbe Formular um das Ergebnis der Untersuchung ergänzt und der Station, welche die Untersuchung angefordert hat, zugeleitet. Eine Kopie des Formulars verbleibt auf dem Funktionsbereich.

An diesem Geschehen sind demnach zwei Kommunikationspartner beteiligt (vgl. Abbildung 11): auf der einen Seite der Kommunikation steht die Station, welche die gewünschte Leistung anfordert und auf der anderen Seite steht der Funktionsbereich (beispielsweise Labor), welcher die angeforderte Leistung durchführt und dokumentiert. Diese beiden Partner einer Kommunikation im OERR-System werden als *anfordernde* bzw. *leistende Kommunikationseinheit* (KE) bezeichnet. Demzufolge wird ein System benötigt, das die Funktionen des *Order Entry* - Teils auf der anfordernden KE bzw. des *Result Reporting* - Teils auf der leistenden KE unterstützt.

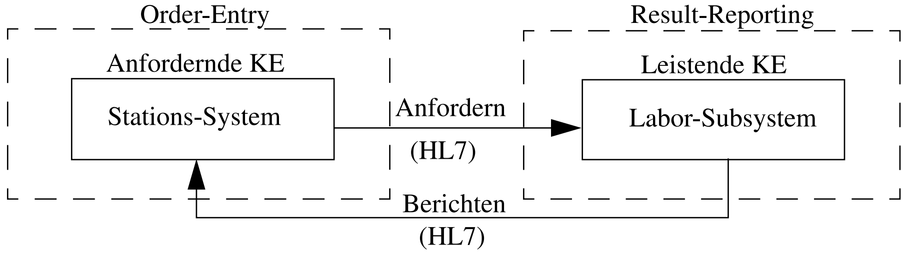

Abbildung 11: OERR

Die wesentlichen Anforderungen an ein OERR-System sind u.a. die Unterstützung des beschriebenen Prozesses von der Generierung der Leistung auf der anfordernden KE bis zu der Dokumentation der Ergebnisse, die Nutzung eines Kommunikationsprotokolls zwischen der anfordernden und der leistenden KE, sowie die Integration aller von den KE's genutzten OERR-Operationen und -Funktionen in graphische Benutzungsoberflächen.

Die Firma ROKD GmbH hat ein OERR-System entworfen und entwickelt, welches den Namen proCOM-OERR trägt. Es ist geplant, dieses System im Mathias-Spital in Rheine einzusetzen, um dort die Abwicklung der Laboranforderungen zu unterstützen. Der Funktionsbereich Labor besitzt ein eigenes Labor-Subsystem. In diesem Fall werden die Informationen der leistenden KE (Labor) über den Kommunikationsstandard HL7 mit der Station ausgetauscht (vgl. Abbildung 11). HL7 (Health-Level-7), abgeleitet aus der siebten Schicht des ISO/OSI-Modells, ist ein herstellerunabhängiges Übertragungsprotokoll, das den Datenaustausch zwischen medizinischen Subsystemen eines Krankenhauses standardisiert (vgl. [GS97] und [Pac94]). HL7 ermöglicht somit die offene Kommunikation zwischen Anwendungssystemen durch Standardisierung von Syntax und Semantik ereignisgetrieben ausgetauschter Mitteilungen (vgl. [SL97]).

Das proCOM-OERR-System ist in dieser Arbeit die Grundlage für die zu konstruierende OERR-Anwendung unter DHE und wird daher im folgenden kurz beschrieben.

3.2.2 OE auf der anfordernden Kommunikationseinheit

Auf der anfordernden KE entstehen im wesentlichen im Rahmen der Visite die Anforderungen an die leistungserbringenden Stellen des Krankenhauses. Das proCOM-OERR bietet hierfür folgende Funktionen (vgl. auch [ROKD96]):

- Funktionen zur Generierung von Leistungen auf den jeweiligen Stationen.
- Generierung von Begleitpapieren (z.B. Etiketten für Probenröhrchen).
- Informationen über den Bearbeitungsstatus der von der Station angeforderten Leistungen, sowohl in patientenindividueller Sichtweise, als auch in einer Sicht der Gesamtstation.
- Freigabe-, Ergänzungs- und Verifikationsmechanismen für besondere Leistungen (*Beispiel: der Arzt muß für eine angeforderte Laboruntersuchung eine klinische Fragestellung ergänzen*).
- Informationen über Ergebnisse und Befunde der angeforderten Leistungen.
- Druckfunktionen für diese Funktionen.

3.2.3 RR auf der leistenden Kommunikationseinheit

Falls von einem Funktionsbereich kein eigenes Subsystem für das Result Reporting zur Verfügung gestellt wird, kann eine leistende KE mit der speziell für diese Stelle entwickelten Teilkomponente von proCOM-OERR unterstützt werden. Die gesamte Anforderungsverarbeitung bzw. Ergebnisrückmeldung erfolgt in diesem Fall innerhalb von proCOM.

Für die leistenden KE's bietet das proCOM-OERR-System folgende Funktionen (vgl. auch [ROKD96]):

- eine Übersicht über die neu eingetroffenen Anforderungen mit Wunschtermin;
- eine Übersicht über den Bearbeitungsstand der noch nicht komplett abgearbeiteten Anforderungen;
- Dokumentationsfunktion für Ergebnisse, Kurzbefunde, Kommentare zur Anforderung;
- Druckfunktionen.

3.2.4 Stammdaten und Parameter

Das proCOM-OERR-System benötigt für eine korrekte Funktionalität sogenannte Stammdaten und Parameter, die im folgenden kurz beschrieben werden. Die Basis für die Stammdaten des proCOM-OERR-Systems bilden (vgl. [ROKD96])

- der Leistungskatalog von proCOM mit Zusatzinformationen;
- die Einrichtungsstruktur von proCOM;
- die Personal-Berechtigungsdaten.

Der Leistungskatalog enthält alle aktuell zur Verfügung stehenden Einzel- und Gruppenleistungen. Dies ermöglicht die Generierung standardisierter und reproduzierbarer Anforderungen von jeder definierten anfordernden KE. Die proCOM-Einrichtungsstruktur ermöglicht die individuelle Definition der einzelnen KE's mit ihren Attributen. Es werden u.a. die Leistungserbring- und -empfang-Mengen festgelegt. Ein Beispiel für eine Leistungserbring-Menge wäre: die leistende KE „EKG“ kann die Leistungen „Rhythmus-EKG“, „Nehb'sche Ableitung“, „Ruhe-EKG“ etc. erbringen. Beispiel für eine Leistungsempfang-Menge ist: die leistende KE „EKG“ kann für die anfordernde KE „Innere Station“ die Leistungen „Ruhe-EKG“ und „Rhythmus-EKG“ erbringen.

Für das spezielle OERR-Umfeld können darüberhinaus die folgenden Parameter und Stammdaten hinterlegt werden:

Für die anfordernde KE:

- Welche Leistungen können von welcher leistenden KE angefordert werden?
- Welches sind die regelmäßig angeforderten Leistungen der anfordernden KE?
- Informationen zu Ausdruckfunktionen.

Für die leistende KE:

- Welches Spektrum deckt die Leistungsstelle ab?
- Wird mit oder ohne Subsystem gearbeitet?
- Informationen zu Ausdruckfunktionen.

3.2.5 OERR und DHE

Nachdem in Abbildung 11 die logische Sicht des OERR-Systems dargestellt wurde, verdeutlicht Abbildung 12 beispielhaft die Rollen, die der DHE-Server und -Client innerhalb des OERR-Systems einnehmen (physikalische Sicht, vgl. Abbildung 12):

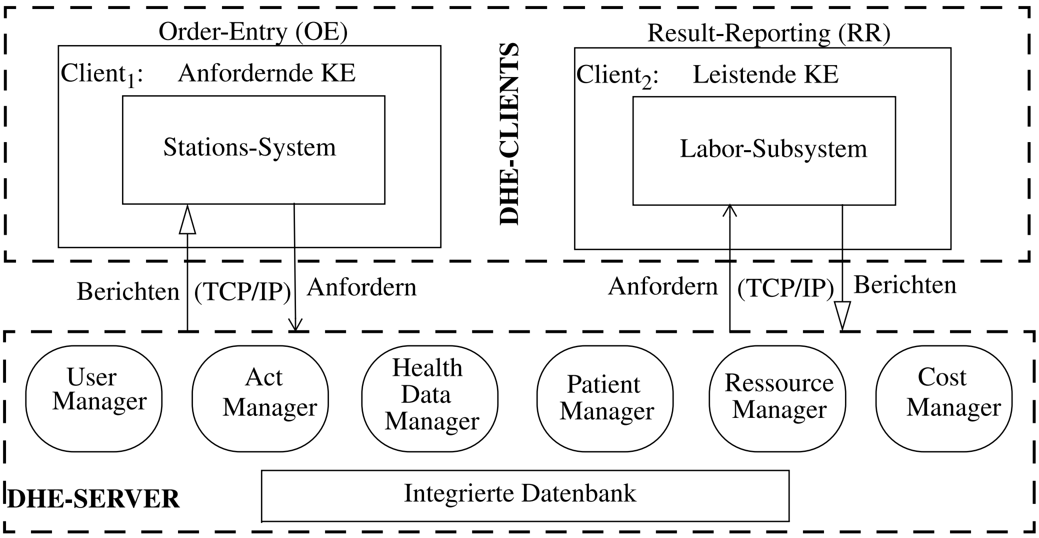

Abbildung 12: OERR unter DHE

Die anfordernde und die leistende KE stellen zwei DHE-Clients dar. Die Kommunikation zwischen den beiden Teilanwendungen (OE- und RR-Komponenten des OERR-Systems) erfolgt über den DHE-Server: jede von Client_1 angeforderte Leistung wird über das TCP/IP-Kommunikationsprotokoll an den DHE-Server gesendet. Dieser leitet sie an die betreffende leistende KE weiter, die von Client_1 angegeben wurde. In der entgegengesetzten Richtung werden die Ergebnisse einer durchgeführten Leistung in derselben Weise über das TCP/IP-Protokoll von Client_2 (leistende KE) zu Client_1 (anfordernde KE) zurückgeschickt. Der DHE-Server stellt beiden Clients die notwendigen Informationen zur Verfügung. Eine detaillierte DHE-Kommunikationsbeschreibung innerhalb des OERR-Systems wird in den Kapiteln 5 bis 7 gegeben.

4 Das Validierungskonzept

In diesem Kapitel wird ein Konzept aufgestellt und beschrieben, das als Ziel die Validierung der DHE-Software auf der Basis des zu konstruierenden OERR-Systems hat. Die Beschreibung der durchgeführten Validierung erfolgt anhand der in diesem Kapitel festgelegten Validierungsziele, Auswertungsmethoden und charakteristischen OERR-Vorgänge, in den Kapiteln 5 bis 7.

Das Validierungskonzept basiert auf den im Kapitel 2.3 beschriebenen Validierungs- und Verifikationstätigkeiten. Es beginnt im Kapitel 4.1 mit der Planung und Spezifikation der Validierungsziele mittels Qualitätsmodellen. In den nachfolgenden Aktivitäten (beschrieben im Kapitel 4.2) werden Auswertungsmethoden für diese spezifizierten Validierungsziele festgelegt. Kapitel 4.3 vereint in einem Validierungsplan sowohl die konstruktiven als auch die analytischen Elemente zur Qualitätssicherung. Es werden für jede Entwicklungsphase des Software Life-Cycles die Validierungsziele, Auswertungsmethoden und Ergebnisdokumente festgelegt. Anhand der festgelegten prozeß- und produktrelevanten Qualitätsmaßstäbe werden im Kapitel 4.4, zwecks Validierung der DHE-Software, charakteristische Vorgänge der zu konstruierenden OERR-Anwendung bestimmt.

4.1 Identifizierung und Klassifizierung der Validierungsziele

Seit Mitte 1995 ist der *Geschäftsbereich Softwarehaus* der ROKD GmbH nach DIN EN ISO 9001 prozeßzertifiziert. Auf der Basis der ISO 9001 prozeßrelevanten Qualitätsmaßstäbe werden von der ROKD GmbH zur Validierung der DHE-Software folgende geeignete produktrelevante Qualitätsmerkmale vorgegeben, die sich an der deutschen Norm DIN 66285 orientieren (vgl. [Lor96a]):

Funktionale und technische Produkteigenschaften und -merkmale: Robustheit, Funktionalität der API-Komponenten, Abstraktionstiefe und Kapselung, Portierfähigkeit in die Unix-Welt, Benutzerdokumentation, Prüfbarkeit, Wiederverwendbarkeit, Erlernbarkeit, Anpaßbarkeit, Performance und Effizienz, Zuverlässigkeit und Sicherheit;

Benutzerorientierte Produkteigenschaften und -merkmale: Benutzungsfreundlichkeit, Benutzerdokumentation;

Allgemeine und phasenspezifische Eigenschaften und Merkmale: Funktionsabdeckung, Funktionale Widerspruchsfreiheit, Korrektheit, Einheitlichkeit, Transparenz, Effektivität etc.

Die ISO-Norm verlangt für die Sicherstellung einer korrekten und vollständigen Spezifikation von Qualitätsanforderungen an ein Software-Produkt eine hierarchische Strukturierung der Produkteigenschaften und -merkmale: nachdem die Besonderheiten einer Anwendung identifiziert wurden, werden dessen wichtigste Qualitätseigenschaften bestimmt. Diese Eigenschaften werden weiterhin in Prozeß- und Produktmerkmale eingeteilt. Die Merkmale werden durch passend ausgewählten Qualitätskenngrößen ausgewertet. Die ISO-Norm stellt die Auswahl der konstruktiven und analytischen Qualitätssicherungsmaßnahmen (Methoden, Werkzeuge, Qualitätsmodelle etc.) frei.

Im Einklang mit den Anforderungen der ISO-Norm an die Klassifizierung der Validierungsziele und den oben aufgezählten Qualitätskriterien, erscheint der Einsatz des SPARDAT-Qualitätsmodells geeignet für die Strukturierung dieser Eigenschaften und Merkmale, aber auch für die Erfüllung der entsprechenden ISO-Norm. Deshalb wird dieses Modell gegenüber den anderen, im Kapitel 2.3.1 vorgestellten Modelle, in dieser Arbeit bevorzugt und angewendet.

Das SPARDAT-Qualitätsmodell (vgl. [Wal90]) teilt die Qualität eines Software-Produktes in genau drei Qualitätseigenschaften ein: die Gebrauchstauglichkeit, die Pflegbarkeit und die Anpaßbarkeit (vgl. Abbildung 13).

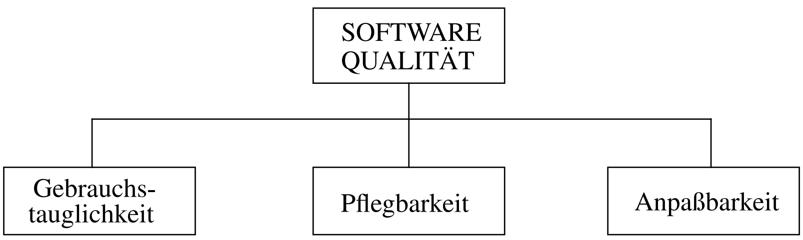

Abbildung 13: Grobstrukturierung der Eigenschaften

Diese drei Kategorien überdecken die oben beschriebenen prozeß- und produktrelevanten Qualitätskriterien zur Validierung der DHE-Software folgendermaßen (vgl. Abbildung 14): die Qualitätseigenschaft `Gebrauchstauglichkeit` gibt Informationen darüber, wie gut sich das Produkt in seinem aktuellen Zustand zur Erfüllung gegebener Aufgaben verwenden läßt. Die `Pflegbarkeit` beschäftigt sich mit der Problematik der Änderung bzw. Weiterentwicklung des Produktes. Die `Anpaßbarkeit` gibt Informationen darüber, wie leicht sich das Produkt in anderen Umgebungen oder Anwendungsgebiete übertragen läßt.

- Gebrauchstauglichkeit
  - Funktionstüchtigkeit
    - Funktionsabdeckung
    - Widerspruchsfreiheit
  - Betriebstüchtigkeit
    - Korrektheit
    - Zuverlässigkeit
    - Sicherheit
    - Robustheit
    - Effizienz
    - Verknüpfbarkeit
  - Benutzungsfreundlichkeit
    - Erlernbarkeit
    - Handhabbarkeit
    - Effektivität
    - Einheitlichkeit
- Pflegbarkeit
  - Wartbarkeit
    - Änderbarkeit
    - Korrigierbarkeit
    - Erweiterbarkeit
  - Prüfbarkeit
- Anpaßbarkeit
  - Portabilität
  - Wiederverwendbarkeit

Abbildung 14: Gesamtüberblick über die Qualitätseigenschaften (vgl. [Wal90])

Jede dieser Eigenschaften lassen sich durch weitere Qualitätseigenschaften verfeinern. So enthält beispielsweise die Gruppe Gebrauchstauglichkeit die Eigenschaften `Funktionstüchtigkeit`, `Betriebstüchtigkeit` und `Benutzungsfreundlichkeit`. Die Pflegbarkeit läßt sich in die Qualitätseigenschaften `Wartbarkeit` und `Prüfbarkeit` zerlegen. Die Anpaßbarkeit wird durch `Portabilität` und `Wiederverwendbarkeit` weiter verfeinert.

Es gibt Qualitätseigenschaften, wie beispielsweise die Portabilität, die nicht mehr durch andere Eigenschaften konstituiert werden können. Sie werden *Basiseigenschaften* genannt. So hat z.B. die Eigenschaft `Benutzungsfreundlichkeit` als Basiseigenschaften die `Erlernbarkeit`, die `Effektivität` und die `Einheitlichkeit`.

In Anlehnung an Kapitel 2.3.1 und [Wal90] wird jede Basiseigenschaft durch Merkmale näher bestimmt und für jedes Merkmal werden Qualitätskenngrößen zur quantitativen Bewertung angegeben. Dieser Ansatz ermöglicht eine hierarchische Klassifizierung der identifizierten Qualitätskriterien zur Validierung der DHE-Software. Die Definitionen der Basiseigenschaften entsprechen den Definitionen im SPARDAT-Modell (vgl. [Wal90]) und werden zusammen mit den Merkmalen und für deren Validierung identifizierten Auswertungsmethoden im folgenden beschrieben. Abweichungen von diesem Modell werden an den Stellen gemacht, wo die SPARDAT-Merkmale industriebezogen und für das Krankenhausumfeld nicht passend sind. Diese Merkmale werden an den Krankenhausbereich angepaßt und orientieren sich an DHE und an dem konkret entwickelten OERR-System.

Innerhalb dieser Arbeit werden, um Lesbarkeit und Verständlichkeit zu fördern, verschiedene, den Textsatz betreffende Konventionen befolgt: zur besseren Unterscheidung werden die Eigenschaften als auch die Basiseigenschaften aus dem SPARDAT-Qualitätsmodell in `courier` und die Merkmale in *kursiv* dargestellt.

4.2 Identifizierung der Auswertungsmethoden

Jedes Merkmal einer Basiseigenschaft wird durch einfach zu bestimmende und zu messende Kenngrößen bewertet. Kenngrößen sind ein wichtiges Hilfsmittel um die qualitative Erstellung und Pflege von Prozessen transparent zu machen. Durch Kenngrößen wird der Prozeß und das daraus resultierende Produkt hinsichtlich der Qualität beurteilt.

Für manche Merkmale ist es möglich, Kenngrößen in Form von bekannten Maßen anzugeben (z.B. das Maß von McCabe für die Auswertung der Komplexität eines Moduls, vgl. [Wal90]). Für solche Kenngrößen werden Berechnungsformeln angegeben. Darüberhinaus wird angegeben, welche Dimension die

Kenngröße besitzt und auf welcher Art von Skala (Intervall, Absolut etc.) die Meßwerte anzugeben sind (vgl. [Wal90]). Weiterhin wird eine Interpretation oder ein Beurteilungsmaßstab für die Meßwerte angegeben, d.h. es wird festgelegt, was ein qualitativ guter und was ein qualitativ schlechter Meßwert ist.

Bei anderen Merkmalen sind nur Kenngrößen möglich, die sich aufgrund von Erfahrungswerten mit Prüflisten auswerten lassen. *Prüflisten* beinhalten Fragen, welche die Bewertung von Qualitätsmerkmalen ermöglichen. Die Aussagekraft von Prüflisten hängt mit den Wertebereichen der Antworten, sowie den Auswertungsmöglichkeiten der Fragen und Ergebnisse zusammen. Es können beispielsweise für die Auswertung der Fragen aus einer Prüfliste die Ja-/Nein-Antworten mit Punkten gewichtet und anschließend die Punktesumme berechnet werden.

Die vollständige Beschreibung der im vorigen Abschnitt identifizierten Qualitäts-Basiseigenschaften mit deren Merkmalen und Auswertungsmethoden, so wie sie in den Kapiteln 5 bis 7 angewendet werden, befindet sich im Kapitel E1.

Es folgt nun eine Diskussion, welche die Auswahl der im Kapitel E1 aufgezählten Prüffragen und Bewertungsmaße begündet. Es werden dabei nur diejenigen Prüffragen und Bewertungsmaße von Merkmalen in Betracht gezogen, die nicht selbstverständlich und somit erläuterungsbedürftig sind.

Gebrauchstauglichkeit

Im Rahmen der Eigenschaft `Gebrauchstauglichkeit` werden die `Funktionstüchtigkeit`, die `Betriebstüchtigkeit` und die `Benutzungsfreundlichkeit` analysiert (vgl. auch Abbildung 14). Die `Funktionsabdeckung`, als Basiseigenschaft der `Funktionstüchtigkeit`, prüft die Vollständigkeit der Funktionen eines Produktes in bezug auf die funktionalen Anforderungen. Die Merkmale der `Funktionsabdeckung` sind die *funktionale Vollständigkeit* und die *Vollständigkeit der Dokumentation*. Die Auswertungsmethoden für die Bewertung dieser Merkmale beziehen sich in dieser Arbeit auf die Funktionalität der API-Funktionen, die DHE zur Verfügung stellt und auf die DHE-Dokumentation. Durch die Auswertung von Prüflisten, wie beispielsweise „Stimmt der Software-Entwurf mit den Anforderungen überein?“, soll festgestellt werden, ob die vorgegebenen Anforderungen des OERR-Systems mit den von DHE zur Verfügung gestellten Funktionen erfüllt werden können.

Im Rahmen der Eigenschaft `Betriebstüchtigkeit` werden folgende Basiseigenschaften analysiert: `Korrektheit`, `Zuverlässigkeit`, `Sicherheit`, `Robustheit`, `Effizienz` und `Verknüpfbarkeit`. Während bei der Bewertung der `Korrektheit` laut [Wal90] von der folgenden Frage ausgegangen wird: „Erfüllt das Produkt das, was von ihm erwartet wird?“, gibt die `Zuverlässigkeit` Aussagen darüber, wie gut das Produkt seine Anforderungen über einen bestimmten Zeitraum erfüllt. Die `Zuverlässigkeit` bezieht sich also auf den Aspekt des fehlerfreien Funktionierens eines Software-Produktes über einen definierten Beobachtungszeitraum. Als Bewertungsmaß wird der *Einsatz des Software-Produktes* definiert, welcher die Anzahl der Fehler pro 1000 LOC (*lines of code*), die in einem bestimmten Zeitraum (z.B. einen Monat) nach Freigabe des Produktes auftreten, berechnet. Erfahrungswerte nach [Wal90] liegen bei ein bis drei Fehler pro 1000 LOC. Wichtig für die Bewertung der `Zuverlässigkeit` ist auch die *Verfügbarkeit des Systems* als Wahrscheinlichkeit, daß das System eine vorgegebene Zeit stabil läuft.

Die Bewertung der `Sicherheit` eines Systems soll Aussagen darüber erlauben, ob der unberechtigte Zugriff auf Programme und Daten eines Produktes kontrolliert werden kann. Die Auswertungsmethoden, die in Form von Prüflisten ermittelt werden, beschreiben die Instrumentierung des Produktes, mit der unerlaubte Zugriffe auf das System verhindert werden können. Weiterhin wird durch Prüflisten die Einhaltung von Sicherungs- und Datenschutzmaßnahmen ermittelt. So ist beispielsweise durch eine positive Antwort der Frage „Gibt es eine Notstromversorgung?“ nachvollziehbar, daß im Falle einer Stromunterbrechung das System weiter betriebsfähig bleibt.

Die `Robustheit` ist die Eigenschaft eines Systems, auch bei Verletzung der spezifizierten Betriebs- und Benutzungsvoraussetzungen seine erwartungsgemäße Funktionsfähigkeit zu erhalten. Es werden hierzu Fehlerbehandlungsprüflisten aufgestellt und ausgewertet.

Die `Effizienz` ist eine weitere wichtige Basiseigenschaft, die die folgende Fragestellung beantwortet: Läuft das Produkt ressourcenschonend auf der Hardware? Für Messungen bietet sich das Zeitverhalten (z.B. Antwortzeiten) und die Inanspruchnahme von Ressourcen, wie beispielsweise Speicher, an. *Ausführungseffizienz* und *Speichereffizienz* sind konkurrierende Merkmale (vgl. [Wal90]). Eine Optimierung einer der beiden Merkmale zieht i.a. eine Verschlechterung des anderen nach sich. Es ist auch sinnvoll, zwischen der Effizienz des gewählten Algorithmus und der Effizienz der Implementation zu unterscheiden. Bei ersterer wird versucht, mit Hilfe einer Algorithmenanalyse und den Mitteln der Komplexitätstheorie die Effizienz des gewählten Algorithmus zu bewerten. Bei letzterer wird das Verhalten des Programms im vorgegebenen Hard- und Software-System untersucht. In dieser Arbeit wird nur die Effizienz der Implementation betrachtet.

Die `Verknüpfbarkeit` gibt Aussagen über den Aufwand, zwei Produkte miteinander zu verbinden. Dieser Eigenschaft kommt mehr und mehr Bedeutung zu, da immer häufiger Systeme mit anderen durch lokale oder globale Kommunikationsnetzwerke in Verbindung stehen. In dieser Arbeit wird die `Verknüpfbarkeit` aus interner und externer Sicht betrachtet: in der internen Sicht wird die Kommunikation zwischen den zwei Teilanwendungen von OERR (OE und RR) innerhalb von DHE ausgewertet. In der externen Sicht ist der Datenaustausch zwischen einem bereits existierenden System, das nicht auf DHE basiert (z.B. das proCOM-System der ROKD GmbH, vgl. [ROKD96]) und der unter DHE entwickelten OERR-Anwendung zu validieren. Dadurch soll festgelegt werden, in welchem Fall der Aufwand größer ist: in DHE neue Anwendungen zu entwickeln oder eine bereits existierende Anwendung mit DHE zu verknüpfen.

Einen weiteren Diskussionspunkt stellt die Qualitätseigenschaft `Benutzungsfreundlichkeit` dar. Die `Benutzungsfreundlichkeit` ist eine der wichtigsten Basiseigenschaften überhaupt: wenn ein Software-Produkt nur schlecht durch den Endbenutzer bedienbar ist, so kann die Bewertung der gesamten Qualität nicht gute Ergebnisse erzielen, auch wenn die anderen Qualitätseigenschaften des Produktes angemessen sind.

Die im Rahmen der `Benutzungsfreundlichkeit` festgelegten Basiseigenschaften `Erlernbarkeit`, `Handhabbarkeit`, `Effektivität` und `Einheitlichkeit` werden konkret in den Kapiteln 5 bis 7 aus zwei Sichten betrachtet: aus der Sicht des Entwicklers des OERR-Systems mit DHE wird die `Benutzungsfreundlichkeit` der DHE-Software validiert. Diese Validierung soll Aussagen erlauben, ob beispielsweise eine rasche Aneignung des Umgangs mit DHE für die Entwicklung von DHE-basierten Anwendungen möglich ist, um dadurch eine Kostenreduzierung in der Entwicklung zu erzielen. Die `Benutzungsfreundlichkeit` wird aber auch aus der Perspektive des zukünftigen Benutzers des OERR-Systems (das Krankenhauspersonal) betrachtet. Aus dieser Sicht wird validiert, inwieweit mit DHE benutzungsfreundliche Anwendungen entwickelt werden können.

Pflegbarkeit

Im Rahmen der zweiten Kategorie von Eigenschaften, der `Pflegbarkeit` eines Systems (vgl. Abbildung 14), werden die `Wartbarkeit` und die `Prüfbarkeit` analysiert. Die Auswertung der `Wartbarkeit` eines Produktes bezieht sich auf den Aufwand, einen Fehler zu lokalisieren und zu beheben (vgl. [Wal90]). Dies ist eine besonders wichtige Qualitätseigenschaft aus der Sicht eines ökonomischen Produktbetriebes, da die dominierenden Kosten im Life-Cycle eines Produktes meist die Betriebs- und Wartungskosten sind. Ein gut strukturierter und ausreichend geplanter Entwicklungsprozeß ist eine Voraussetzung für gute Wartbarkeit. Anwendbare Auswertungsmethoden arbeiten mit Prüflisten, welche die Struktur und Minimalität der Entwürfe und der Implementierung und die Einfachheit des Codes abfragen.

Die Auswertung der `Prüfbarkeit` eines Produktes bezieht sich auf den Aufwand für das Testen eines Programms, um sicherzustellen, daß es die spezifizierten Anforderungen erfüllt.

Anpaßbarkeit

Die dritte und letzte Kategorie betrachteter Eigenschaften ist die `Anpaßbarkeit` (vgl. Abbildung 14). Sie enthält die Basiseigenschaften `Portabilität` und `Wiederverwendbarkeit`. Die Auswertung der `Portabilität` erlaubt Aussagen über die Möglichkeit des Betreibens des Produktes in einer anderen Hardware- oder Betriebssystemumgebung. Auswertungsmethoden beziehen sich auf die Unabhängigkeit von Maschinen- und Betriebssystemdetails. Die Auswertung der `Portabilität` bezieht sich in dieser Arbeit auf die Portierfähigkeit einerseits der DHE-Software auf verschiedenen Maschinen und Betriebssysteme und andererseits des OERR-Systems in verschiedene Umgebungen (*Beispiel: Ist das entwickelte OERR-System ohne viel Aufwand von MS Windows 3.1 auf MS Windows 95 zu portieren?*).

Die Auswertung der `Wiederverwendbarkeit` eines Produktes erlaubt Aussagen darüber, ob Teile des Produktes für andere Anwendungen eingesetzt werden können. Der Unterschied zu der `Portabilität` besteht darin, daß Teile des Codes in anderen Anwendungssystemen unter derselben Betriebsumgebung verwendet werden. Auswertungsmethoden dieser Eigenschaft beziehen sich auf die

Zugänglichkeit und Güte der Dokumentation und die Software-Systemunabhängigkeit. In dieser Arbeit bezieht sich die `Wiederverwendbarkeit` auf die Übernahme verschiedener Teile des OERR-Systems in anderen Anwendungen die mit DHE entwickelt werden (*Beispiel: Das OERR-System besitzt ein Rollensystem: kann dieser Teil in anderen DHE-Anwendungen, wie beispielsweise einem Pflegesystem, wiederverwendet werden?*).

Ein letzter Diskussionspunkt ist die Kosten-Nutzen-Betrachtung des zu konstruierenden OERR-Systems mit Hilfe von DHE. Dies ist keine explizite Qualitätseigenschaft im SPARDAT-Modell, sondern stellt eher einen Nachweis der Wirtschaftlichkeit des Entwicklungsprozesses und des entwickelten Systems dar.

Es wird zwischen zwei Kostenarten unterschieden: die Abschätzung der Software-Entwicklungskosten des OERR-Systems und die danach resultierenden Wartungskosten. Die Software-Entwicklungskosten können mit Hilfe des COCOMO (COnstructive COst MOdel)-Modells in Form einer Entwicklungszeitschätzung in Mann-Monaten (die Projektwartung ausgenommen) abgeschätzt werden (vgl. [Dum92]). Die Schätzformel ist dabei den verschiedenen Klassen von Software-Projekten entsprechend ausgerichtet: es werden je nach der Entwicklungskomplexität bzw. -schwierigkeit einfache, mittlere und komplexe Projekte unterschieden. Wegen des Vorteils der Programmiersprache Visual Basic, keinen Programmcode für die Konstruktion der Benutzungsoberfläche (Masken, Dialogboxen etc.) programmieren zu müssen, aus dem sich eine relativ geringe Anzahl an Quellcodezeilen für die Entwicklung des OERR-Systems ergibt, wird für diese Anwendung die mittlere Entwicklungsklasse angenommen. Ausgangswert ist die Abschätzung der Codezeilen insgesamt. Weitere Einflußfaktoren sind Produktattribute (z.B. Größe der Datenbank), Rechnereigenschaften (z.B. Speicherkapazität), Erfahrungen des Entwicklers (z.B. Erfahrungen mit der eingesetzten Programmiersprache) und Projekteigenschaften (z.B. Anwendung von Software-Tools). Die Auswahl genau dieser Attribute ist eine Spezifik des Boehmschen COCOMO-Modells (vgl. [Dum92]).

Da das COCOMO-Modell die Wartungskosten nicht berücksichtigt, werden diese der Vollständigkeit halber im folgenden kurz erwähnt. Eine Grobschätzung der Kosten der Wartungsanforderung unter Berücksichtigung des Zustandes und der Wartbarkeit des Produktes wird in [Wal90] aus folgenden Elementen berechnet:

- Kosten der Lokalisierung eines Fehlers,
- Kosten der Aktualisierung der Dokumentation,
- Kosten der Änderungen des Codes und der Datenbank,
- Kosten der Software-Prüfung (Testen, Review),
- Kosten der Installation der geprüften Module in der Produktionsumgebung.

Der tatsächliche Aufwand wird in einem Wartungsbericht dokumentiert.

Im laufe der Arbeit wird sich jedoch herausstellen, daß sowohl das COCOMO-Modell als auch der Wartungsbericht und die Wartungskosten für das betrachtete OERR-System nicht von Relevanz sind.

Es stellt sich nun die Frage, welche der beschriebenen Basiseigenschaften in welchen Entwicklungsphasen des OERR-Systems bewertet werden können. So kann beispielsweise die `Zuverlässigkeit` des entwickelten Produktes erst in der Inbetriebnahme bewertet werden. Die funktionale `Widerspruchsfreiheit` dagegen kann schon bei der Anforderungsspezifikation validiert werden. Die Frage wird im nachfolgenden Kapitel beantwortet: in einem Validierungsplan wird festgelegt, wie die konstruktiven und analytischen Qualitätssicherungsmaßnahmen zusammenarbeiten.

4.3 Der Validierungsplan

In einem Validierungsplan werden für jede Software-Entwicklungsphase des V-Modells (vgl. Kapitel 2.2.1) die Tätigkeiten und ihre Ergebnisse festgelegt. Dabei handelt es sich sowohl um Entwicklungstätigkeiten, wie beispielsweise das Aufstellen eines Phasen-Enddokumentes, als auch um Validierungs- und Verifikationstätigkeiten.

Für jede Phase werden eine kurze Beschreibung, sowie die In- und Output-Parameter der Phase angegeben. Es folgt, als Tätigkeit der Entwicklungsphase, die Aufstellung der Phasen-Enddokumente. Auf der Basis dieser Dokumente können die Validierungstätigkeiten beginnen. Als erstes werden die Qualitätsanforderungen an den Prozeß und an die Phasenprodukte definiert. Als Hilfsmittel dazu werden für die betreffende Phase typische Fehler aufgelistet, aus denen die Qualitätseigenschaften des Prozesses und der Phasen-Endprodukte hergeleitet werden können. Diese werden in einem Inhaltsverzeichnis strukturiert. Beispielsweise sind für eine Anforderungsanalyse Merkmale wie *Vollständigkeit* oder *Widerspruchsfreiheit* zu spezifizieren. Alle diese Validierungsziele sind in einer quantifizierbaren und nachprüfbaren Form zu definieren. Dies bedeutet, daß im Validierungsplan auch Methoden und Hilfsmittel anzugeben sind, um die Prüfung der spezifizierten Kriterien sicherzustellen. Anhand der aufgestellten Phasen-Enddokumente werden die Qualitätseigenschaften mittels der festgelegten Auswertungsmethoden bewertet. Es folgt die Auswertung der Ergebnisse der durchgeführten Validierung sowie Angaben über die Durchführungszeit der Phase.

Es sollte noch hinzugefügt werden, daß sich die im Kapitel E1 aufgelisteten Qualitätseigenschaften und Qualitätsmerkmale auf den gesamten Entwicklungsprozeß beziehen. Die Fragen in den Prüflisten beispielsweise sind unabhängig von der Entwicklungsphasen, in denen diese bewertet werden, aufgelistet. Folgendes Beispiel soll eine solche Situation besser veranschaulichen: für das OERR-System ist beispielsweise die Qualitätseigenschaft `Funktionale Widerspruchsfreiheit` von Bedeutung (vgl. Kapitel E1). Diese Eigenschaft besitzt die Merkmale *Widerspruch zwischen Funktionen* und *Widerspruch zwischen dynamischem Produktverhalten und Dokumentation*. Die Auswertungsmethode für das erste Merkmal ist eine Prüfliste, die folgende zwei Fragen enthält:

F_1: Gibt es Widersprüche zwischen Funktionen auf der Ebene der Spezifikation?

F_2: Gibt es Widersprüche zwischen Funktionen auf der Ebene der Implementierung?

Aus den obigen Fragen kann festgestellt werden, daß das Merkmal *Widerspruch zwischen Funktionen* in zwei verschiedenen Entwicklungsphasen bewertet werden muß. D.h., daß die Frage F_1 in der Spezifikationsphase, während F_2 erst in der Implementierungsphase beantwortet werden kann. Eine Beurteilung dieses Merkmals ist demnach in der Spezifikationsphase nicht möglich. Deshalb werden in den Kapiteln 5 bis 7 die Merkmale bzw. Auswertungsmethoden nach den Entwicklungsphasen aufgesplittet. Die Validierung wird so durchgeführt, daß nur diejenigen Fragen gestellt und bewertet werden, die für die Phase spezifisch sind. Die Antworten werden dann gemäß dem vorgestellten SPARDAT-Modell aus Kapitel E1 erst im Kapitel 8 wieder zusammengeführt und eine entsprechende Interpretation der Qualitätseigenschaft bzw. -merkmal als Ganzes vorgenommen.

Der Inhalt eines Validierungsplans für die Entwicklungsphasen Spezifikation, Entwurf, Implementierung und Test gliedert sich für jede dieser Phasen wie folgt:

1. Kurze Beschreibung der Entwicklungsphase.
2. Input (*z.B. die Ist-Analyse für die Phase Spezifikation*) und Output (*z.B. das Anwendungsfallmodell für die Phase Spezifikation*) der durchgeführten Entwicklungsphase.
3. Durchführung der Phase mit Aufstellen der Phasen-Enddokumente.
4. Validierung
    - 4.1 Inhaltsverzeichniss der Validierungsziele für die entsprechende Phase.
    - 4.2 Inhaltsverzeichniss der Auswertungsmethoden für die entsprechende Phase.
    - 4.3 Durchführung der Validierung.
    - 4.4 Auswertung der Ergebnisse.
5. Dauer der Tätigkeiten der Phase.

Der hier beschriebene Format eines Validierungsplans wird in den Kapiteln 5 bis 7 für jede Entwicklungsphase angewendet.

4.4 Festlegung charakteristischer Vorgänge anhand der Validierungsziele

Wie im Kapitel 1 festgestellt wurde, ist das OERR-System eine repräsentative Komponente eines KIS. Um festlegen zu können, inwieweit die Anforderungen von KIS-Anwendungen mit DHE erfüllt werden können, wird in diesem Kapitel das OERR-System repräsentativ gestaltet: es werden charakteristische Vorgänge festgelegt, deren Eigenschaften und Merkmale auf das gesamte KIS übertragbar sind. Die Auswahl dieser Vorgänge hängt von den im Kapitel 4.1 festgelegten Qualitätsmaßstäben ab. Wenn es möglich ist, diese Vorgänge zu implementieren und die Auswertung der Qualitätsmaßstäbe positive Ergebnisse liefert, kann DHE für die Entwicklung eines gesamten KIS angewendet werden und somit eine Standard-Architektur implementieren.

Im Kapitel 4.4.1 wird der Unterschied zwischen Produkt- und Prozeßspezifika hinsichtlich der Festlegung charakteristischer Vorgänge erläutert. Es folgt im Kapitel 4.4.2 die Identifizierung sowie eine kurze Beschreibung der ausgewählten OERR-Vorgänge. Kapitel 4.4.3 faßt die identifizierten Vorgänge zusammen.

4.4.1 Produkt-und Prozeßspezifika

Grundsätzlich wird zwischen der Qualität des Produktes (in diesem Fall das OERR-System) und der Qualität des Entwicklungsprozesses unterschieden. Die Qualitätsziele für das Software-Produkt bestimmen die Qualitätsziele für den Entwicklungsprozeß (vgl. auch [Wal90]). Aus der Bewertung dieser Qualitätsziele läßt sich die Qualität des Entwicklungsprozesses ableiten. Diese hat wiederum entscheidenden Einfluß auf die Qualität des Produktes (OERR). Dieses Beziehungsgefüge, welches mit der Bestimmung der Qualitätsziele des Software-Produktes beginnt, wird bei konkreten Entwicklungsprojekten mehrfach durchlaufen (vgl. Abbildung 15).

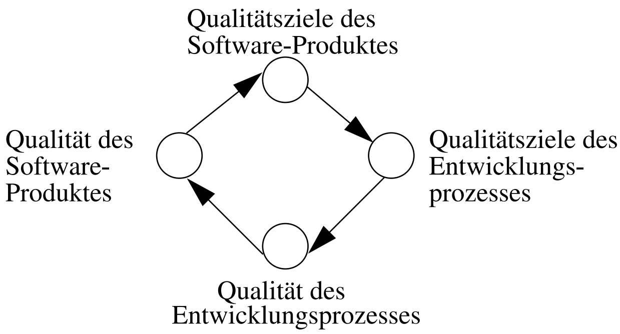

Abbildung 15: Qualitativer Zusammenhang von Entwicklungsprozeß und Software-Produkt (vgl. [Wal90])

Der Entwicklungsprozeß kann durch Phasen strukturiert werden. Die Qualität sollte in jeder Phase entstehen. Wichtig ist, daß an jede Phase Anforderungen (bezogen auf die Phasenergebnisse, aber auch auf die Aktivitäten) gestellt werden und diese am Ende der Phase auf Erfüllung hin überprüft werden. Die Gesamtqualität des OERR-Systems setzt sich nun stufenweise aus der Qualität der Phasenergebnisse und der Erfüllung der Prozeßanforderungen zusammen. Der Unterschied zwischen Produkt- und Prozeßqualitätsmaßstäbe wird anhand folgender Beispiele verdeutlicht.

Die `Wiederverwendbarkeit` (vgl. Kapitel 4.1) ist eine produktspezifische Qualitätseigenschaft. Die Umsetzung dieser Eigenschaft wird beispielsweise durch die Implementierung des Rollensystems, als Teil des OERR-Systems konkretisiert.

Ein weiteres Beispiel ist die Kostenabschätzung für die Entwicklung des OERR-Systems. Es wurde im Kapitel 4.2 darauf hingewiesen, daß eine Kostenreduktion in der Entwicklung einer auf DHE-basierten Anwendung angestrebt wird. Eine Kostenabschätzung ergibt sich aus der Zeitberechnung der Durchführung von Entwicklungsphasen. Die Durchführungszeit einer Phase hängt jedoch von der Anzahl der zu implementierenden Vorgänge ab. Es ist unrealistisch, nur anhand eines Vorganges die Entwicklungskosten abzuschätzen. Es sollten demnach mindestens drei OERR-Vorgänge verschiedener Komplexität und Schwierigkeit implementiert werden.

4.4.2 Identifizierung charakteristischer OERR-Vorgänge

Aus Kapitel 3.2 resultiert, daß sich das OERR-System von der Generierung der Leistungen auf der anfordernden KE bis hin zu deren Durchführung und Dokumentation der Ergebnisse auf der leistenden KE erstreckt. Im folgenden werden anhand der im Kapitel 4.1 festgelegten Validierungsziele die charakteristischen OERR-Vorgänge identifiziert. Dabei wird folgende Vorgehensweise angewendet:

Es werden zwei unterschiedliche Ebenen betrachtet: E_1 und E_2. Auf der Ebene E_1 wird der von der ROKD GmbH vorgegebene Qualitätsmaßstab und das angestrebte Auswertungsergebnis für die DHE-Software beschrieben. Auf der zweiten Ebene (E_2) wird festgelegt, wie das in E_1 festgelegte Ziel mittels eines OERR-Vorganges umgesetzt werden kann, um anschließend das Erfolgsfaktor validieren zu können. Nach der Auswertung des Maßstabes in E_2 wird das Ergebnis mit dem angestrebten Resultat aus E_1 verglichen.

Folgende *produktspezifische* Qualitätsmaßstäbe unterstützen die Identifizierung von OERR-Vorgängen:

Wiederverwendbarkeit

Ebene E_1: Beschreibung des Qualitätsmaßstabes

Wenn eine KIS-Anwendung, oder Teile davon, in anderen DHE-basierten KIS-Anwendungen wiederverwendet werden können, so könnte DHE in Bezug auf die `Wiederverwendbarkeit` als angemessen betrachtet werden.

Ebene E_2: Umsetzung des Qualitätsmaßstabes in OERR

Es soll im OERR-System ein Vorgang gefunden werden, dessen Eigenschaften und Merkmale auch in anderen Anwendungen eines KIS wiederverwendet werden können.

Die meisten KIS-Anwendungen erfordern eine differenzierte Arbeitsweise aus der Sicht der Benutzer. So kann beispielsweise das Pflegepersonal nicht dieselben Tätigkeiten in einem Krankenhaus durchführen wie das medizinische Personal. Diese reale Situation spiegelt sich auch in dem zu konstruierenden OERR-System wider: es gibt Funktionen im System, die nur von dem medizinischen Personal ausgeführt werden dürfen. Demzufolge erfordern die meisten KIS-Anwendungen die Entwicklung eines Rollensystems, das die unterschiedlichen Authorisierungsprofile der Benutzer unterstützen soll. Falls ein Rollensystem im OERR mit Hilfe von DHE implementiert werden kann, so wird davon ausgegangen, daß dies auch in weiteren DHE-basierten KIS-Anwendungen wiederverwendbar sein könnte.

Für die Implementierung des Rollensystems in OERR, mit dem Zweck der Wiederverwendung, wird der Vorgang `Allgemeine Sitzung` definiert. Dieser Vorgang beschreibt die Tätigkeiten, die jeder Benutzer durchzuführen hat, um mit dem entwickelten System arbeiten zu können. Es handelt sich dabei um das Einloggen in das System mittels eines im Authorisierungsprofil festgelegten Namens und eines Paßwortes, das Aktivieren einer oder mehrerer der gewünschten Funktionen, für die der Benutzer einen authorisierten Zugriff hat, und das Ausloggen aus dem System.

Sicherheit

Ebene E_1: Beschreibung des Qualitätsmaßstabes

Die Bewertung der `Sicherheit` einer Anwendung soll Aussagen darüber erlauben, ob der unberechtigte Zugriff auf Programme und Daten kontrolliert und verhindert werden kann.

Ebene E_2: Umsetzung des Qualitätsmaßstabes in OERR

Der OERR-Vorgang, der für die Validierung der Qualitätseigenschaft `Sicherheit` gesucht wird, ist ebenfalls ein Teil des Rollensystems. Es soll vom System eine Fehlermeldung ausgegeben werden, falls ein Benutzer auf Daten oder Funktionen zugreifen will, für die er keine Zugriffsberechtigung hat. Ein Beispiel für einen solchen Vorgang im OERR wäre die Implementierung eines Leistungs-Verifikationsmechanismus: Leistungen werden von der Stationsschwester generiert. Sie müssen vor ihrer Versendung zu der leistenden KE von einem Stationsarzt verifiziert werden. Der Stationsarzt muß über die entsprechenden Verifikationsrechte verfügen. Ein Zugriff des Pflegepersonals auf solche Funktionen, wie beispielsweise *Verifizieren einer Leistung,* soll vom System verhindert werden. Dieser Vorgang der Leistungsverifikations wird als `Anforderung verifizieren` bezeichnet.

Ein ähnlicher Vorgang ist `Ergebnisse bestätigen`, welches ebenfalls auf der Leistungsverifikation basiert. Jedes Ergebnis einer Leistung, das von der leistenden KE zu der anfordernden KE versendet wurde, ist von einem Stationsarzt der anfordernden KE zu bestätigen. Der Stationsarzt kann ein eingetroffenes Ergebnis kommentieren oder aber, falls das Ergebnis nicht ausreichend dokumentiert wurde, Nachfragen an die leistende KE zurücksenden.

Verknüpfbarkeit

Ebene E_1: Beschreibung des Qualitätsmaßstabes

Die Bewertung der `Verknüpfbarkeit` gibt Aussagen über den Aufwand, zwei KIS-Anwendungen miteinander zu verbinden.

Ebene E_2: Umsetzung des Qualitätsmaßstabes in OERR

Wie im Kapitel 4.2 erwähnt wurde, basiert die Qualitätseigenschaft `Verknüpfbarkeit` auf der Fähigkeit einer Anwendung, mit einer anderen, durch globale oder lokale Kommunikationsnetzwerke in Verbindung zu treten. Diese wesentliche EDV-technische Anforderung an ein KIS wurde im Kapitel 3.1 als physikalische Integration bezeichnet. Durch eine physikalische Integration ist ein effizienter Datenaustausch zwischen Anwendungen möglich.

Im Hinblick auf OERR ist die `Verknüpfbarkeit` aus zwei Sichten zu betrachten: zum einen wird die Kommunikation der zwei Teilanwendungen von OERR (OE und RR) innerhalb von DHE betrachtet. Es soll dadurch festgestellt werden, ob eine korrekte und stabile Kommunikation der Anwendungen innerhalb von DHE möglich ist. Für die Validierung dieses Ziels werden folgende OERR-Vorgänge implementiert: `Anforderung versenden` und `Ergebnisse berichten`.

Der Vorgang `Anforderung verifizieren` ist ein Teilvorgang des Vorganges `Anforderung versenden` (siehe weiter oben). Eine Anforderung wird von einem Stationsarzt der leistenden KE versendet, nachdem sie verifiziert wurde. Die Kommunikation zwischen der anfordernden und der leistenden KE sollte innerhalb von DHE über das TCP/IP-Protokoll erfolgen (vgl. Abbildung 12). Der Vorgang `Ergebnisse berichten` beinhaltet das Zurücksenden von Ergebnissen einer durchgeführten Leistung von der leistenden an die anfordernde KE.

Die zweite Sicht, aus der die `Verknüpfbarkeit` betrachtet wird, ist der Datenaustausch zwischen dem schon bestehenden proCOM-Stationssystem der Firma ROKD, das nicht auf DHE basiert und der DHE-basierten OERR-Anwendung. Das proCOM-Stationssystem leistet u.a. alle Arten von Aufnahme-, Verlegung- und Entlassungsvorgängen auf der Station (vgl. auch [ROKD96]). Bei der Generierung einer Anforderung auf der anfordernden KE werden u.a. auch Patientenstammdaten (Name, Vorname, Geburtsdatum, Aufenthalts-ID etc.) benötigt. Wenn die OERR-Anwendung mit dem schon vorhandenen proCOM-Stationssystem verbunden werden könnte, so wäre es möglich, diese Daten in OERR aus proCOM zu laden.

Für die Validierung dieses Ziels wird in OERR der Vorgang `Anforderung generieren` implementiert. Dieser Vorgang beschreibt die Tätigkeiten, die bei der Generierung einer Leistung auf der anfordernden KE anfallen. Diese Tätigkeiten werden von einer Stationsschwester durchgeführt. Da die Generierung einer Anforderung patientenbezogen ist, werden persönliche Patientendaten benötigt, die aus dem proCOM-Stationssystem geladen werden könnten.

Kostenreduktion

Ebene E_1: Beschreibung des Qualitätsmaßstabes

Es wird eine Kostenreduktion in der Entwicklung von DHE-basierten Anwendungen gegenüber herkömmliche Anwendungen angestrebt.

Ebene E_2: Umsetzung des Qualitätsmaßstabes in OERR

Für eine Abschätzung der Entwicklungskosten einer auf DHE-basierten Anwendung wird die Zeit für die Durchführung jeder Entwicklungsphase gemessen und mit der Entwicklung von Anwendungen in konventionellen KIS-Architekturen verglichen. Für eine reale Abschätzung sollten alle oben beschriebenen Vorgänge entwickelt werden.

Die hier aufgezählten produktspezifischen OERR-Vorgänge werden *phasenspezifisch* noch auf folgende Eigenschaften überprüft: `Funktionstüchtigkeit`, `Betriebstüchtigkeit`, `Benutzungsfreundlichkeit` und `Pflegbarkeit`. Für die Auswertung dieser Eigenschaften werden keine expliziten Vorgänge ausgewählt, sondern die schon vorhandenen benutzt. So könnte beispielsweise die

`Funktionstüchtigkeit` dadurch ausgewertet werden, ob mit den von DHE zur Verfügung gestellten API-Funktionen die erwähnten OERR-Vorgänge vollständig und korrekt implementiert werden können.

Durch die Entwicklung einer graphischen Benutzungsoberfläche für die Interaktion der OERR-Vorgänge mit dem Benutzer soll die `Benutzungsfreundlichkeit` des Systems erhöht werden. Es ist zu validieren, ob mit den von Visual Basic zur Verfügung gestellten Interaktionsformen, die benötigten Daten aus DHE in der gewünschten Form dargestellt werden können. Darüberhinaus werden Handhabung und Zugriffsmöglichkeiten auf Daten und Operationen aus DHE, Eingaben von Daten mit Visual Basic, sowie die Art und der Ablauf der Interaktion bewertet.

4.4.3 Zusammenfassung

Folgende OERR-Vorgänge werden aufgrund der genannten Kriterien ausgewählt:

- Allgemeine Sitzung
- Anforderung generieren
- Anforderung verifizieren
- Anforderung versenden
- Ergebnisse berichten
- Ergebnisse bestätigen

Diese Vorgänge werden in den Kapiteln 5 bis 7 näher spezifiziert und implementiert. Für die Implementierung der genannten Vorgänge werden darüberhinaus noch folgende Hilfsfunktionen benötigt:

- **Arztablage bearbeiten**

  Die Arztablage dient als Briefkasten für die zu verifizierenden Leistungen. Der Inhalt dieser Ablage wird durch den Stationsarzt bearbeitet.

- **Ergebnisablage bearbeiten**

  Dieser Vorgang ermöglicht die Auswahl eines eingetroffenen Ergebnisses einer durchgeführten Leistung zweck Bestätigung.

- **Formular drucken**

  Jedes bearbeitete Formular kann auf einem Stationsdrucker ausgedruckt werden.

- **Schwesternablage bearbeiten**

  Die Schwesternablage dient als Briefkasten für den Schriftverkehr auf der Station und zwischen den Stationen des Krankenhauses.

Die hier beschriebenen Vorgänge werden nach der OOSE-Methode (vgl. Kapitel 2.2.2) als *Anwendungsfälle* betrachtet. In den Kapiteln 5 bis 7 wird eine exakte und vollständige Beschreibung dieser Anwendungsfälle angegeben.

5 Die Analysephase

In diesem und den nachfolgenden zwei Kapiteln wird, basierend auf dem im Kapitel 2.2 beschriebene Vorgehensmodell, die OOSE-Methode nach [JCJ+93] für die Entwicklung des OERR-Systems angewendet. Die charakteristischen Vorgänge des Systems wurden im Kapitel 4.4.3 festgelegt. Es wird weiterhin der Validierungsplan aus Kapitel 4.3 für jede Entwicklungsphase durchgeführt.

Diese Kapiteln gliedern sich nach der Einteilung des Entwicklungsprozesses durch OOSE: in diesem Kapitel werden die Tätigkeiten, die im Validierungsplan aufgezählt wurden, für die Phase der Analyse durchgeführt. Die Analyse entspricht den Phasen „Systemdurchführbarkeitskonzept“, „Anforderungsdefinition“ und „Systemspezifikation“ aus dem V-Modell (vgl. Abbildung 3). In jedem Abschnitt dieses Kapitels (5.1 bis 5.5) wird jeweils eine Tätigkeit aus dem Validierungsplan beschrieben und durchgeführt. Es folgen in den Kapiteln 6 und 7 die Beschreibung der Durchführung der Tätigkeiten der Konstruktions- und der Testphase (vgl. Abbildung 4). Die Konstruktion entspricht den Vorgehensmodellphasen „Komponenten- und Modulentwurf“, sowie „Code“ (vgl. Abbildung 3). Der Test bezieht sich auf die V-Modell-Phasen „Einzel-, Integrations-, Akzeptanz-/Systemtest“. Dem Test schließen sich die Phasen Pilotbetrieb/Einführung und Betrieb an.

Alle Entwicklungsphasen werden beispielhaft anhand des Anwendungsfalls `Anforderung generieren` beschrieben. Die Beschreibung aller Anwendungsfälle befindet sich im Teil II dieser Arbeit („Entwicklungsdokument“).

5.1 Beschreibung der Entwicklungsphase *Analyse*

Ziel der Analysephase der OOSE-Methode ist das Analysieren der System-Anforderungen, sowie das Spezifizieren und Definieren des zu entwickelnden Systems (vgl. [JCJ+93], S.149). Die Modelle, die während der Analysephase entwickelt werden, sind anwenderorientiert und berücksichtigen nicht die reale Implementationsumgebung, in der das System zu realisieren ist (wie beispielsweise die Programmiersprache, DBMS, Hardware-Konfiguration etc.). Dadurch, daß in dieser Phase noch keine konkrete Begriffe bezogen auf die Implementierung vorkommen, können die von den Entwicklern aufgestellten Modelle einfacher von dem Auftraggeber verstanden werden.

Wie im Kapitel 2.2.2 erwähnt wurde, besteht die Analysephase aus zwei Teilphasen, der *Anforderungs-* und der *Robustheitsanalyse*. Die aus diesen Phasen resultierenden Modelle sind das *Anforderungs-* und das *Analysemodell*. (vgl. Abbildung 4).

5.2 Input- und Output-Parameter der Entwicklungsphase *Analyse*

Die Input-Parameter für die Durchführung der Anforderungsanalyse des zu entwickelnden OERR-System sind die Ist-Analyse der Projektgruppe *PROMETHEUS* (vgl. [BCF+97]) und das Konzept des proCOM-OERR-Systems (vgl. Kapitel 3.2), sowie die identifizierten charakteristischen OERR-Vorgänge aus Kapitel 4.4. Output der Phase Anforderungsanalyse ist das Anforderungsmodell bestehend aus den Dokumenten *Problembereichsmodell*, *Anwendungsfallmodell* und *Schnittstellenbeschreibung* (vgl. Abbildung 16). Input-Parameter für die Durchführung der Robustheitsanalyse ist das Anforderungsmodell der Anforderungsanalyse. Output dieser Teilphase ist das *Analysemodell*. Die hier erwähnten Modelle wurden im Kapitel 2.2.2 näher erläutert.

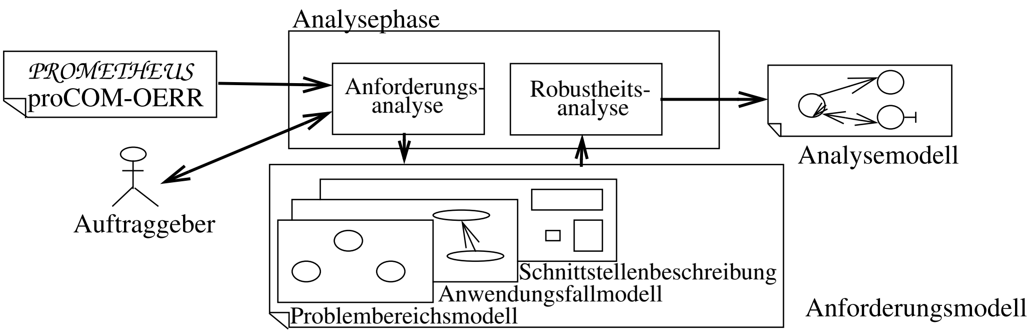

Abbildung 16: Der Analyse-Prozeß (vgl. [JCJ+93], S.149)

5.3 Aufstellung der Phasen-Enddokumente

Die Tätigkeit im Rahmen der Analysephase ist das Aufstellen der Phasen-Enddokumente. Diese sind das Problembereichsmodell, das Anwendungsfallmodell, die Schnittstellenbeschreibung und das Analysemodell (vgl. Kapitel 2.2.2).

5.3.1 Das Problembereichsmodell

Die Problembereichsobjekte werden nach der folgenden Vorgehensweise identifiziert:

- es wird eine Liste mit fachspezifischen Begriffe mit deren Charakteristika aus der Ist-Analyse der Projektgruppe aufgestellt;
- anschließend wird aus dieser Liste, in Abhängigkeit von den Anforderungen, die das System erfüllen muß, diejenigen Begriffe extrahiert, die für die Entwicklung des OERR-Systems notwendig sind.

Im folgenden wird exemplarisch der Teil des Problembereichsmodells vorgestellt, der zum Anwendungsfall `Anforderung generieren` gehört. Es wird der Name sowie eine kurze Beschreibung für jedes Problembereichsobjekt angegeben. Das vollständige Problembereichsmodell befindet sich im Kapitel E2.1.

Problembereichsmodell des Anwendungsfalles `Anforderung generieren`

Objektname	Objektbeschreibung
Formular	Ein Formular enthält verschiedene anforderungsspezifische Felder, die ausgefüllt werden müssen.
Belegungsplan	Der Belegungsplan ist eine Liste der auf der Station befindlichen Patienten.
Arztablage	Die Arztablage dient der Ablage der generierten, von dem Stationsarzt noch nicht verifizierten Anforderungen.
Einzelleistung	Eine Einzelleistung ist eine nicht weiter zerlegbare Anforderung. *Beispiele: Rhythmus-EKG, Klinisches Labor, Zervikal-Röntgen.*
Anforderung	Die Anforderung ist ein Formular für Anforderungen von externen Untersuchungen an einem Patienten, die außerhalb der Station durchgeführt werden. Jede Anforderung kann eine oder mehrere Einzelleistungen enthalten. Die Anforderung ist ein Oberbegriff für funktional zusammengehörende Einzelleistungen, die sich auf jeweils einen speziellen Funktionsbereich beziehen können. Eine EKG-Anforderung für einen Patienten könnte beispielsweise die Einzelleistungen Rhythmus-EKG und Bett-EKG enthalten.
Anforderungskatalog	Der Anforderungskatalog ist die Zusammenfassung der gesamten Einzelleistungen, die sich auf jeweils einen Funktionsbereich beziehen. *Beispiel: Der EKG-Anforderungskatalog enthält insgesamt folgende Einzelleistungen: Ruhe-EKG, EKG am Bett, Rhythmus-EKG, Belastungs-EKG, Nehb‘sche Ableitung, Extremitäten, Langzeit-EKG, Langzeit-EKG mindestens 18h, Phonokardiogramm, Carotispulskurve, Carotisdruck, Schrittmacherkontrolle, Schrittmacherüberstimulation, Inspiration/Expiration, Atropin/Isoptintest, Rechtsherzkatheter und 24h-RR-Überwachung.*
Patientendaten	Zu den Daten eines Patienten gehören: Patienten-ID, Aufenthalts-ID, Name und Vorname, Geschlecht, Geburtsdatum.
Fortbewegung	Die Fortbewegung gibt Informationen über die Fortbewegung eines Patienten. Es gibt folgende Fortbewegungsarten: zu Fuß, sitzend, liegend.

5.3.2 Das Anwendungsfallmodell

Das Anwendungsfallmodell besteht aus elf Anwendungsfällen und den zwei Akteuren ✍Stationsschwester und ✍Stationsarzt, wobei beide sowohl Akteure der anfordernden als auch der leistenden KE darstellen. Grundsätzlich wird zwischen dem Personal auf der Station und dem in den Funktionsbereichen unterschieden. Diese Unterscheidung ist aber im Rahmen dieser Arbeit weniger von Interesse. Die Anwendungsfälle ergeben sich aus den identifizierten OERR-Vorgängen aus Kapitel 4.4.2 und sind in Abbildung 17 aufgeführt. Das in Abbildung 17 von Jacobsen et al. in [JCJ+93] empfohlene Symbol für die Kennzeichung von Akteuren wird nur in den Diagrammen verwendet. Für eine Erhöhung der Lesbarkeit wird im Text jedoch das Symbol „✍“ für die Kennzeichnung der Akteure benutzt.

Zwischen den Akteuren und den Anwendungsfällen sind in dieser Abbildung gerichtete Kanten eingetragen. Diese Kanten zeigen an, daß zwischen Instanzen der Akteure und Anwendungsfallinstanzen *Stimuli* ausgetauscht werden (zur Erläuterung des Stimulus-Konzeptes vgl. [JCJ+93], S. 213). Diese Beziehungen werden als *communication*-Beziehungen bezeichnet. Eine Kante, die bei einem Akteur beginnt und bei einem Anwendungsfall endet, zeigt an, daß eine Instanz des Akteurs, Stimuli an eine Instanz des Anwendungsfalls senden kann, um dort die Ausführung von Operationen zu veranlassen. Eine Kante, die bei einem Anwendungsfall beginnt und bei einem Akteur endet, zeigt an, daß eine Instanz des Anwendungsfalls Stimuli an eine Instanz des Akteurs senden kann (z.B. zur Ausgabe einer Fehlermeldung).

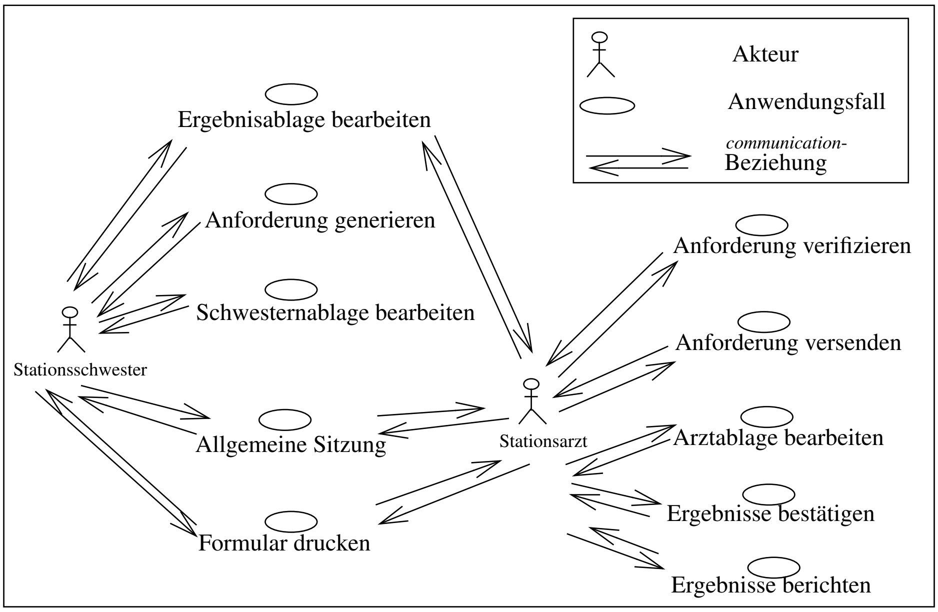

Abbildung 17: Anwendungsfälle und Akteure

Die folgende Beschreibung des Anwendungsfalls `Anforderung generieren` enthält den Namen des Anwendungsfalls, den Ereignisablauf mit Vorbedingung, Nachbedingung und normalem Ablauf, sowie eventuell vorhandene Ausnahmesituationen. Das komplette Anwendungsfallmodell mit den vollständigen Beschreibungen sämtlicher Anwendungsfälle befindet sich im Kapitel E2.2.

Der Anwendungsfall `Anforderung generieren`

Dieser Anwendungsfall findet auf der anfordernden KE statt.

Ereignisablauf

Für die Anforderung von Untersuchungen eines Patienten, die nicht auf der Station durchgeführt werden, müssen von der ✍Stationsschwester entsprechend vorbereitete Formulare ausgefüllt werden, die anschließend von dem ✍Stationsarzt verifiziert und an die leistende Kommunikationseinheit verschickt werden. Das Aussehen dieser Formulare am Bildschirm ist von der Art der Anforderung abhängig und orientiert sich möglichst weitgehend an den Originalformularen im Papierformat.

Vorbedingungen:

Es wurde für einen Patienten eine externe Untersuchung von dem ✍Stationsarzt angeordnet und in die Visiteverordnung eingetragen. Die ✍Stationschwester hat sich gegenüber dem OERR-System auf der anfordernden KE angemeldet. Der Belegungsplan der anfordernden KE wurde geladen.

Nachbedingungen:

Die gewünschte Anforderung wurde generiert und in der Arztablage der anfordernden KE abgelegt.

Ablauf:

Die ✍Stationsschwester wählt einen Patienten aus dem Belegungsplan der Station aus, für den sie eine Anforderung generieren muß. Sie ruft über die Funktionsleiste die Funktion `Leistungsanforderung generieren` auf (vgl. Abbildung 19). Die Formularmaske, welche das entsprechende Anforderungsformular enthält, wird geladen. Die Informationen, die durch die aktuelle Patientenauswahl und die aufgerufene Funktion als Kontext übergeben wurden, werden in den entsprechenden Felder des geöffneten Formulars geladen. Es handelt sich dabei um die Felder „Patient", „Transport", „Kommunikationseinheiten" und „Einzelleistungen" (vgl. Abbildung 20). Das Feld „Kommunikationseinheiten" enthält Informationen über die anfordernde und leistende Kommunikationseinheit. Sollten hier für die ausgewählte Anforderung und die anfordernde KE mehrere leistende KE's, die die Anforderung durchführen können, möglich sein, so werden diese beim Aufruf des Formulars in eine Liste im Feld „Kommunikations-einheiten" geladen. Die ✍Stationsschwester muß eine leistende KE aus dieser Liste auswählen.

Bei der Generierung einer Anforderung können gleichzeitig mehrere Einzelleistungen angefordert werden. Die Auswahl dieser Einzelleistungen wird aus der „Liste der auf der Station häufig benutzten Einzelleistungen" getroffen. Falls sich eine gewünschte Einzelleistung nicht in dieser Liste befindet, so besteht die Möglichkeit, eine Suche im gesamten Anforderungskatalog zu starten.

Für jede ausgewählte Einzelleistung müssen Details, wie z.B. Periodenangabe für die Durchführung der Einzelleistung, klinische Fragestellungen und ein Terminvorschlag angegeben werden. Der vorgeschlagene Termin wird automatisch mit dem Status „vorgeschlagen" im Terminplaner der Station eingetragen.

Da alle von der ✍Stationsschwester ausgefüllten Formulare mit Anforderungen vor dem Abschicken an die ausgewählte leistende KE von dem ✍Stationsarzt eingesehen und verifiziert werden müssen (vgl. Anwendungsfall `Anforderung verifizieren` im Kapitel E2.2), werden diese, nachdem die ✍Stationsschwester den Befehl `SENDE` ausgewählt hat (vgl. Abbildung 20), automatisch in die jeweiligen Einzelleistungen zerlegt und in der Arztablage der anfordernden KE abgelegt. Die Einzelleistungen bekommen den Status „vorgeschlagen".

Jede Anforderung kann auf der anfordernden KE ausgedruckt werden (vgl. Anwendungsfall `Formular drucken`).

<u>Ausnahmen</u>

Keine Patientenauswahl:

Es kann nur dann eine Anforderung generiert werden, wenn aus dem Belegungsplan ein Patient ausgewählt wurde. Die Funktion `Anforderung generieren` aus dem Menü des Belegungsplans bleibt solange inaktiv, bis ein Patient ausgewählt wurde.

5.3.3 Die Schnittstellenbeschreibung

Im Schnittstellenmodell werden dem zukünftigen Benutzer des OERR-Systems die Anwendungsfälle anhand der einzelnen Masken, aus dem die Schnittstelle besteht, abgebildet. Es wird nicht festgelegt, wie diese zu realisieren sind. Eine solche Technik verringert die Wahrscheinlichkeit für die Entstehung von Mißverständnissen zwischen dem zukünftigen Benutzer und dem Entwickler (vgl. [JCJ+93], S.161).

Außer den Benutzungsschnittstellen können in diesem Modell auch Schnittstellen zu fremden Systemen beschrieben werden. Ein mögliches System, mit dem das OERR-System kommunizieren könnte, wäre das proCOM-Stationssystem der ROKD GmbH. Da jedoch im Rahmen dieser Arbeit eine eigenständige Anwendung entwickelt wird, wird dieser Aspekt nur durch theoretische Ansätze in den folgenden Kapitel angedeutet. So beschränkt sich die Schnittstellenbeschreibung auf die Beschreibung des entwickelten Oberflächenprototypen.

Der durch die Beschreibung der Masken zu den Anwendungsfälle entstandene Oberflächenprototyp kann mit den Formalismen, die die OOSE-Methode zur Verfügung stellt, nur unzureichend beschrieben werden. Im Kapitel 2.2.3 wurden deshalb für die Spezifikation und Dokumentation der Dialogabläufe des Oberflächenprototyps als Formalismus *Dialognetze* vorgeschlagen (vgl. [Jan93]). Mit Hilfe der Dialognetze kann das Verhalten der in der Schnittstellenbeschreibung dargestellten Masken formal dargestellt werden. So entsteht in der Analysephase das Dialognetz der Order-Entry-Teilanwendung, das in der Abbildung 18 präsentiert wird.

Die in der Order-Entry-Anwendung beteiligten Bildschirmmasken sind: *Maske_Login*, *Belegungsplan*, *Formularmaske*, *Formularmaske Details*, *Einzelleistung Suchergebnisse*, *Details der selektierten Einzelleistung* und *Anforderung verifizieren*. Diese Fenster sind in Abbildung 18 durch Kreise dargestellt. Die modalen Stellen in dem aufgestellten Dialognetz sind fett gedruckt und sind die *Formularmaske* und *Einzelleistung Suchergebnisse*. *Modale Stellen* in einem Dialognetz bedeuten, daß systemweit nur noch Transitionen schalten, die diese Stelle als Eingansstelle haben (vgl. [Jan93]). Sie dienen zur Modellierung modaler Dialoge. Die Rechtecke repräsentieren Transitionen und entsprechen den Buttons in den Dialogboxen. Die geteilten Rechtecke stellen *voll spezifizierte Transitionen* dar. Voll spezifizierte Transitionen beschreiben Bedingungen und Aktionen, wobei die obere Hälfte des Rechtecks die Bedingung und die untere Hälfte die Aktion anhand der Bedingung darstellt (vgl. [Jan93]).

Die Komponentenbeschreibung des betrachteten Dialognetzes wird im folgenden anhand der am Anwendungsfall `Anforderung generieren` beteiligten Bildschirmmasken vorgenommen. Der Teil des Dialognetzes zu diesem Anwendungsfall ist in Abbildung 18 durch den gestrichelten Rahmen hervorgehoben.

Zu Beginn des Anwendungsfalls wird das Fenster *Belegungsplan* geöffnet, in dem der Benutzer einen Patienten auswählen soll. Durch die Transition „Leistungsanforderung-Generieren“ kann zu dem Fenster *Formularmaske* verzweigt werden. Von dort aus kann die Transition „Details_1“ zu dem Fenster *Formularmaske Details* verzweigt werden. Das Fenster *Einzelleistung Suchergebnisse* enthält eine vollspezifizierte Transition, in dem eine Bedingung (Drücken des Optionsfeldes `JA` oder `NEIN`) und eine Aktion (wenn `JA` gedrückt wurde, wird die aus dem Anforderungskatalog ausgewählte Einzelleistung in der „Liste der am häufig benutzten Leistungen auf der Station“ in der Formularmaske eingetragen, ansonsten nicht) beschrieben wird. Das Fenster *Formularmaske* bleibt geöffnet, da die zugehörige Stelle als Nebenstelle (eingehender und ausgehender Pfeil) dargestellt wurde (vgl. Abbildung 18).

Es folgt eine nähere Beschreibung der Bildschirmmasken die in dem Anwendungsfall `Anforderung generieren` beteiligt sind. Die Abbildungen der Bildschirmmasken enthalten in Klammern der entsprechende Name aus dem Dialognetz aus Abbildung 18. Die Schnittstellenbeschreibung des gesamten Order-Entry-Teils befindet sich im Kapitel E2.3. Die Schnittstellenbeschreibung des Result-Reporting-Teils wird in dieser Phase nicht näher spezifiziert. Die Bildschirmmasken wurden in der Analysephase mit dem Dialog-Editor von Microsoft Word, mit den dazugehörigen Steuerelemente konstruiert und im Rahmen des 4. German HANSA Meetings (vgl. [Pri97]) den ROKD- und Mathias-Spital-Mitarbeiter vorgestellt. Die sich aus dieser Validierung ergebenden Ergebnisse bzw. Verbesserungsvorschläge sind in der folgenden Schnittstellenbeschreibung an den entsprechenden Stellen erwähnt.

Schnittstellenbeschreibung des Anwendungsfalles `Anforderung generieren`

Nachdem sich die ✍Stationsschwester gegenüber dem OERR-System identifiziert hat, erscheint als erste Bildschirmmaske der Belegungsplan der Station, die von der ✍Stationsschwester ausgewählt wurde (vgl. Abbildung 19).

Der Belegungsplan wird mittels einer Tabelle dargestellt, die sämtliche Informationen über die auf der Station befindlichen Patienten enthält. Die Spalten der Tabelle enthalten Name, Vorname und Bett-Nr. eines Patienten. Die Zeilen der Tabelle stellen die Zimmernummern dar. So ist beispielsweise Zimmer 426 ein 2-Bett-Zimmer und das zweite Bett wird von dem Patient „Schuster, Johann“ belegt. Unter dem Belegungsplan befinden sich drei Buttons, die jeweils das Schwester-, Arzt- und Rundgehfach (gemeinsame Ablage für Ärzte und Schwester auf einer Station) symbolisieren. Falls sich ungelesene Nachrichten in einer der Fächer befinden, wird dies optisch durch einen Briefkasten dargestellt (vgl. in Abbildung 19 das Schwesternfach).

Neben den vorgestellten Dialogobjekten stellt die Bildschirmmaske aus Abbildung 19 auch eine Menüleiste dar, die das für diesen Anwendungsfall relevante Menü „Leistungsanforderung“ zur Verfügung stellt. In diesem Menü sind die Funktionen *Generieren*, *Editieren* und *Drucken* auswählbar, wobei diese

Menüpunkte weitere, in Abbildung 19 nicht dargestellte Untermenüpunkte enthalten. Diese Untermenüpunkte ermöglichen die Auswahl des Anforderungstyps. So könnten beispielsweise Generierungsfunktionen für die Anforderungstypen Röntgen, EKG oder Labor vorhanden sein.

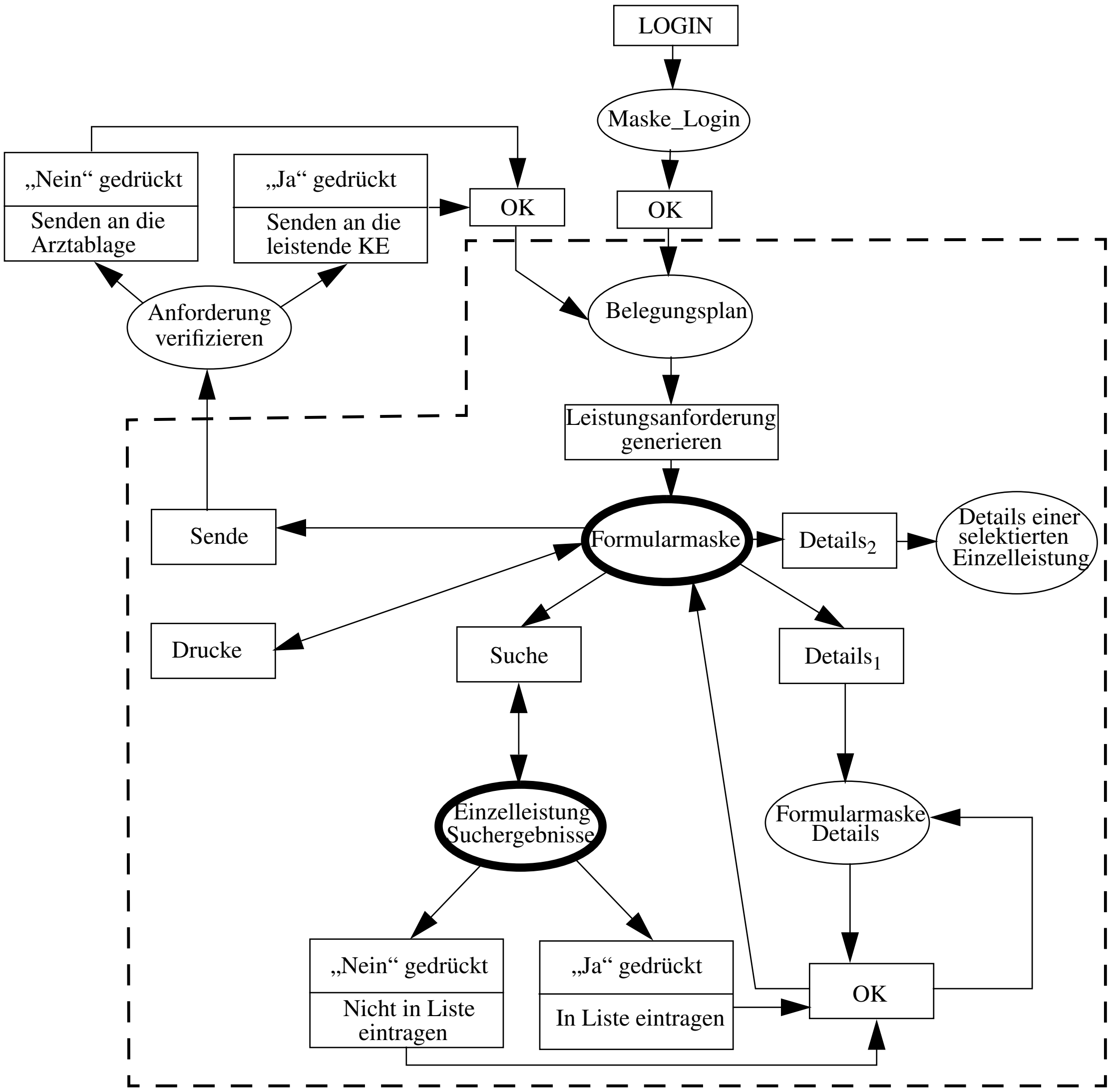

Abbildung 18: Dialognetz der Order-Entry-Anwendung in der Analysephase

Die ✍Stationsschwester muß folgende Schritte durchführen, um eine Anforderung (in diesem Beispiel vom Typ Röntgen) für einen Patienten zu generieren:

1. Auswählen eines Patienten aus dem Belegungsplan der Station, für den eine Röntgen-Anforderung beispielsweise bei der Visite verordnet wurde: die Auswahl erfolgt durch Anklicken des Namens des Patienten in dem betreffenden Feld auf der Bildschirmoberfläche und ist Voraussetzung für die Durchführung der weiteren Schritte.

2. Auswählen des Menüpunktes `Leistungsanforderung -> Generieren -> Röntgen`: Auf dem Bildschirm erscheint eine zweite Bildschirmmaske mit der Röntgen-Formularmaske (vgl. Abbildung 20).

   Diese Formularmaske besteht aus fünf Gruppenfeldern (vgl. Terminologie des MS-Word Dialogeditors in [MS94]), die folgende Informationen des ausgewählten Patienten enthalten:

   Gruppenfeld „Patient“ beinhaltet: *ID*, *Name*, *Geburtsdatum*, *Blutdruck*, *Abfrage über vorige Aufenthalte*, *Fortbewegungsart*, *Patientenliste*. Die Felder ID, Name und Geburtsdatum werden automatisch mit Informationen ausgefüllt, die durch die aktuelle Patientenauswahl aus der ersten Bildschirmmaske (vgl. Abbildung 19) als Kontext übergeben wurden. Der Blutdruck des Patienten wird von der ✍Stationsschwester gemessen und in den entsprechenden Felder in der Formularmaske eingetragen. Weiterhin besteht die Möglichkeit, falls der falsche Patient aus dem Belegungsplan ausgewählt wurde, aus der Patientenliste einen anderen Patienten zu selektieren, ohne in dem Belegungsplan zurückkehren zu müssen. Diese Patientenliste ist durch eine *Listbox* dargestellt und enthält die Namen aller Patienten auf der Station. In dem Teil „Vorherige Aufenthalte“ kann die Stationsschwester angeben, ob der Patient sich in diesem Krankenhaus in der Vergangenheit aufgehalten hat. Diese Informationen sind für den Röntgen-Funktionsbereich bedeutend, da anhand der Krankengeschichte des Patienten nachvollziehbar ist, wann er beispielsweise das letzte mal geröntgt wurde und mit welchem Ergebnis. Der „Transport“-Teil gibt Angaben über die Fortbewegungsart des Patienten zum Zeitpunkt der Anforderungsgenerierung. Hier stehen drei Optionsfelder zur Auswahl: zu Fuß, liegend und sitzend. Durch diese Angaben soll festgelegt werden, ob der Patient zu der leistenden KE geschickt oder gebracht werden soll, sowie die benötigten Transportgeräte (Rollstuhl, mobiles Bett).

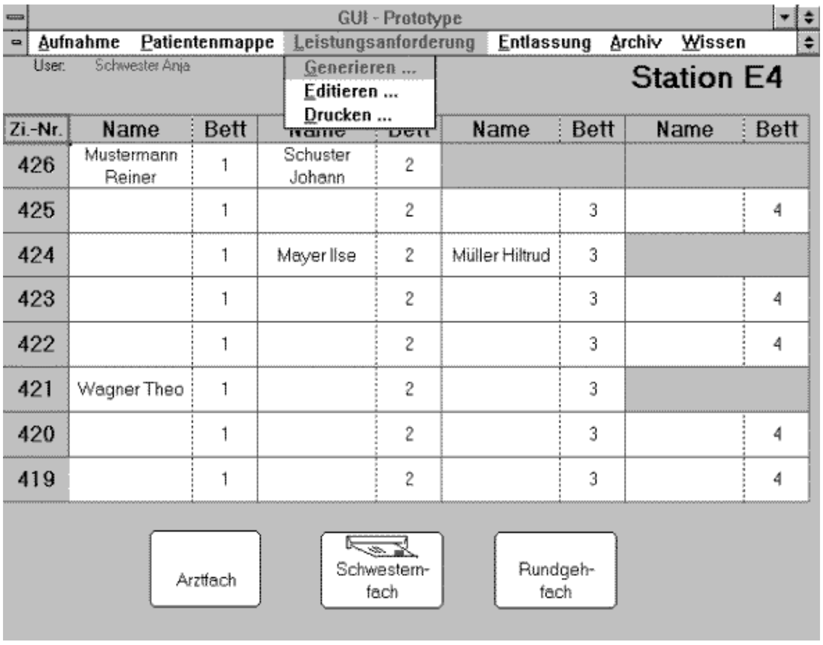

Abbildung 19: Belegungsplan einer Station (Belegungsplan)

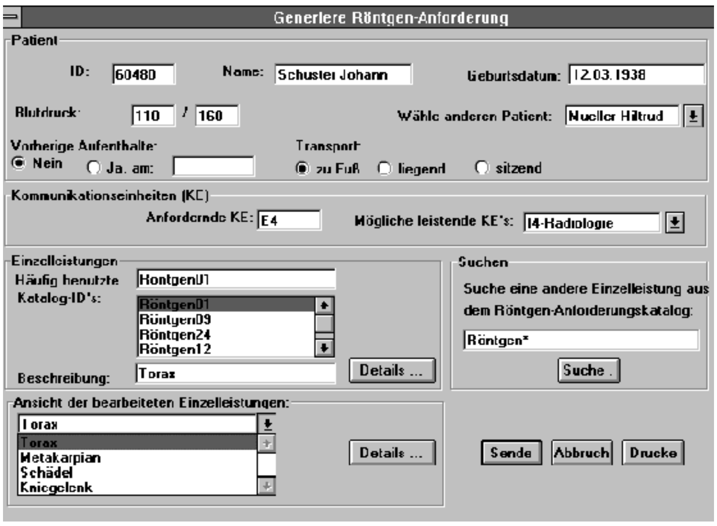

Abbildung 20: Formularmaske für die Generierung einer Röntgen-Anforderung (Formularmaske)

Gruppenfeld „Kommunikationseinheiten (KE)" beinhaltet: *ID der anfordernden KE*, *Liste mit ID's der möglichen leistenden KE's*. Diese Felder bzw. Liste werden automatisch beim Laden der Bildschirmmaske ausgefüllt. Die ✍Stationsschwester muß aus der Liste eine leistende KE auswählen, für die sie die Anforderung generiert.

Gruppenfeld „Einzelleistungen" beinhaltet: *Liste mit häufig benutzten Katalog-ID's*. Diese Liste enthält die ID's und Kurzbeschreibungen der Einzelleistungen aus einem Anforderungskatalog, die am häufigsten auf der Station angefordert werden.

Gruppenfeld „Suchen" beinhaltet: *Eingabefeld*. In diesem Gruppenfeld ist es möglich, eine Suche in dem gesamten Anforderungskatalog zu starten. Diese Suche ist dann erforderlich, falls eine Einzelleistung generiert werden soll, die sich nicht in der „Liste mit häufig benutzten Katalog-ID's" befindet. In dem Eingabefeld wird der gewünschte Suchbegriff angegeben.

Gruppenfeld „Ansicht der bearbeiteten Einzelleistungen" beinhaltet: *Liste mit den gewählten Einzelleistungen*. Eine Anforderung kann aus mehreren Einzelleistungen zusammengesetzt werden (vgl. die Beschreibung des Begriffes Einzelleistung im Problembereichsmodell). Falls beispielsweise für einen Patienten eine Torax- und eine Metakarpian-Röntgen-Anforderung von dem ✍Stationsarzt gleichzeitig verordnet wurde, so können auch beide gleichzeitig generiert werden, um somit Zeit zu sparen und eventuelle Eingabefehler zu vermeiden. Die gemeinsam ausgewählten Einzelleistungen können in diesem Gruppenfeld mittels eines *Drop-Down*-Listenfeldes eingesehen werden.

3. Ein dritter Schritt ist das Bearbeiten des Gruppenfeldes „Einzelleistungen": nachdem eine Einzelleistung aus der Liste des Gruppenfeldes „Einzelleistungen" ausgewählt wurde (Beispiel: Einzelleistung „Röntgen01"), müssen weitere Details zu dieser Leistung ausgefüllt werden. Dies geschieht, indem die ✍Stationsschwester auf den Button `Details` aus dem Gruppenfeld „Einzelleistungen" anklickt. Infolge dessen erscheint eine zusätzliche Formularmaske auf dem Bildschirm mit dem Namen „Röntgen-Anforderung generieren - Details" (vgl. Abbildung 21).

   Die Gruppenfelder „Anforderung" und „Patient" sind automatisch ausgefüllt und können durch den Benutzer in dieser Maske nicht geändert werden. Sie dienen lediglich dazu, den Überblick der Dialogabläufe nicht zu verlieren und immer zu wissen, auch bei einer kurzen Unterbrechung der Arbeit, an welcher Stelle sich der Benutzer befindet.

   In der Maske aus Abbildung 21 befinden sich noch vier weitere Gruppenfelder:

   Gruppenfeld „Priorität" beinhaltet: *drei Optionsfelder* - Routine, Eilt und Notfall. Dadurch kann die Priorität der ausgewählten Einzelleistung angegeben werden.

   Gruppenfeld „Periode" beinhaltet: *Drop-Down-Listenfeld*. Dieses Listenfeld ermöglicht die Auswahl der Periode für die Durchführung der ausgewählten Einzelleistung (*Beispiel: die ausgewählte Einzelleistung „Röntgen01" soll nur einmal durchgeführt werden*).

   Gruppenfeld „Klinische Fragestellung" beinhaltet: *Textfeld*. In diesem Textfeld können eventuelle Hinweise oder Mitteilungen bezüglich der Durchführung der ausgewählten Einzelleistung an die leistende KE gesendet werden.

   Gruppenfeld „Termin" beinhaltet: *Optionsfelder*, *Textfelder*, *Listenfeld*. Die Optionsfelder geben dem Benutzer die Möglichkeit, einen Termin für die Durchführung der Einzelleistung automatisch zu vereinbaren oder vorzuschlagen. Falls ein Termin vorgeschlagen wird, so werden zusätzlich für die Eingabe des Datums und der Uhrzeit zwei Felder aktiviert. Im Listenfeld wird beim Öffnen dieser Maske automatisch der Terminkalender der leistenden KE für den aktuellen Monat geladen. So kann die ✍Stationsschwester schon bei der Generierung der Anforderung einen gewünschten Termin vorschlagen, der dann von der leistenden KE zu bestätigen ist.

   Diese Möglichkeit, eine Einsicht in dem Terminplaner einer anderen Station bzw. Funktionsbereich zu erlauben, war bei der Präsentation der Bildschirmmasken beim 4. German HANSA Meeting sehr umstritten. Insbesondere die Mathias-Spital-Mitarbeiter hielten diese Lösung für unrealistisch und ungeeignet, da keine Station den eigenen Terminkalender auch nur zum Einsehen anderen Stationen freigeben würde, obwohl in diesem Szenario nur eine Einsicht und kein Selbsteinfügen von Terminen in fremde Terminplaner möglich ist. Diese Vorgehensweise der Terminvereinbarung wurde bei der tatsächlichen Realisierung des OERR-Systems geändert. So wurde auf das Laden des Terminkalenders der entsprechenden leistenden KE verzichtet. Stattdessen wurde die Möglichkeit geschaffen, einen Termin auszuwählen und der leistenden KE zu senden, zweck Bestätigung oder Neuvorschlag (vgl. Anwendungsfall `Anforderung generieren` im Kapitel E2.2).

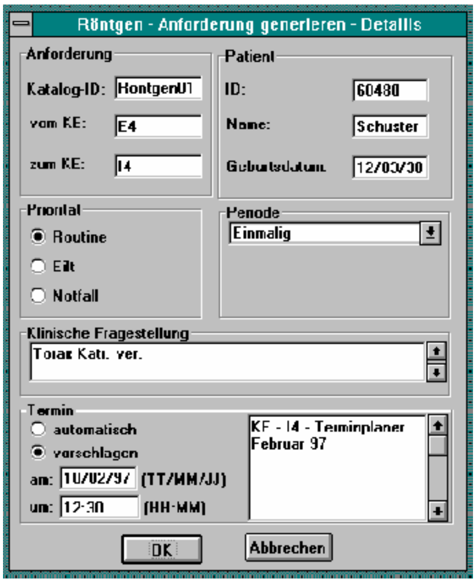

Abbildung 21: Formularmaske mit zusätzliche Details für die Röntgen-Anforderung (Formularmaske Details)

Nachdem die ✍Stationsschwester die Gruppenfelder in der Maske aus Abbildung 21 ausgefüllt hat, kehrt sie durch Anklicken des Buttons `OK` zu der Formularmaske aus Abbildung 20 zurück. Falls sie die Eingaben stornieren möchte, so braucht sie nur auf den Button `ABBRECHEN` zu klicken.

Falls für einen Patienten eine Einzelleistung anzufordern ist, die sich nicht in der Liste der am häufig angeforderten Leistungen auf der Station befindet, so besteht durch Eingabe eines Suchbegriffes die Möglichkeit einer Suche im gesamten Anforderungskatalog. Wird beispielsweise die Einzelleistung mit dem Katalog-ID „Röntgen02“ benötigt, so wird im Textfeld des Gruppenfeldes „Suchen“ in der Maske aus Abbildung 20 der Suchbegriff „Röntgen*“ eingegeben. Durch einen Filter wird beim Anklicken des Buttons `SUCHE` aus demselben Gruppenfeld eine Suche der Einzelleistungen im Anforderungskatalog gestartet, deren ID's mit „Röntgen“ beginnen und beliebig viele weitere Zeichen enthalten. Die Ergebnisse der Suche werden in einer weiteren Bildschirmmaske dargestellt (vgl. Abbildung 22).

Die Maske „Ergebnisse - Suche Einzelleistungen“ aus Abbildung 22 besteht aus dem Gruppenfeld „Ergebnisse“ sowie zwei weiteren Optionsfelder.

Gruppenfeld „Ergebnisse“ beinhaltet: *Verknüpftes Listenfeld (Listenfeld und Textfeld)*. Das Listenfeld enthält die Ergebnisse der Suche, d.h. die Katalog-ID's der Einzelleistungen, die mit „Röntgen“ beginnen. Durch Anklicken eines Eintrages in der Liste wird die dazugehörige Beschreibung der Einzelleistung in einem Textfeld angegeben. Die ✍Stationsschwester hat die Möglichkeit, die ausgewählte Einzelleistung in die Liste der häufig angeforderten Leistungen aufzunehmen oder die Einzelleistung nur für die aktuelle Anforderung zu benutzen. Unabhängig davon, welche der beiden erwähnten Möglichkeiten ausgewählt wurde, wird nach dem Klicken des Buttons `OK` aus Abbildung 22 die Maske aus Abbildung 21 erscheinen (vgl. Dialognetz in der Abbildung 18) und der vorige Schritt kann wiederholt werden.

4. Die Schritte zwei und drei können mehrmals wiederholt werden. Durch Anklicken einer schon ausgefüllten Einzelleistung im Gruppenfeld „Ansicht der bearbeiteten Einzelleistungen“ und des Buttons `Details` (vgl.Abbildung 20) können die schon festgelegten Details in einer weiteren Bildschirmmaske angesehen werden (vgl. Abbildung 23).

   Diese Bildschirmmaske beinhaltet folgende Spalten: Einzelleistung, Priorität, Periode, Klinische Frage, Termin. Durch Anklicken des Buttons `OK` wird diese Maske geschlossen und die Maske aus Abbildung 20 wird wieder aktiv. Dieses Schritt kann für jeden Eintrag aus dem Gruppenfeld „Ansicht der bearbeiteten Einzelleistungen“ aus Abbildung 20 wiederholt werden.

Abbildung 22: Maske für die Ergebnisse der Einzelleistungssuche (Einzelleistung-Suchergebnisse)

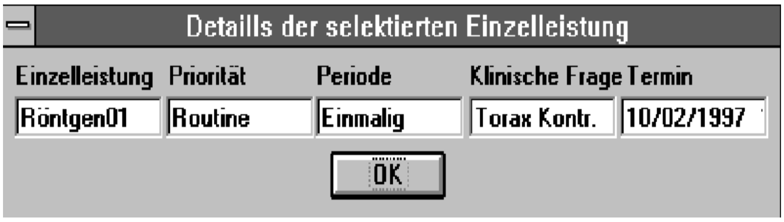

Abbildung 23: Details einer selektierten Einzelleistung

5. Die Maske aus Abbildung 20 beinhaltet abschließend noch drei Buttons: `SENDE`, `ABBRUCH`, `DRUK-KE`. Falls die ✍Stationsschwester die abgearbeitete Anforderung dem ✍Stationsarzt zwecks Verifizierung senden möchte, so muß sie auf den Button `SENDE` klicken. Die Anforderung wird dann in dem Arztfach abgelegt. Die Anforderung hat den Status „vorgeschlagen“ und dieser Status wird erst zu „angefordert“ geändert, wenn der ✍Stationsarzt sie verifiziert und der leistenden KE gesendet hat. Durch Anklicken des Buttons `DRUCKE` besteht die Möglichkeit, das ausgefüllte Anforderungsformular, mit einem speziellen Layout, das auch alle festgelegten Details berücksichtigt, auf dem Stationsdrucker auszudrucken. Das Betätigen des Buttons `ABBRUCH` verwirft alle Eingaben aus dieser Formularmaske und der Belegungsplan der Station wird geladen.

5.3.4 Das Analysemodell

In der Robustheitsanalyse wird aus den dokumentierten Anwendungsfällen ein Modell miteinander kommunizierender Objekte aufgestellt: das Analysemodell. Hierbei stehen drei grundsätzliche Objekttypen mit den dazugehörigen Assoziationen zur Verfügung, die im Kapitel 2.2.2 beschrieben wurden: *Schnittstellenobjekte*, *Entitätsobjekte* und *Kontrollobjekte*. Da bei der Aufstellung des Analysemodells von *idealen Umgebungsbedingungen* ausgegangen wird (vgl. Kapitel 2.2.2), kann die im Entwurfsmodell festgelegten Entwicklungsumgebung zu Änderungen im Analysemodell führen.

Analog zu den anderen vorgestellten Modellen werden auch bei der Beschreibung des aufgestellten Analysemodells zunächst exemplarisch die Objekte und Assoziationen zwischen den Objekten beschrieben, die zum Anwendungsfall `Anforderung generieren` gehören. Diese sind in Abbildung 24 dargestellt. Das vollständige Analysemodell befindet sich im Kapitel E2.4.

Teil des Analysemodells für den Anwendungsfall `Anforderung generieren`

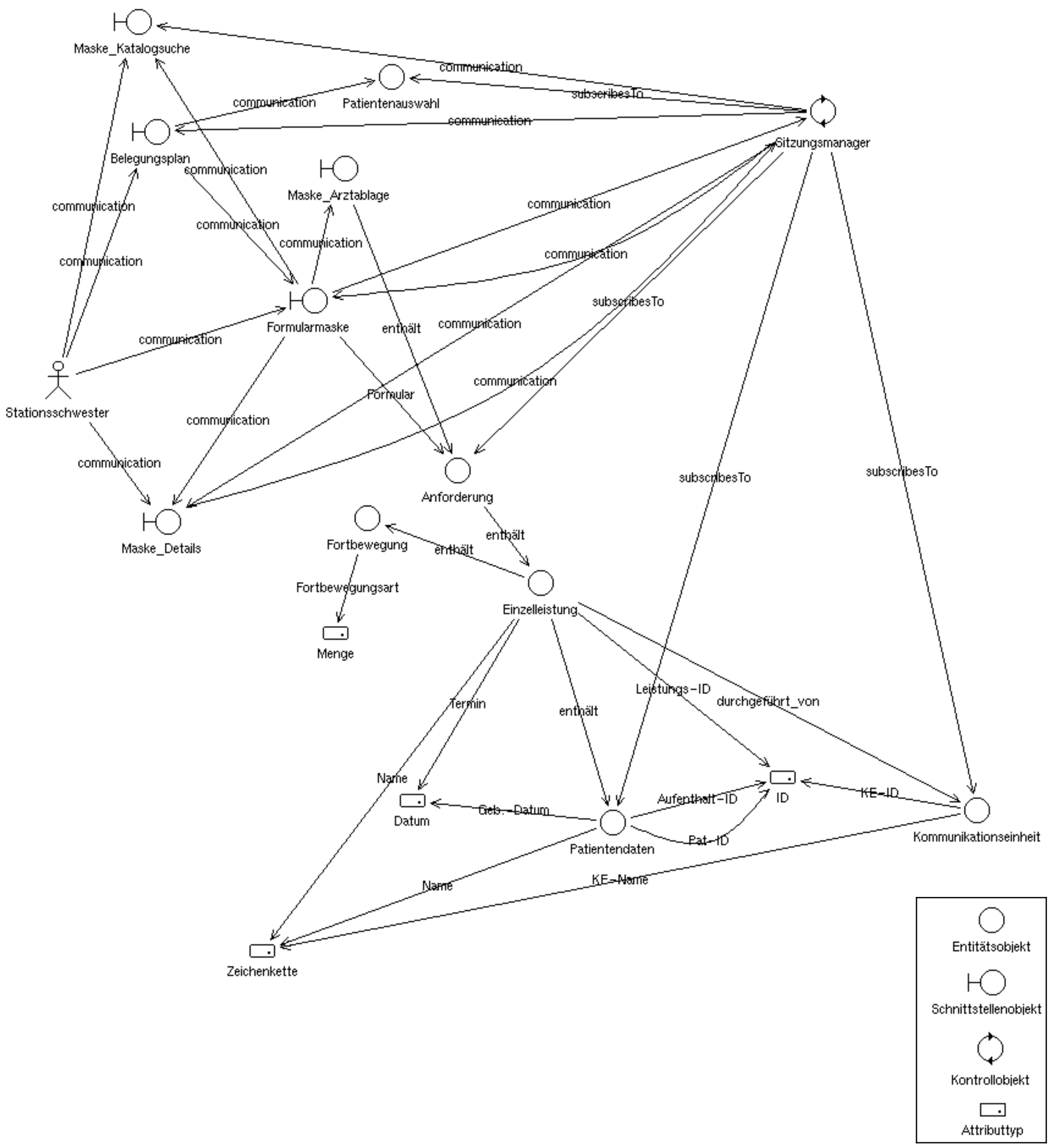

Abbildung 24: Analysemodell - Anwendungsfall `Anforderung generieren`

Das aufgestellte Diagramm enthält Entitätsobjekte, Schnittstellenobjekte, Kontrollobjekte, Attributtypen und Assoziationen zwischen diesen Objekttypen. Diese Elemente werden im folgenden einzeln beschrieben.

5.3.4.1 Entitätsobjekte

Die **Entitätsobjekte** (vgl. auch Kapitel 2.2.2) entsprechen weitgehend den im Problembereichsmodell dargestellten Objekte. Bei dem hier betrachteten Anwendungsfall `Anforderung generieren` werden folgende Objekte aus dem Problembereichsmodell im Analysemodell als Entitätsobjekte modelliert: *Einzelleistung*, *Anforderung*, *Patientendaten* und *Fortbewegung* (vgl. Abbildung 24). Zusätzliche Informationen zu diesen Objekten werden in Attributen gespeichert. Die einzige Ausnahme stellen die Entitätsobjekte *Patientenauswahl* und *Kommunikationseinheit* dar, für die keine entsprechende Objekte im Problembereichsmodell existieren. Diese Entitätsobjekte werden nicht aus dem Problembereichsmodell übernommen, sondern für das Analysemodell neu eingefügt. Weil sich die Objekte *Patientenauswahl* und *Kommunikationseinheit* bereits auf den internen Aufbau eines OERR-Systems beziehen, aber keine allgemeine Begriffe eines solchen Systems darstellen, die auch von dem zukünftigen Anwender benutzt werden, sind diese Objekte nicht im Problembereichsmodell aufgeführt. Beide Objekte zusammen mit den weiteren, für die Modellierung des Anwendungsfalls `Anforderung generieren` benötigten Entitätsobjekten, werden mit deren Attributen und Assoziationen zu anderen Objekten im folgenden beschrieben:

Entitätsobjekt *Anforderung*

Beschreibung: In diesem Objekt werden Informationen gespeichert, die für die Durchführung von Untersuchungen bei einem Patienten auf einer Station notwendig sind.

Assoziationen:

Zum Objekt	Name	Kardinalität	Beschreibung
Einzelleistung	enthält	1..n	Die Einzelleistung ist Bestandteil einer Anforderung.

Attribute: keine

Entitätsobjekt *Einzelleistung*

Beschreibung: In diesem Objekt werden die Informationen einer Einzelleistung gespeichert.

Assoziationen:

Zum Objekt	Name	Kardinalität	Beschreibung
Fortbewegung	enthält	1	Das Objekt *Einzelleistung* enthält Informationen über die Fortbewegung eines Patienten.
Kommunikations-einheit	durchgeführt_von	1	Eine *Einzelleistung* wird von einer Kommunikationseinheit durchgeführt.
Patientendaten	enthält	1	Das Objekt *Einzelleistung* enthält Informationen über Patientendaten eines Patienten.

Attribute:

Name	Typ	Kardinalität	Beschreibung
Leistungs-ID	ID	1	Die Identifikationsnummer der Einzelleistung ist eindeutig.
Termin	Datum	1	Termin der Einzelleistung
Name	Zeichenkette	1	Name der Einzelleistung

Entitätsobjekt *Fortbewegung*

Beschreibung: Dieses Objekt speichert die Informationen über die Fortbewegungsart eines Patienten.

Assoziationen: keine

Attribute:

Name	Typ	Kardinalität	Beschreibung
Fortbewegungsart	Menge	3	Fortbewegungsart des Patienten: zu Fuß, liegend, sitzend.

Entitätsobjekt *Kommunikationseinheit*

Beschreibung: Dieses Objekt speichert Informationen über die Station und den Funktionsbereich, die in die Durchführung einer Anforderung involviert sind.

Assoziationen: keine

Attribute:

Name	Typ	Kardinalität	Beschreibung
KE-ID	ID	1	Die Identifikationsnummer der KE ist eindeutig.
KE-Name	Zeichenkette	1	Name einer Kommunikationseinheit

Entitätsobjekt *Patientendaten*

Beschreibung: Dieses Objekt speichert Informationen über einen Patienten.

Assoziationen: keine

Attribute:

Name	Typ	Kardinalität	Beschreibung
Pat-ID	ID	1	Die Identifikationsnummer des Patienten ist eindeutig.
Aufenthalt-ID	ID	1..n	Aufenthalts-Identifikationsnummer des Patienten; Ein Patient kann mehr als eine Aufenthalts-ID besitzen, falls er mehrmals in demselben Krankenhaus behandelt wurde.
Name	Zeichenkette	1	Name des Patienten
Geb.-Datum	Datum	1	Geburtsdatum des Patienten

Entitätsobjekt *Patientenauswahl*

Beschreibung: In diesem Objekt werden Informationen eines aus dem Belegungsplan ausgewählten Patienten gespeichert.

Assoziationen: keine

Attribute: keine

5.3.4.2 Schnittstellenobjekte

Die **Schnittstellenobjekte** (vgl. auch Kapitel 2.2.2), die bei der Modellierung des Anwendungsfalls `Anforderung generieren` identifiziert wurden, sind: *Belegungsplan*, *Formularmaske, Maske_Details, Maske_Katalogsuche* und *Maske_Arztablage* (vgl. Abbildung 24). Keiner dieser Objekte enthält Attribute. Mit Hilfe von „communication"-Beziehungen wird dargestellt, welche Objekte (bzw. Akteure) untereinander verschiedene Stimuli austauschen. Ein Objekt, von dem eine Kante zur Darstellung einer „communication"-Beziehung ausgeht, kann Stimuli an das Objekt senden, auf das die Kante zeigt, um dort die Ausführung von Operationen zu veranlassen. Folgendes Beispiel soll dieses Verhalten veranschaulichen: wenn der Akteur ✍*Stationsschwester* den Belegungsplan ansehen möchte, so sendet er einen Stimulus an das Schnittstellenobjekt *Belegungsplan*, das die Ausführung der gewählten Operation veranlaßt. Im folgenden werden die erwähnten Schnittstellenobjekte beschrieben:

Schnittstellenobjekt *Belegungsplan*

Beschreibung: In diesem Objekt sind die Operationen für die Kommunikation der Akteure ✍Stationsschwester und ✍Stationsarzt mit dem *Belegungsplan* enthalten.

Assoziationen:

Zum Objekt	Name	Kardinalität	Beschreibung
Patientenauswahl	communication	1	Das Objekt *Belegungsplan* sendet einen Stimulus an das Objekt *Patientenauswahl*, wenn ein Patient im Belegungsplan durch den Akteur ausgewählt wurde.
Formularmaske	communication	1	Das Objekt *Belegungsplan* sendet einen Stimulus an das Objekt *Formularmaske*, damit diese geöffnet wird.

Schnittstellenobjekt *Formularmaske*

Beschreibung: In diesem Objekt sind die Operationen für die Kommunikation des Akteurs ✍Stationsschwester mit der *Formularmaske* enthalten. Die *Formularmaske* dient der Darstellung von Formularen auf der Bildschirmoberfläche.

Assoziationen:

Zum Objekt	Name	Kardinalität	Beschreibung
Sitzungsmanager	communication	1	Das Objekt *Formularmaske* sendet einen Stimulus an das Objekt *Sitzungsmanager* mit den Daten, die in einer weiteren Maske geladen werden müssen (z.B. *Maske_Details*).
Maske_Details	communication	1	Das Objekt *Formularmaske* sendet einen Stimulus an das Objekt *Maske_Details*, wenn der entsprechende Button von dem Benutzer betätigt wurde.
Maske_Katalogsuche	communication	1	Das Objekt *Formularmaske* sendet einen Stimulus an das Objekt *Maske_Katalogsuche*, wenn der entsprechende Button von dem Benutzer betätigt wurde.
Maske_Arztablage	communication	1	Das Objekt *Formularmaske* sendet einen Stimulus an das Objekt *Maske_Arztablage*, wenn der entsprechende Button (`ARZTABLAGE`, siehe Abbildung 11) von dem Benutzer betätigt wurde.
Anforderung	Formular	1	In der *Formularmaske* wird eine Anforderung geöffnet.

Schnittstellenobjekt *Maske_Details*

Beschreibung: In diesem Objekt sind die Operationen für die Kommunikation des Akteurs ✍Stationsschwester mit der *Maske_Details* enthalten. *Maske_Details* dient der Darstellung von detaillierten Informationen zu einer ausgewählten Einzelleistung auf der Bildschirmoberfläche (vgl. Abbildung 12).

Assoziationen:

Zum Objekt	Name	Kardinalität	Beschreibung
Sitzungsmanager	communication	1	Das Objekt *Maske_Details* sendet einen Stimulus an das Objekt *Sitzungsmanager* mit den Daten, die in einer weiteren Maske geladen werden müssen (z.B. *Formularmaske*).

Schnittstellenobjekt *Maske_Katalogsuche*

Beschreibung: In diesem Objekt sind die Operationen für die Kommunikation des Akteurs ✍Stationsschwester mit der *Maske_Katalogsuche* enthalten. *Maske_Katalogsuche* dient der Darstellung von Informationen für die Suche von Einzelleistungen im gesamten Anforderungskatalog auf der Bildschirmoberfläche.

Assoziationen: keine

Schnittstellenobjekt Maske *Arztablage*

Beschreibung: Dieses Schnittstellenobjekt dient der Kommunikation zwischen den beiden Akteuren des Systems, dem ✍Stationsarzt und der ✍Stationsschwester.

Assoziationen:

Zum Objekt	Name	Kardinalität	Beschreibung
Anforderung	enthält	1..n	Das Objekt *Anforderung* ist Bestandteil des Objektes *Maske_Arztablage*.

5.3.4.3 Kontrollobjekte

Das **Kontrollobjekt** (vgl. auch Kapitel 2.2.2), das bei der Modellierung des Anwendungsfalls `Anforderung generieren` identifiziert wurde, ist der *Sitzungsmanager* und wird im folgenden erläutert (vgl. Abbildung 24):

Kontrollobjekt *Sitzungsmanager*

Beschreibung: Der Sitzungsmanager steuert den Ablauf der Sitzung. Er nimmt die Anmeldung des Akteurs entgegen, erstellt eine benutzerspezifische Funktionsleiste, sorgt für den ordnungsgemäßen Abschluß der Sitzung und koordiniert und kontrolliert den Aktivitätsfluß innerhalb der Sitzung.

Assoziationen:

Zum Objekt	Name	Kardinalität	Beschreibung
Patientenauswahl	subscribesTo	1	Der *Sitzungsmanager* wird über die aktuelle *Patientenauswahl* benachrichtigt.
Patientendaten	subscribesTo	1	Der *Sitzungsmanager* wird über die aktuellen *Patientendaten* aus der *Formularmaske* informiert.
Anforderung	subscribesTo	1	Der *Sitzungsmanager* wird über die aktuell ausgewählte *Anforderung* aus der *Formularmaske* benachrichtigt.
Kommunikationseinheit	subscribesTo	1	Der *Sitzungsmanager* wird über die aktuell ausgewählte *Kommunikationseinheit* aus der *Formularmaske* informiert.

Zum Objekt	Name	Kardinalität	Beschreibung
Formularmaske	communication	1	Der *Sitzungsmanager* sendet Stimuli an das Objekt *Formularmaske* mit Daten, die in dieser Maske geladen werden müssen.
Maske_Details	communication	1	Der *Sitzungsmanager* sendet Stimuli an das Objekt *Maske_Details* mit Daten, die in dieser Maske geladen werden müssen.
Maske_Katalogsuche	communication	1	Der *Sitzungsmanager* sendet Stimuli an das Objekt *Maske_Katalogsuche* mit Daten, die in dieser Maske geladen werden müssen.
Belegungsplan	communication	1	Der *Sitzungsmanager* sendet Stimuli an das Objekt *Belegungsplan* mit Daten, die in dieser Maske geladen werden müssen.

Attribute: keine

5.3.4.4 Attributtypen

Folgende **Attributtypen** (vgl. auch Kapitel 2.2.2) wurden für die Modellierung des Anwendungsfalls `Anforderung generieren` definiert (vgl. Abbildung 24):

Attributtyp *Datum*

Beschreibung: Zahlenwert im Format TT/MM/JJJJ (T-Tag, M-Monat, J-Jahr).

Attributtyp ID

Beschreibung: Ganze Zahl.

Attributtyp *Zeichenkette*

Beschreibung: Ein Wert dieses Typs ist eine Zeichenkette.

Attributtyp *Menge*

Beschreibung: Ein Wert dieses Typs ist genau ein Eintrag aus der Menge.

5.4 Validierungstätigkeiten in der Analysephase

In diesem Kapitel werden Validierungstätigkeiten vorgestellt, die anhand des aufgestellten Validierungsplans aus Kapitel 4.3 in der Analysephase durchgeführt wurden.

Bei der Durchführung der Analysephase können u.a. folgende Kategorien von typischen Anforderungsfehlern auftreten (vgl. auch [Wal90]):

- unklare, widersprüchliche Anforderungen;
- fehlende und unvollständige Anforderungen;
- fehlerhafte, untestbare Anforderungen;

Daraus und aus den im SPARDAT-Qualitätsmodell beschriebenen Qualitätszielen (vgl. Kapitel E1) ergeben sich die im folgenden Kapitel aufgelisteten Qualitätseigenschaften und -merkmale, die in der Analysephase zu bewerten sind.

5.4.1 Validierungsziele und Auswertungsmethoden

Folgende Qualitätseigenschaften und -merkmale aus dem SPARDAT-Qualitätsmodell lassen sich in der Analysephase validieren (die entsprechenden Beschreibungen können dem Kapitel E1 entnommen werden):

Qualitätseigenschaft	**Qualitätsmerkmal**
Funktionsabdeckung	• Funktionale Vollständigkeit
Widerspruchsfreiheit	• Widerspruch zwischen Funktionalitäten, die die Funktionen erfüllen
Handhabbarkeit	• Dialogabläufe
	• Rationalität
Erweiterbarkeit	• Flexibilität der Struktur

Die zu diesen Qualitätsmerkmalen gehörenden Auswertungsmethoden sind eine Teilmenge der im Kapitel E1 aufgelisteten Auswertungsmethoden (vgl. auch Kapitel 4.4.2) und sind im Kapitel E2.5 zusammengefaßt.

5.4.2 Durchführung der Validierung

In diesem Kapitel wird exemplarisch ein Qualitätsmerkmal auf die aufgestellten Phasen-Enddokumente angewendet und bewertet. Die Validierung der gesamten Qualitätseigenschaften und -merkmale der Analysephase befindet sich im Kapitel E2.5.

Für die exemplarische Durchführung der Validierung in der Analysephase wird die Qualitätseigenschaft `Funktionsabdeckung` mit dem Qualitätsmerkmal *Funktionale Vollständigkeit* validiert. Die `Funktionsabdeckung` prüft die Vollständigkeit der Funktion eines Produktes in Bezug auf die funktionalen Anforderungen (vgl. Kapitel E1.1). Die *Funktionale Vollständigkeit* läßt sich in der Analysephase durch folgende sieben Prüffragen auswerten:

F_1: Ist jede funktionale Anforderung in der Spezifikation klar definiert?

F_2: Ist die Zusammenarbeit verschiedener funktionaler Anforderungen in der Spezifikation definiert worden?

F_3: Sind alle in der Spezifikation aufgeführten funktionalen Anforderungen notwendig?

F_4: Sind die funktionalen Anforderungen vollständig spezifiziert?

F_5: Wurden alle funktionalen Anforderungen, die der Benutzer benötigt, identifiziert und spezifiziert?

F_6: Sind alle Vorbedingungen für den Ablauf der funktionalen Anforderungen spezifiziert?

F_7: Sind die funktionalen Anforderungen verständlich für den Auftraggeber und für diejenigen, die den Entwurf durchführen?

Die Fragen F_1 bis F_7 können mit einer 1 (Zutreffend) beantwortet werden, da die geforderten Merkmale der Analysephase bereits von den charakteristischen Eigenschaften der OOSE-Methode, u.a. durch die Nachvollziehbarkeit zwischen den Modellen (vgl. Kapitel 2.2.2), sichergestellt werden. So ist beispielsweise eine klare Definition der funktionalen Anforderungen in der Analysephase (vgl. Frage F_1) durch die Aufstellung der erwähnten Modelle gewährleistet.

Die Zusammenarbeit der verschiedenen funktionalen Anforderungen (vgl. Frage F_2) wird in OOSE sowohl durch die Anwendungsfälle selbst als auch durch die Beziehungen zwischen den Anwendungsfällen definiert und sichergestellt.

Die Begründung für die positive Antwort auf die Fragen F_3, F_4, F_5 und F_7 ergibt sich aus der informellen Beschreibung der Anwendungsfälle. So hat der zukünftige Benutzer selbst die Kontrolle und kann beobachten, ob alle für ihn notwendigen funktionalen Anforderungen identifiziert und vollständig spezifiziert wurden. Die informelle Beschreibung der Anwendungsfälle zusammen mit der Schnittstellenbeschreibung kann dem Anwender schon in der ersten Entwlickungsphase eine „realistische“ Vision über das zukünftige System geben.

Die notwendigen Initialisierungsoperationen für den Ablauf der funktionalen Anforderungen (vgl. Frage F_6) wurden bei jedem Anwendungsfall als Vorbedingung im Ereignisablauf angegeben.

5.4.3 Auswertung der Ergebnisse

Durch die positive Auswertung der Fragen F_1 bis F_7 aus dem vorigen Kapitel läßt sich nachvollziehen, daß die im Kapitel 2.2.2 ausgewählte OOSE-Methode sich durch ihre systematische und nachvollziehbaren Vorgehensweise, sowie der Vereinheitlichung der informellen Dokumentation der Arbeitsergebnisse in der Analysephase, wesentliche Vorteile aufweist.

Durch den Einsatz der OOSE-Methode kann eine positive Bewertung des Entwicklungsprozesses in der Analysephase erreicht werden, durch die sichergestellt werden kann, daß die Qualitätssicherung bereits in den frühen Entwicklungsphasen berücksichtigt wird. Eine gute Basis kann somit für das weitere Entwicklungsprozeß schon in der Analysephase aufgebaut werden.

5.5 Dauer der Tätigkeiten der Analysephase

Für die Abschätzung der Enwicklungszeit eines Software-Systems wurde im Kapitel 4.2 das COCOMO-Modell beschrieben. Dieses Modell erfordert jedoch Kenntnisse über die Entwicklungszeit ähnlicher Systeme, um in der Beurteilung mittels COCOMO einen Durchschnittswert der Quellcodezeilen angeben zu können. Durch „ähnliche Systeme“ werden neue Versionen eines Produktes verstanden, die in der gleichen Programmiersprache wie frühere Produkte geschrieben wurden, um einen Vergleich und somit eine Abschätzung möglich zu machen.

Das zu entwickelte OERR-System wurde zwar auch von der Firma ROKD GmbH entwickelt (proCOM-OERR), jedoch in einer anderen Umgebung (Programmiersprache Natural LightStorm und nicht Visual Basic). Die Entwicklungszeit des OERR-Systems in Visual Basic mit Hilfe von DHE kann also nicht mit dem des proCOM-OERR-Systems verglichen werden. Die Berechnung der Dauer der Tätigkeiten der Enwicklungsphasen wird daher eher qualitativ als quantitativ relativiert. So führen zum Beispiel Vorkenntnisse über im Entwicklungsprozeß angewendete Methoden zu einer Zeitminimierung und somit zu einer guten Validierung.

6 Die Konstruktionsphase

Die Konstruktionsphase ist eine weitere Entwickungsphase, dessen Durchführung von der OOSE-Methode unterstützt wird. In diesem Kapitel wird der Validierungsplan aus Kapitel 4.3 auf die Phase der *Konstruktion* angewendet. In jedem Abschnitt (6.1 bis 6.5) wird jeweils eine Tätigkeit aus diesem Plan beschrieben und durchgeführt.

6.1 Beschreibung der Entwicklungsphase *Konstruktion*

Ziel der Konstruktionsphase ist die Konstruktion einer Implementierung, d.h. die Abbildung des idealisierten Analysemodells auf die tatsächlichen technischen und pragmatischen Randbedingungen und Umstände des Projektes (vgl. [JCJ+93]). In dieser Phase wird zum ersten Mal in dem Entwicklungsprozeß die Entwicklungsumgebung berücksichtigt. Wie im Kapitel 2.2.2 erwähnt wurde, erreicht die Konstruktionsphase dieses Ziel in zwei Schritten: die *Entwurfs*- und die *Implementierungsphase*. Die daraus resultierenden Modelle sind das *Entwurfs*- und das *Implementierungsmodell* (vgl. Abbildung 4).

6.2 Input- und Output-Parameter der Entwicklungsphase *Konstruktion*

Eingaben für die Durchführung der Konstruktionsphase des zu entwickelnden OERR-Systems sind das Anforderungs- und das Analysemodell aus der Analysephase (vgl. Abbildung 25). Output der Konstruktionsphase sind die Phasen-Enddokumente Entwurfsmodell und Implementierungsmodell. Diese beiden Modelle wurden im Kapitel 2.2.2 näher erläutert.

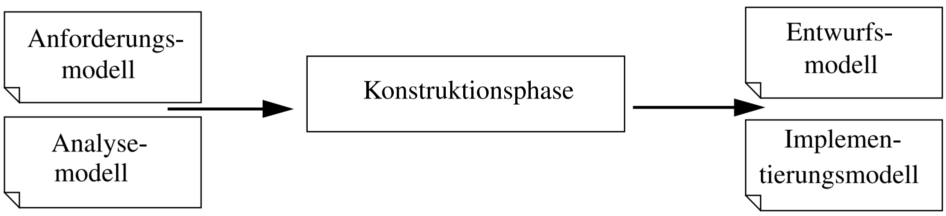

Abbildung 25: Input- und Output-Modelle der Konstruktionsphase

6.3 Aufstellung der Phasen-Enddokumente

In diesem Kapitel wird die Vorgehensweise bei der Aufstellung der Phasen-Enddokumente *Entwurfsmodell* und *Implementierungsmodell* vorgestellt.

6.3.1 Das Entwurfsmodell

Bei der Übersetzung des Analysemodells in das Entwurfsmodell wird die von Jacobson et al. (vgl. [JCJ+93]) vorgeschlagene Vorgehensweise, bestehend aus folgenden drei Schritten, angewendet:

- Identifizierung der Systemumgebung;
- Übersetzung der Analyseobjekte in Entwurfsobjekte;
- Aufstellung von Interaktionsdiagrammen.

Im folgenden werden diese Schritte einzeln näher beschrieben.

6.3.1.1 Identifizierung der Systemumgebung

Im Rahmen dieses Schrittes wird zum ersten Mal die Entwicklungsumgebung, in der das OERR-System entstehen soll, berücksichtigt. Die Umgebung bezieht sich sowohl auf Hardware- als auch auf Software-Komponenten. Es werden vorhandene Bibliothekskomponenten (beispielsweise DHE), Diagramme zur graphischen Darstellung der benutzten Datenmodelle (beispielsweise ER-Diagramm), Prozeßkommunikation (TCP/IP-Protokoll) etc. in Betracht gezogen.

Wie im Kapitel 3.1.3 erwähnt wurde, basiert DHE konzeptionell auf dem Client-Server-Prinzip. Der DHE-Server Version 1.00a wurde in der ROKD GmbH auf einem Rechner vom Typ SunSPARC20 mit einer Festplatte von 2*2GB und 128 MB RAM unter dem Betriebssystem Solaris 2.5 installiert. Als DBMS wurde dazu Sybase 10.0.2 auf demselben Rechner eingerichtet (vgl. auch Kapitel 3.1.3.1). Für die Entwicklung der OERR-Anwendung auf der Client-Seite wurde ein Compaq-PC 80486 DX2/66 MHz mit 8 MB RAM und einem Festplattenspeicher von 540 MB benutzt (vgl. auch Kapitel 3.1.3.2). Das Betriebssystem ist MS-DOS 6.22. Die ereignis-orientierte Implementierungssprache ist Visual Basic 3.0 unter Windows 3.1. Die Kommunikation zwischen dem Server und dem Client ist durch das TCP/IP-Protokoll von Novell 3.11 sichergestellt.

An dieser Stelle wird auf jene Eigenschaft der OOSE-Methode hingewiesen, die trotz ihres objekt-orientierten Ansatzes, doch die Anwendung einer nicht objekt-orientierten Programmiersprache erlaubt. Wie später bei der Beschreibung des Implementierungsmodells erwähnt wird, können objekt-orientierte in ereignis-orientierte Konzepte nach [JCJ+93] umgewandelt werden, ohne die vorigen Entwicklungsmodelle ändern zu müssen. So können die Vorteile einer objekt-orientierten Entwicklung und der damit verbundenen OOSE-Methode bis hin zu der Implementierungsphase ausgenutzt werden.

Die im Kapitel 3.1.3 beschriebene DHE-Komponente (der DHE-Server) verfügt über sechs Manager, deren statisches Verhalten durch ein erweitertes ER-Diagramm beschrieben werden kann. Durch eine genauere Betrachtung der identifizierten charakteristischen Vorgänge des OERR-Systems aus Kapitel 4.4 wird deutlich, daß die Funktionalität des *Act Manager* der DHE-Software der zentrale Aspekt des OERR-Systems darstellt. Diese Aussage kann wie folgt begründet werden:

Der Act Manager (Aktivitätenmanager) definiert die Beziehungen zwischen Leistungsanbietern und anfordernden Stellen und sichert für die Dauer der Aktivität eine ständige Kommunikationsverbindung zwischen beiden. Ein *Act* kann als eine Aktivität im klinischen Alltag verstanden werden. Er kennzeichnet beispielsweise einen Kommunikationsvorgang mit folgenden charakteristischen Eigenschaften (vgl. Abbildung 26):

- die Aktivität bezieht sich auf einen bestimmten Patienten;
- die Aktivität wird von einer Person angefordert (Anfordernder);
- die Aktivität wird von einer oder mehreren verantwortlichen Personen durchgeführt (Leistender);
- die Aktivität hat einen veränderbaren Status (beispielsweise *angefordert*, *durchgeführt*, *befundet*);
- die Aktivität erzeugt ein oder mehrere Ergebnisse.

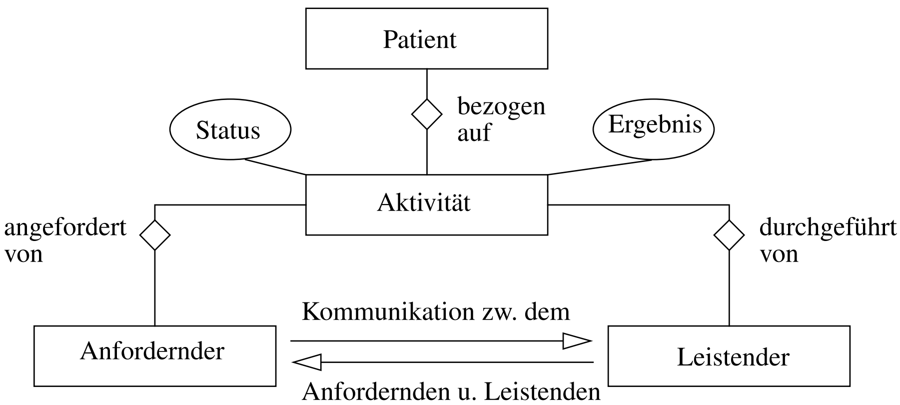

Abbildung 26: Act Management bei OERR

Aktivitäten im OERR-Kontext sind beispielsweise die Anforderung einer Leistung, die Durchführung dieser Leistung, die Dokumentation der Ergebnisse der durchgeführten Leistung etc.

Der Vorteil des Act Managements liegt zum einen in der Möglichkeit, sämtliche notwendige Kommunikationen über einen zentralen Prozeß, den Act Manager, ablaufen zu lassen, wobei dieser automatisch die Korrektheit der Kommunikation überprüft (*Beispiel: wurde eine angeordnete Aktivität auch zum richtigen Leistenden weitergeleitet?*). Zum anderen erfährt der Anfordernde zu jedem Zeitpunkt, welchen Status seine Aktivität besitzt.

Der Act Manager ist zwar der zentrale Aspekt des OERR-Systems, jedoch nicht der einzige Manager der DHE-Software, welches vom OERR-System benutzt wird. Für die Verwaltung der Daten durch DHE werden noch folgende Manager verwendet: *Patient Manager*, *Ressource Manager*, *User & Authorisation Manager* und *Healthdata Manager*. Angelehnt an die ER-Notation wird in [Fer95] mittels eines Diagramms die Zusammenarbeit sowie die Möglichkeit des Einsatzes dieser Manager für das OERR-System beschrieben (vgl. Abbildung 27).

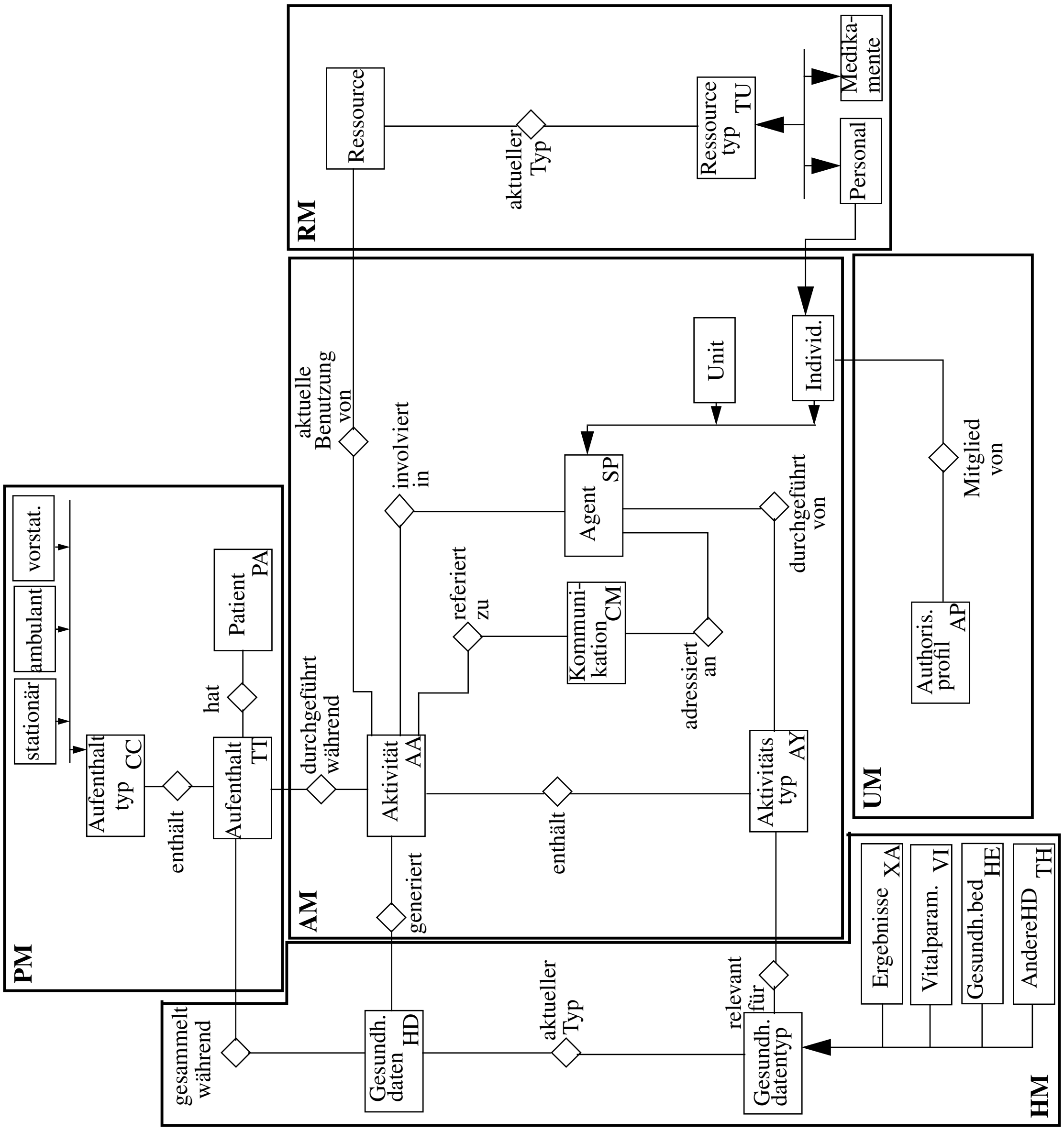

Abbildung 27: Darstellung der für die aus DHE in OERR verwendeten Daten (vgl. [Fer95])

Der Patient Manager PM verwaltet die personenbezogenen Daten eines Patienten. Ein Patient (Entitätsbezeichnung laut DHE: PA) kann einen oder mehrere Aufenthalte (TT) in einem Krankenhaus haben. Die Aufenthalte können unterschiedlicher Art sein (stationär, ambulant, vorstationär etc.)(CC). Jede klinische Aktivität (AA) in einem Krankenhaus wird in Zusammenhang mit einem Aufenthalt eines Patienten durchgeführt (Act Manager AM). Die Aktivitäten sind in verschiedene Aktivitätstypen eingeteilt (z.B. Ruhe-EKG, Rhythmus-EKG)(AY). Ein Aktivitätstyp wird von einem Agent (Station, Funktionsbereich, Person)(SP) durchgeführt. Ein Agent ist somit an der Durchführung einer Aktivität beteiligt. Die Aktivitäten können durch eine Kommunikation (CM) einem Agenten gesendet werden. Die Durchführung einer Aktivität erfordert Ressourcen (Ressource Manager RM). Die Ressourcen können unterschiedlicher Art sein (Personal, Medikamente, Betten etc.)(TU). In erster Linie wird eine Aktivität von einer Person durchgeführt. Für sie wurde in DHE ein Authorisierungsprofil definiert (AP)(User & Authorisation Manager UM). Jede durchgeführte Aktivität generiert Gesundheitsdaten (HD) unterschiedlicher Art (Leistungsergebnisse, Vitalparameter etc.)(HealthData Manager HM). Diese Gesundheitsdatenttypen (XA,VI, HE,TH) sind für verschiedene Aktivitätstypen relevant. Alle Gesundheitsdaten werden während des Aufenthaltes eines Patienten gesammelt.

In welcher Art und Weise sich die Entitäten und Beziehungen aus Abbildung 27 in das weitere Modellieren des OERR-Systems miteinbeziehen lassen, wird in den nächsten beiden Schritten der Aufstellung des Entwurfsmodells festgelegt.

6.3.1.2 Übersetzung der Analyseobjekte in Entwurfsobjekte

Die Berücksichtigung der Randbedingungen der Systemumgebung im vorigen Schritt ermöglicht nun das weitere Aufstellen des Entwurfsmodells. In diesem Schritt werden die Objekte aus dem Analysemodell zusammen mit den dazugehörigen Assoziationen im Entwurfsmodell modelliert. Für Objekte im Entwurfsmodell schlagen Jacobson et al. die Bezeichnung *Block* vor (vgl. [JCJ+93]). Die Bezeichnungen Block und Objekt werden im folgenden bei der Beschreibung des Entwurfsmodells synonym benutzt.

Die Übersetzung der Analyseobjekte in Entwurfsobjekte kann mechanisch durchgeführt werden: jedes Analyseobjekt, sei es Entitätsobjekt, Schnittstellenobjekt oder Kontrollobjekt, wird in jeweils einen Block umgesetzt. Diese Transformation führt zu einer klaren und eindeutigen Nachvollziehbarkeit in den Modellen, so wie sie im Kapitel 2.2.2 beschrieben wurde. Es kann jedoch vorkommen, daß in dem so entstandenen Entwurfsmodell, wegen der identifizierten Systemumgebung, Änderungen gegenüber dem Analysemodell vorgenommen werden müssen. Um jedoch die Nachvollziehbarkeit und Robustheit der Modelle beizubehalten, müssen diese Änderungen begründet und dokumentiert werden.

Für den in der Analysephase betrachteten Anwendungsfall `Anforderung generieren` (vgl. Kapitel 5) ist eine Eins-zu-Eins Übersetzung des Analysemodells in das Entwurfsmodell, wegen der im vorigen Schritt identifizierten Systemumgebung, nicht möglich. Dieses Aspekt wird im folgenden beschrieben.

Durch eine nähere Betrachtung der vorhandenen DHE-Software konnte festgestellt werden, daß die gesamte Datenverwaltung und -speicherung im OERR-System von den DHE-Managern übernommen werden kann. Die in der Analysephase modellierten Entitätsobjekte für die persistente Speicherung von Informationen (vgl. Abbildung 24) entsprechen weitgehend den Entitäten aus dem Datenmodell von DHE, das in ER-ähnlicher Notation in Abbildung 27 abgebildet ist. So entspricht beispielsweise das Entitätsobjekt *Patientendaten* zusammen mit dessen Attributen den Entitäten *Patient* und *Aufenthalt* in DHE. Das Entitätsobjekt *Fortbewegung* mit seinem Attribut *Fortbewegungsart* betrifft die Mobilität des Patienten zum Zeitpunkt der Durchführung der ausgewählten Einzelleistung und kann von DHE als Gesundheitsdatentyp in der Entität *AndereHD* gespeichert werden (vgl. Abbildung 27). Die *Kommunikationseinheiten* aus dem Analysemodell entprechen der Entität *Agent* in dem ER-Diagramm. Das Entitätsobjekt *Anforderung* kann von DHE in der Entität *Aktivität* gespeichert werden. Eine Anforderung ist die Sammlung mehrerer Einzelleistungen. Eine *Einzelleistung* entspricht der Entität *Aktivitätstyp*.

Das einzige Entitätsobjekt aus dem Analysemodell, welches keine Entsprechung in dem Datenmodell von DHE findet, ist *Patientenauswahl*. In diesem Objekt wird die aktuelle Patientenauswahl aus dem Belegungsplan gespeichert, welcher als Kontext für die weitere Ausführung des Anwendungsfalls übergeben wird. Diese Information wird lediglich transient gespeichert. DHE stellt dafür jedoch keine explizite Entität zur Verfügung, so daß das Entitätsobjekt *Patientenauswahl* innerhalb des OERR-Systems modelliert werden muß.

Zusammenfassend kann die Gegenüberstellung der Entitätsobjekte aus dem Analysemodell des Anwendungsfalles `Anforderung generieren`, mit den Entitäten des Datenmodells von DHE aus Abbildung 27, in der folgenden Tabelle 1 veranschaulicht werden:

Entitätsobjekte im Analysemodell	**Entitäten im Datenmodell von DHE**
Patientendaten	Patient (PA) Aufenthalt (TT)
Fortbewegung	AndereHD (TH)
Kommunikationseinheit	Agent (SP)
Anforderung	Aktivität (AA)

Entitätsobjekte im Analysemodell	Entitäten im Datenmodell von DHE
Einzelleistung	Aktivitätstyp (AY)
Patientenauswahl	-

Tabelle 1. Entitätsobjekte aus dem Analysemodell vs. Entitäten in DHE

Um die Informationen, die DHE speichert, mit dem zu entwickelten OERR-System austauschen zu können, wird ein neuer Akteur eingefügt, nämlich der Akteur ✍DHE. Laut Jacobson et al. (vgl. [JCJ+93]) modelliert ein Akteur das, was einen Informationsaustausch mit dem zu entwickelnden System benötigt. Akteure können sowohl menschliche Benutzer modellieren, aber auch andere Systeme (wie in diesem Fall DHE), die mit dem zu entwickelnden System kommunizieren müssen.

Eine weitere Abweichung des Entwurfsmodells vom Analysemodell liegt in der Modellierung des Kontrollobjektes *Sitzungsmanager*. Wie im Analysemodell beschrieben worden ist, koordiniert und kontrolliert der Sitzungsmanager den Aktivitätsfluß innerhalb einer Sitzung. Er sorgt einerseits für eine konsistente Speicherung der Daten und einer korrekt funktionierenden Kommunikation zwischen der anfordernden und der leistenden KE. Andererseits muß der Sitzungsmanager auch dafür sorgen, daß bei der Ausführung der verschiedenen Ereignisse innerhalb eines Anwendungsfalles die notwendigen Daten zur Verfügung gestellt bzw. zwischengespeichert werden (vgl. das Entitätsobjekt *Patientenauswahl*). Die Aufgabe der Datenspeicherung, -verwaltung und Kommunikation kann ✍DHE übernehmen. Im OERR-System bleibt demnach als Aufgabe nur noch die Modellierung einer eingeschränkten Version des im Analysemodell benutzten Sitzungsmanagers. Der Sitzungsmanager soll den Ereignisablauf im OERR-System steuern. Im Entwurfsmodell wird der entsprechende Block weiterhin *Sitzungsmanager* genannt, auch wenn er nur einem Teil des Kontrollobjektes *Sitzungsmanager* aus dem Analysemodell entspricht.

Was die Schnittstellenobjekte aus dem Analysemodell anbetrifft (vgl. Abbildung 24), lassen sich diese Eins-zu-Eins in das Entwurfsmodell übersetzen.

Zur Modellierung der Assoziationen im Entwurfsmodell werden die gleichen Regeln benutzt, die bereits bei der Modellierung dieser Assoziationen im Analysemodell angewendet wurden (vgl. Kapitel 5). Eine *communication*-Beziehung beispielsweise ermöglicht das Versenden von Stimuli zwischen zwei Entwurfsobjekten.

Für den in der Analysephase betrachteten Anwednungsfall `Anforderung generieren`, ergibt sich das in Abbildung 28 dargestellte Entwurfsmodell. Die Rechtecke in der Abbildung stellen jeweils ein Entwurfsobjekt dar. Das gesamte Entwurfsmodell des OERR-Systems befindet sich im Kapitel E3.1.

Die im Kapitel 6.3.1.1 betrachteten umgebungsabhängigen Aspekte führen zu einem erneuten Durchlaufen der Modelle aus der Analysephase. So ergibt die Iteration der Analysephase die Aktualisierung des Analysemodells um das Einfügen eines neuen Akteurs (✍DHE) und die Entfernung einiger Entitätsobjekte. Das so aktualisierte Analysemodell kann im Kapitel E2.4.2 nachgeschlagen werden.

6.3.1.3 Aufstellen von Interaktionsdiagrammen

Um die Reihenfolge der Ereignisse aus einem Anwendungsfall darzustellen, werden in einem letzten Schritt bei der Aufstellung des Entwurfsmodells *Interaktionsdiagramme* konstruiert. In einem Interaktionsdiagramm wird beschrieben, welche Arten von Stimuli in welcher Reihenfolge zwischen den Blöcken des Systems bei der Ausführung eines Anwendungsfalls versendet werden und welche Aktivitäten in den einzelnen Blöcken ausgeführt werden (vgl. [JCJ+93]).

Das Entwurfsmodell enthält für jeden Anwendungsfall ein Interaktionsdiagramm. Für den Anwendungsfall `Anforderung generieren` kann für das bessere Verständnis des Einsatzes von Interaktionsdiagrammen, ein Teil des entsprechenden Interaktionsdiagramms in Abbildung 29 betrachtet werden. Die restlichen Interaktionsdiagramme zu weiteren Anwendungsfälle befinden sich im Kapitel E3.1.

Jedes Entwurfsmodellobjekt, welches an der Ausführung dieses Anwendungsfalles beteiligt ist (vgl. Abbildung 28), wird durch einen Quadrat und einen senkrechten Balken im Interaktionsdiagramm dargestellt. Stimuli werden durch Pfeile repräsentiert, die vom Initiator einer Kommunikation zum Empfänger gerichtet sind. Die in Abbildung 29 verwendeten Stimuli-Bezeichner (über den Pfeilen) entsprechen den Funktionsaufrufen aus Visual Basic und werden bei der Beschreibung der Implementierungsphase zusammen mit dem dazugehörigen Quellcode näher beschrieben.

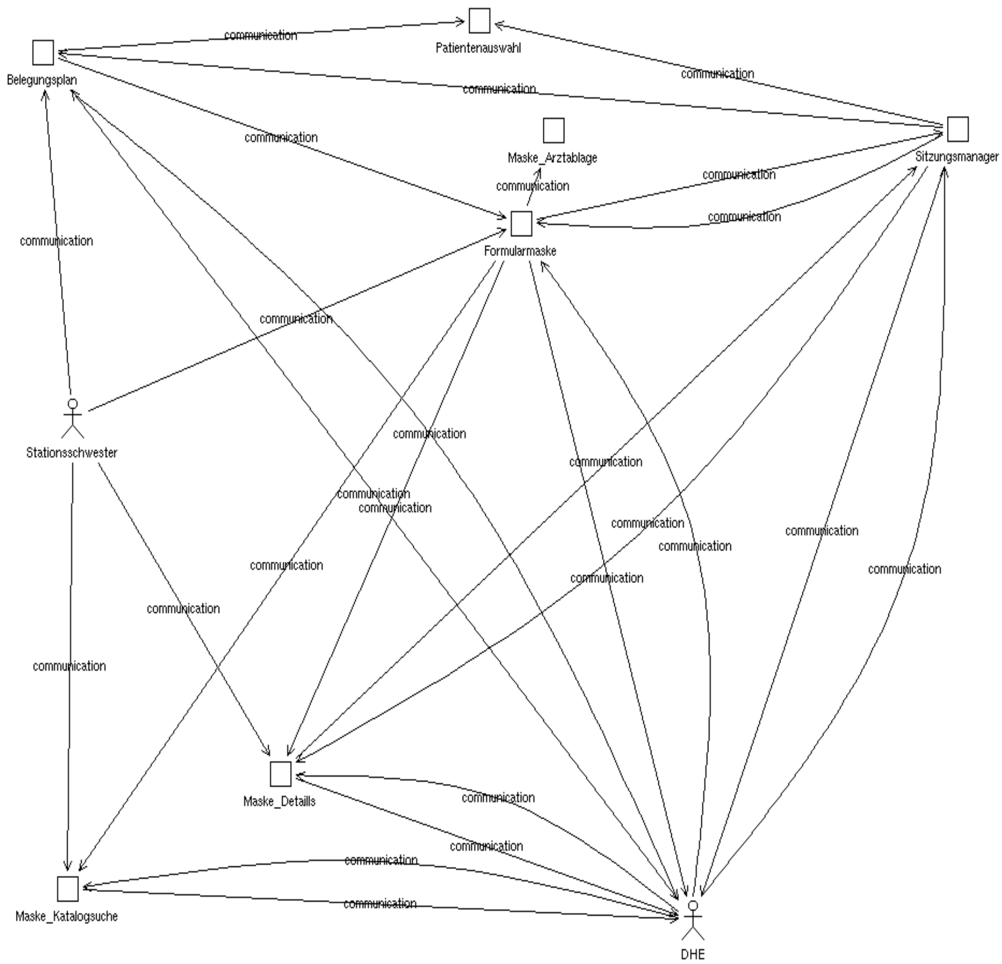

Abbildung 28: Der Anwendungsfall `Anforderung generieren` im Entwurfsmodell

Wenn in einem Objekt Aktionen ausgeführt werden sollen oder wenn auf eine direkte Reaktion eines Objektes, das ein Stimulus empfangen hat, gewartet wird, wird der entsprechende Balkenabschnitt weiß gekennzeichnet, ansonsten ist er schwarz. Ein zusätzlicher Balken wird für die *Systemgrenze* gezeichnet. Die Systemgrenze repräsentiert alles, was außerhalb des modellierten Software-Systems liegt. Für den in Abbildung 29 betrachteten Anwendungsfall besteht die Systemgrenze aus den Akteuren ✍Stationsschwester und ✍DHE (vgl. [JCJ+93]). Die Aktionen, die in den Blöcken ausgeführt werden, sind am linken Rand beschrieben. Die Beschreibung besteht aus strukturiertem Text.

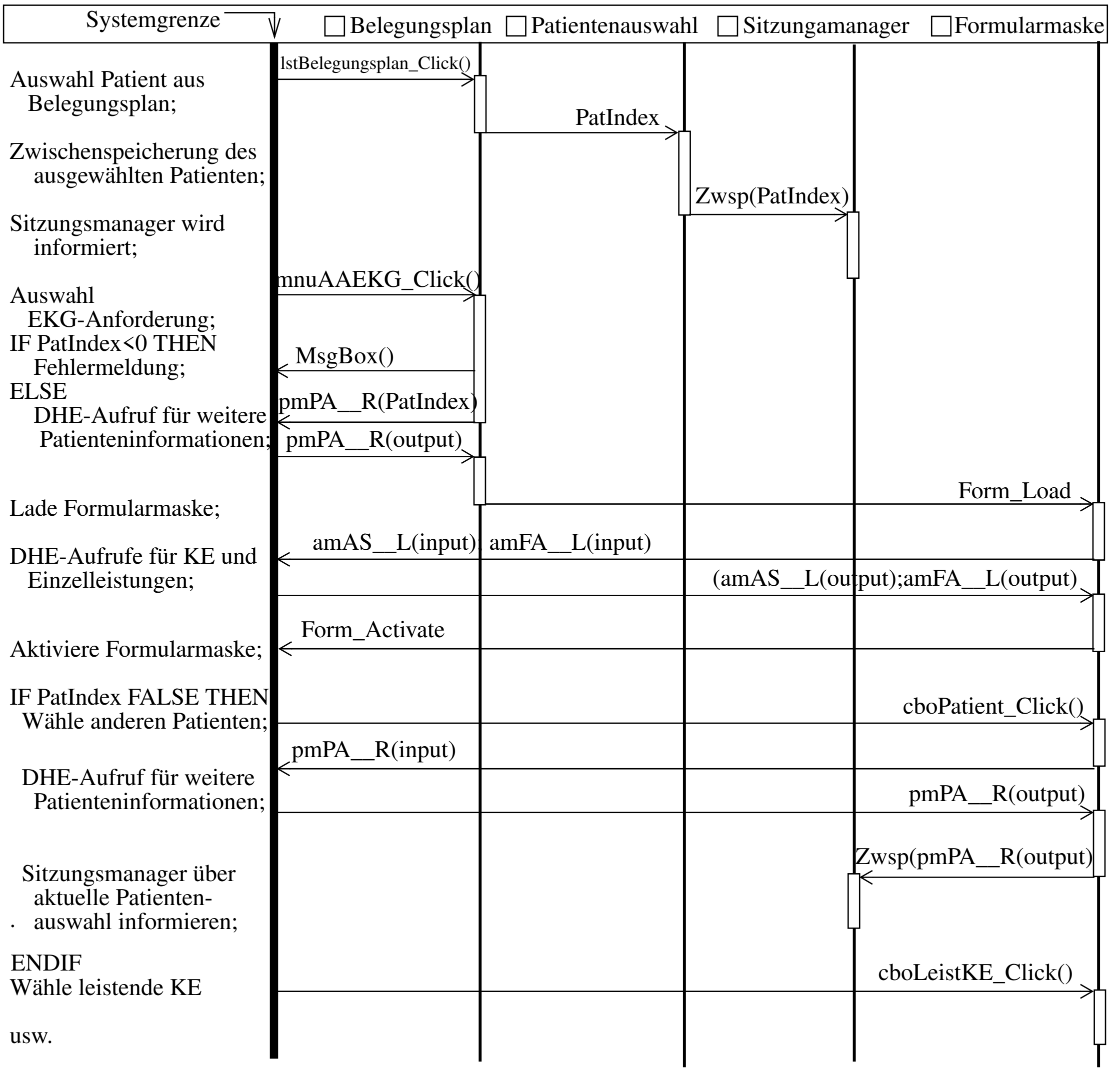

Abbildung 29: Teil des Interaktionsdiagramms des Anwendungsfalles `Anforderung generieren`

6.3.2 Das Implementierungsmodell

Das zweite aufzustellende Modell in der Konstruktionsphase ist das *Implementierungsmodell* (vgl. Abbildung 25). Dieses Modell, das auf der Basis des Entwurfsmodells entwickelt wird, besteht aus dem Code in der festgelegten Programmiersprache.

Im Implementierungsmodell wird festgelegt, wie die Übersetzung der Entwurfsobjekte und -Assoziationen in Begriffe und Eigenschaften der Programmiersprache stattfinden soll. Weiterhin werden die zu benutzenden Entwicklungsrichtlinien der entsprechenden Programmiersprache festgelegt. Erst danach kann mit dem tatsächlichen Schreiben des Quellcodes begonnen werden.

Folgende drei Schritte, die in den kommenden Kapiteln (6.3.2.1 bis 6.3.2.3) näher erläutert werden, werden in dieser Arbeit beim Aufstellen des Implementierungsmodells durchgeführt:

- Aufbau des DHE-Clients;
- Übersetzung der Entwurfskonzepte in Implementierungskonzepte;
- Implementierung des OERR-Systems in Visual Basic.

6.3.2.1 Aufbau des DHE-Clients

Das Implementierungsmodell entsteht auf der DHE-Client-Seite (PC) (vgl. Kapitel 3.1.3.2). Im Entwurfsmodell wurden die hardware-technischen Rahmenbedingungen des DHE-Clients erläutert (vgl. Kapitel 6.3.1.1). Software-technisch besteht der DHE-Client aus den folgenden Komponenten (vgl. Abbildung 30):

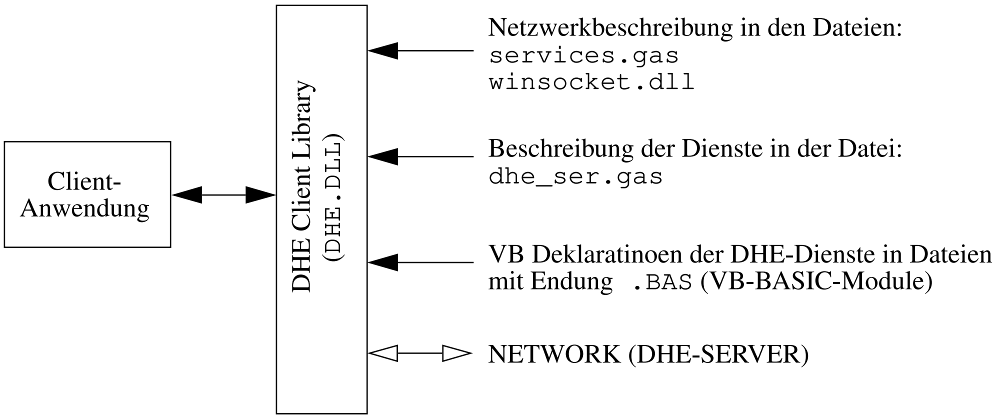

Abbildung 30: Komponenten des DHE-Client (vgl. [Fer96a])

Die *DHE Client Library* (`dhe.dll` in MSWindows) interagiert mit der Client-Anwendung und ist verantwortlich für die Verwaltung aller Interaktionen mit dem DHE-Server zur Ausführung der angeforderten Dienste. Diese Bibliothek wird das erste Mal aufgrund der Inhalte der Datei `dhe_ser.gas`, die sich ebenfalls auf der Client-Seite befindet, initialisiert. Diese Datei enthält eine codierte Beschreibung der DHE-Dienste. Für die Netzwerkkonfiguration ruft die Datei `dhe.dll` Informationen aus der Datei `services.gas` auf. Diese Datei befindet sich sowohl auf der Client-Seite (`services.gas`) als auch auf der Server-Seite (in `/etc/services`). Sie enthält die Adresse und die Portnummer des DHE-Servers.

Die Kommunikation zwischen dem Client und dem Server basiert auf dem TCP/IP-Protokoll und auf „Sockets“, die in einem Windows-basierten System vorhanden sind. Für Windows Version 3.1 ist es deshalb notwendig, daß die Datei `WinSocket.dll` im System zur Verfügung steht.

Zusammen mit der Datei `dhe.dll` für Windows Version 3.1 wurden auch eine Reihe von Visual Basic BASIC-Module geliefert, die die Typdeklarationen der DHE-Datenstrukturen und -Dienste für die Programmierung in Visual Basic enthalten. Abbildung 31 zeigt einen Ausschnitt aus den BASIC-Modulen `0dhe_.bas`, `pm_s.bas` und `pm_api.bas`, so wie sie von DHE, mittels des „GAS“-Werkzeuges der Firma GESI, generiert werden. Dieses Beispiel soll dazu dienen, den im Implementierungsmodell weiter exemplarisch vorgestellten Quellcode zu verstehen.

Die Zeilen aus Abbildung 31, die mit einem Anführungszeichen (') beginnen sind in VB3 Kommentare. Die Funktionen, die in dem Modul `pm_api.bas` deklariert sind (hier: `pmPA__R`) sind DHE-API-Aufrufe und sind wie folgt bezeichnet (vgl. [Fer96a]):

`<manager_prefix><data_object_id>__<type_of_service>`

Die ersten zwei Buchstaben eines API identifizieren den Manager des Dienstes (`<manager_prefix>`). Folgende Abkürzungen werden für die Manager benutzt:

- `dh` entspricht den Systemfunktionen, die das gesamte DHE-Verhalten koordinieren;
- `pm` entspricht dem Patient Manager;
- `am` entspricht dem Act Manager;
- `hm` entspricht dem Healthdata Manager;
- `um` entspricht dem Users and Authorisation Manager;
- `rm` entspricht dem Ressource Manager.

```
` -------------------------------------------------------
` DHE-BASIC-Modul 0dhe_.bas
` Specification file with data types
` DHE - Distributed Healthcare Environment Version 1.00a
` generated on 04/10/1996 12:44:56
` by GAS (C) GESI Version 0.9
`
`*******************************************************
` Size of data types

Global Const DEF_LONG_CODE = 17
Global Const DEF_SYSCOD = 13
`usw.
`------------------------------------------------------
` DHE-BASIC-Modul pm_s.bas
` Specification file with data types
` DHE - Distributed Healthcare Environment Version 1.00a
` generated on 04/10/1996 12:46:56
` by GAS (C) GESI Version 0.9
`
`******************************************************
` Structured Data Types for object: Patient Manager
` Structure containing one full instance of object: Patients

Type pmPA__OR
 pa__icode As String *DEF_SYSCODE
 pa_iperm  As String *DEF_LONG_CODE
 pa_inato  As String *DEF_LONG_CODE
 pa_iregi  As String *11
 pa_fname  As String *DEF_PAT_NAME
 pa_lname  As String *DEF_PAT_NAME
 pa_birth  As String *DEF_DATE
 `usw.
End Type
` Structure containing the selection parameters for retrieving
` one instance of object: Patient
Type pmPA__IR
 pa__icode As String *DEF_SYSCOD
End Type
`------------------------------------------------------
` DHE-BASIC-Modul pm_api.bas
` Function prototypes
` DHE-Distributed Healthcare Environment
` generated on 04/10/1996 12:47:56
` by GAS (C) GESI Version 0.9
`
`******************************************************
` For object: Patient Manager Version 1.00a

Declare Function pmPA__R Lib „dhe.dll“ (ByVal i_env As String,
               is_select As pmPA__IR, os_instance As pmPA__OR,
               ByVal o_servmsg As String) As Integer
```

Abbildung 31: VB3-Deklarationen für DHE-Datenstrukturen und -Dienste

Die nächsten zwei Buchstaben des Aufrufes betreffen das involvierte Datenobjekt (<data_object_id>). Diese Buchstaben sind im Datenmodell von DHE jeder Entität zugeordnet. Beispiel: `PA` Patient, `CC` Aufenthalt, `AA` Aktivität (vgl. Abbildung 27).

Der letzte Buchstabe bestimmt den Rückgabetyp des API-Aufrufes (<type_of_service>). Folgende Möglichkeiten stehen zur Verfügung:

- `L` - der API-Aufruf gibt eine Liste zurück;
- `R` - der API-Aufruf gibt einen spezifizierten Datensatz zurück;
- `M` - der API-Aufruf erlaubt die Eingabe, Änderung und Löschen eines spezifizierten Datensatzes.

Folgendes Beispiel eines API-Aufrufes soll die oben genannten Begriffe besser verdeutlichen:

API-Aufruf: `pmPA__R(...,is_select,...)`

Beschreibung: diese Funktion ist im Patient Manager enthalten (`pm`) und gibt die Instanz zurück (`R`), die zu der im Aufrufeparamter enthaltenen eindeutigen ID eines Patienten (`PA`) gehört.

Für Einzelheiten zu weiteren API-Aufrufe sei auf [Fer96a] verwiesen.

Letztendlich besteht der DHE-Client aus dem VB3-Quellcode des entwickelten OERR-Systems (vgl. Abbildung 30).

6.3.2.2 Übersetzung der Entwurfskonzepte in Implementierungskonzepte

Ein zweiter Schritt beim Aufstellen des Implementierungsmodells besteht in der Übersetzung der Entwurfsobjekte. Während eine objektorientierte Analyse und Systementwurf durchgeführt wurde, wird weiter nach [JCJ+93] eine nicht-objektorientierte Implementierung angewendet. Bei VB3 handelt es sich, so wie im Kapitel 2.2.4 erwähnt wurde, um eine ereignisorientierte Programmiersprache. Visual Basic bietet jedoch Konzepte an, die eine objekt-basierte Programmierung zulassen. In Tabelle 2 kann die Übersetzung der Konzepte, die nach [JCJ+93] von einer objektorientierten Programmiersprache gefordert werden, in die Konzepte, die Visual Basic unterstützt, betrachtet werden.

Analyse	Entwurf	VB-Implementierung
Entitätsobjekt	Block	Variable
Schnittstellenobjekt	Block	Formular
Kontrollobjekt	Block	Variable
Objektverhalten	Methode	Ereignisprozedur
„communication“	„communication“	Ereignisprozedur-Aufruf
Interaktion	Stimulus	Ereignis
Anwendungsfall	Anwendungsfall	Sequenz von Prozeduraufrufen

Tabelle 2. Übersetzung der Analyse- und Entwurfskonzepte in Visual Basic Konzepte

Die Konzepte von Visual Basic und die in der Tabelle 2 verwendeten Begriffe werden im Kapitel 6.3.2.3 anhand einer kurzen Beschreibung erläutert.

Der Entwurf von Dialogen sowie die Programme aus dem Implementierungsmodell, werden gemäß den Richtlinien des ROKD-Styleguides und der Visual Basic-Entwicklungsrichtlinien gestaltet (vgl. [ROKD97]).

6.3.2.3 Implementierung des OERR-Systems

Der letzte Schritt bei der Aufstellung des Implementierungsmodells ist die Implementierung selbst. Für die Implementierung des OERR-Systems wird in Visual Basic ein Projekt namens `OERR.MAK` erzeugt. Diese Projektdatei enthält die Namen der einzelnen Bestandteile (Formulare, Module, Custom Control Dateien).

Das *Formular*, also eine Bildschirmmaske, zählt zu den wichtigsten Bestandteile einer VB3-Anwendung. Sie enthält nicht nur die einzelnen Steuerelemente mit deren Eigenschaften (Name, Größe, Farbe etc.), sondern auch den zugehörigen Programmcode. Die *BASIC-Module* sind eine weitere Komponente von VB3 und enthalten den Programmcode, der nicht direkt einem Ereignis oder einem Objekt zuzuordnen ist. Module speichern Prozeduren und Deklarationen in einer Datei, die dann global der gesamten Anwendung zur Verfügung stehen. Die *Custom Control Dateien* werden ebenfalls von VB3 zur Verfügung gestellt und stellen der VB3-Anwendung die externen Steuerelemente zur Verfügung.

Es wird in VB3 zwischen *Ereignisprozeduren* und *allgemeine Prozeduren* unterschieden. *Ereignisprozeduren* sind grundsätzlich, wie bereits der Name sagt, bestimmten Ereignissen zugeordnet, werden also nur bei Eintritt dieses Ereignisses ausgeführt. Die Ereignisse werden immer an Steuerelemente gebunden. Mit den Ereignissen zeigen die Steuerelemente schließlich Aktionen des Benutzers an, wie beispielsweise Klicken oder Doppelklicken, Ziehen und Ablegen, Ändern des Inhaltes usw. *Allgemeine Prozeduren* werden normalerweise nicht durch ein Ereignis aufgerufen, sondern erst dann, wenn ein Teil der VB3-Anwendung sie explizit aufruft (vgl. [MR93]) . Sie können auch aus Ereignisprozeduren heraus aufgerufen werden und sind vor allem hier sinnvoll, wenn bei mehreren verschiedenen Ereignissen eine bestimmte Reaktion stattfinden soll.

Bei der Aufstellung des Projektes `OERR.MAK` werden fünf Schritte angewendet, die sich auf das Konzept der ereignis-orientierten Programmierung stützen. Diese Schritte sind:

Schritt 7. Aufstellen der Applikations-Oberfläche mit Hilfe der einzelnen Objekte.

Schritt 8. Benennen der einzelnen Objekte.

Schritt 9. Setzten der Eigenschaften der Objekte in einem Dialogfenster.

Schritt 10. Schreiben des Programmcodes und Anbinden des Codes an die Objekte.

Schritt 11. Aufstellen globaler Routinen und Module.

Bei der ereignis-orientierten Programmierung wird als erstes die Oberfläche konstruiert, es werden dort Objekte identifiziert, Ereignisse festgelegt und dann der entsprechende Programmcode hinzugefügt. Hinzu kommen noch globale Routinen und Module. Diese Vorgehensweise lehnt sich stark an die Programmierung mit Hilfe von GUI-Bildern an.

Es wird in VB3 unter zwei Entwicklungssichten auf die Steuerelemente unterschieden: die *Entwurfs-* und die *Laufzeitsicht*. In der *Entwurfssicht* werden einzelne Objekte definiert und mit Eigenschaften und Quellcode versehen. Durch die schnellen und gleichzeitig komfortablen Entwurfsmöglichkeiten bietet sich VB3 besonders für das Prototyping von Anwendungen an. In der Entwurfssicht können die Namen der einzelnen Objekte auf die Benutzungsoberfläche gesehen werden, so wie diese im Quellcode benutzt werden. Die *Laufzeitsicht* der VB3-Programmiersprache ist die Ausführung des Projektes. Dabei wird das Formular geladen, welches als Startformular im Projekt definiert wurde. Die Namen der Objekte, die in der Entwurfssicht auf der Benutzungsoberfläche noch sichtbar waren, verschwinden bei der Laufzeit.

Das `OERR.MAK`-Projekt besteht aus elf Formularen, 17 BASIC-Modulen und drei Custom Control Dateien. Von den 17 BASIC-Modulen sind 16 aus DHE übernommen. Das Modul *Global.bas* enthält projektinterne globale Deklarationen von Variablen und Funktionen. Die DHE-BASIC-Module stellen Schnittstellen zu den Manager und Dienste der DHE-Komponente zur Verfügung. Ein Überblick über das gesamte `OERR.MAK`-Projekt bietet Tabelle 3. Folgende Herkunft der Komponenten wird unterschieden: OERR - selbst geschrieben; DHE - von DHE übernommen; Windows - Komponente von MS Windows.

Dateiname	**VB-Objekt**	**Herkunft**
FRMANFDE.FRM	frmDetails	OERR
FRMANFGE.FRM	frmAnfGen	OERR
FRMANFVE.FRM	frmAnfVerif	OERR
FRMARZTA.FRM	frmArztablage	OERR
FRMBELEG.FRM	frmAnforderndeKE	OERR
FRMBEST.FRM	frmBestätigen	OERR
FRMERGEB.FRM	frmErgebnisablage	OERR
FRMLEIST.FRM	frmLeistendeKE	OERR
FRMLOGIN.FRM	frmLogin	OERR
FRMSUKAT.FRM	frmKatalogsuche	OERR
FRMSCHWE.FRM	frmSchwester	OERR
FRMBERIC.FRM	frmBerichten	OERR
FRMANSEH.FRM	frmAnsehen	OERR
ODHE_.BAS	BASIC-Modul	DHE
AM_API.BAS	BASIC-Modul	DHE
AM_S.BAS	BASIC-Modul	DHE
AM_S1.BAS	BASIC-Modul	DHE

Dateiname	VB-Objekt	Herkunft
AM_S2.BAS	BASIC-Modul	DHE
AM_S3.BAS	BASIC-Modul	DHE
AM_S4.BAS	BASIC-Modul	DHE
AM_S5.BAS	BASIC-Modul	DHE
DH_API.BAS	BASIC-Modul	DHE
DH_S1.BAS	BASIC-Modul	DHE
DHE_E.BAS	BASIC-Modul	DHE
DHE_S.BAS	BASIC-Modul	DHE
DHE_TBC.BAS	BASIC-Modul	DHE
PM_API.BAS	BASIC-Modul	DHE
PM_S.BAS	BASIC-Modul	DHE
PM_S1.BAS	BASIC-Modul	DHE
GLOBAL.BAS	BASIC-Modul	OERR
ANIBUTTON.VBX	Control Object	Windows
GRID.VBX	Control Object	Windows
THREED.VBX	Control Object	Windows

Tabelle 3. Die Bestandteile des `OERR.MAK`-Projektes

Durchführung der Schritte 1, 2, 3

Nach der Beschreibung der Vorgehensweise bei der Aufstellung des `OERR.MAK`-Projektes werden in den ersten drei Schritte für den Anwendungsfall `Anforderung generieren` folgende Entwurfsobjekte (vgl. auch Abbildung 28) in Visual Basic Formulare übersetzt und mit Eigenschaften versehen (vgl. Tabelle 4).

Entwurfsobjekt	VB-Formular
Belegungsplan	frmAnforderndeKE
Formularmaske	frmAnfGen
Maske_Details	frmDetails
Maske_Katalogsuche	frmKatalogsuche
Arztablage	frmArztablage
Sitzungsmanager	-
Patientenauswahl	-

Tabelle 4. Bestandteile der Applikations-Oberfläche in VB3

Jedes Entwurfsobjekt aus dem Entwurfsmodell entspricht einem Formular (*frm*) in Visual Basic. Die einzigen Entwurfsobjekte, die keine Entsprechung in einem Visual Basic Formular haben, sind der *Sitzungsmanager* und die *Patientenauswahl*. Der *Sitzungsmanager* ist, so wie im Entwurfsmodell beschrieben wurde, für die Zwischenspeicherung von Daten zuständig, die für die weitere Ausführung des Anwendungsfalls notwendig sind, und die von DHE nicht gespeichert werden können. Die Zwischenspeicherung von Daten in dem `OERR.MAK`-Projekt erfolgt durch den Einsatz globaler Variablen. Im Quellcode wird an der entsprechenden Stelle, wo der Einsatz des Sitzungsmanagers deutlich wird, dies entsprechend kommentiert.

Das Entwurfsobjekt *Patientenauswahl* speichert Informationen über den im Belegungsplan aktuell ausgewählten Patienten und gibt diese an den Sitzungsmanager weiter. Dafür werden in VB3 für die entsprechenden Informationen Variablen deklariert und mit dem entsprechenden Wert belegt.

Jedes Formular enthält verschiedene Objekte, die mit Eigenschaften versehen sind. Die Namen der Objekte werden im Quellcode benutzt und sind nur in der Entwurfssicht des VB-Projektes auf der Formularoberfläche sichtbar. Abbildung 32 zeigt beispielhaft die Formularoberfläche des Formulars `frmAnfGen` (Formularmaske) in der Entwurfssicht. Dabei können die Namen der benutzten Objekte identifiziert werden. Formulare zu weiteren Anwendungsfällen des OERR-Systems, mit den dazugehörigen Objekte und Eigenschaften, können dem Kapitel E3.2.3 entnommen werden.

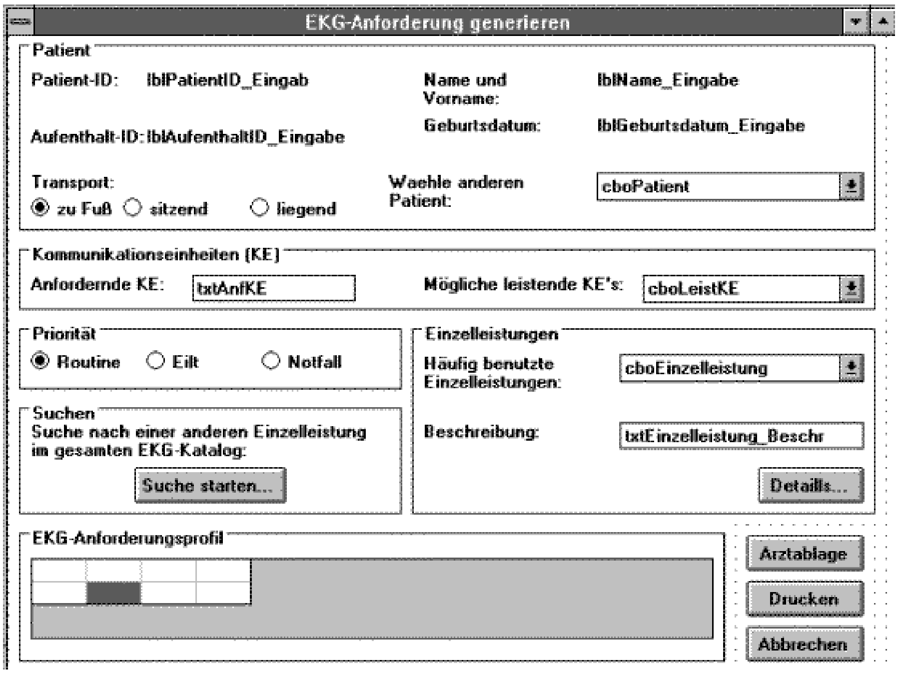

Abbildung 32: Die VB-Entwurfssicht des Formulars `frmAnfGen`

Durchführung des 4. Schrittes

Der vierte Schritt in der Aufstellung des `OERR.MAK`-Projektes umfaßt das tatsächliche Schreiben des Programmcodes und dessen Anbindung an die Objekte und Formulare. Die Ereignisprozeduren entsprechen den Operationen aus dem Interaktionsdiagramm aus Abbildung 29 und werden dann ausgeführt, wenn ein bestimmtes Ereignis eintrifft. Die für den Anwendungsfall `Anforderung generieren` implementierten Ereignisprozeduren, mit ihren Formular-Zugehörigkeiten und Beschreibungen, können der Tabelle 5 entnommen werden.

Name	Formular	Beschreibung
`lstBelegungsplan_Click()`	`frmAnforderndeKE`	Der Benutzer wählt einen Patient aus dem *Belegungsplan* aus.
`mnuAAEKG_Click()`	`frmAnforderndeKE`	Funktion `Anforderung generieren` aus dem Menü wird aktiviert.
`Form_Load()`	`frmAnfGen`	*Formularmaske* (`frmAnfGen`) wird mit den entsprechenden Daten geladen.
`Form_Activate()`	`frmAnfGen`	*Formularmaske* wird aktiviert.
`cboPatient_Click()`	`frmAnfGen`	Daten eines ausgewählten Patienten werden in der *Formularmaske* geladen.
`cboLeistKE_Click()`	`frmAnfGen`	Der Benutzer wählt eine leistendende KE aus.
`cboEinzelleistung_Click()`	`frmAnfGen`	Die gewünschte Einzelleistung aus dem Katalog wird ausgewählt.

Name	Formular	Beschreibung
`cmdDetails_Click()`	`frmAnfGen`	Eingabe von weiteren Details zu einer ausgewählten Einzelleistung wird gestartet.
`Form_Load()`	`frmDetails`	*Maske_Details* (`frmDetails`) wird mit den entsprechenden Daten geladen.
`grdUhrzeit_Click()`	`frmDetails`	Ein Termin für die Durchführung der angeforderten Einzelleistung wird ausgewählt.
`cmdOK_Click()`	`frmDetails`	Eingaben aus *Maske_Details* akzeptieren.
`cmdAbbr_Click()`	`frmDetails`	Das Formular `frmDetails` wird geschlossen.
`cmdKatalogsuche_Click()`	`frmAnfGen`	Katalogsuche wird gestartet.
`Form_Load()`	`frmKatalogsuche`	*Maske_Katalogsuche* (`frmKatalogsuche`) wird mit den entsprechenden Daten geladen.
`cboKatalog_Click()`	`frmKatalogsuche`	Die gewünschte Einzelleistung aus dem Katalog wird ausgewählt.
`cmdOK_Click()`	`frmKatalogsuche`	Die aus dem Katalog ausgewählte Einzelleistung wird zwecks Weiterverarbeitung aktiviert.

Tabelle 5. Ereignisprozeduren des Anwendungsfalls `Anforderung generieren`

Die DHE-API-Aufrufe aus dem Interaktionsdiagramm aus Abbildung 29 (Beispiel: `pmPA__R`) werden innerhalb der oben beschriebenen Ereignisprozeduren als Funktionen aufgerufen. Folgendes Beispiel soll dies veranschaulichen:

```
Sub mnuAAEKG_Click()
        .......
        ret = pmPA__R(i_env,is_patient,os_patient,o_servmsg)
        ......
End Sub
```

In diesem Beispiel wird in der Ereignisprozedur `mnuAAEKG_Click()` der DHE-Dienst `pmPA__R` mit den entsprechenden Parameter aufgerufen.

Der einzige Prozeduraufruf aus dem Interaktionsdiagramm, der nicht als expliziter Prozeduraufruf in VB3 umgesetzt wird, ist `Zwsp(...)`, der für die Zwischenspeicherung zuständig ist. Die Zwischenspeicherung von Daten durch den Sitzungsmanager erfolgt in Form einer Variablenzuweisung.

Durchführung des 5. Schrittes

Der letzte Schritt ist die Implementierung globaler Routinen und Modulen. Die BASIC-Module, die für das Ansprechen der Datei `dhe.dll` zuständig sind, sind mit der DHE-Software zusammen geliefert worden und in das `OERR.MAK`-Projekt eingebunden (vgl. Tabelle 4). Außerdem wird das BASIC-Modul *Global.bas* implementiert, das sowohl globale Variablendeklarationen als auch globale Funktionen enthält. So wird beispielsweise eine Funktion `DatumFormat(Datum)` implementiert, welches das im amerikanischen Format von DHE ausgegebene Datum (MM/TT/JJ) in das deutsche Format (TT/MM/JJJJ) umwandelt.

Abschließend soll der folgende Quellcode der Ereignisprozedur `mnuAAEKG_Click()` die im Implementierungsmodell beschriebenen Konzepte, so wie diese im VB3 umgesetzt wurden, verdeutlichen. Die Ereignisprozedur `mnuAAEKG_Click()` wird dann aufgerufen, wenn das Ereignis `Click` auf dem Objekt `mnuAAEKG` eintrifft, welches dem Menüpunkt `Anforderung->Anforderung generieren->EKG` aus der Funktionsleiste des Belegungsplans entspricht. Die Ausführung dieser Ereignisprozedur öffnet die Formularmaske `frmAnfGen`, nachdem ein Patient aus dem Belegungsplan ausgewählt wurde, und füllt die Felder mit den entsprechenden Daten aus. Das gesamte Quellcode des implementierten OERR-Systems befindet sich im Kapitel E3.2.

```
'*****************************************************************
'Autorin: Ruxandra Privighitorita
'Name der Ereignisprozedur: mnuAAEKG_Click()
'Beschreibung der Ereignisprozedur: Diese Prozedur wird aufgerufen,
'       wenn der Benutzer den Menüpunkt "Anforderung->Anforderung
'       generieren->EKG" aus der Funktionsleiste angeklickt hat, und
'       öffnet die Formularmaske frmAnfGen mit den entsprechenden
'       Daten.
'Aufrufparameter: keine
'Rückgabewerte: keine
'*****************************************************************
Sub mnuAAEKG_Click ()
'Variablendeklaration

Dim PatIndex As Integer     ' Patientenauswahl-Variable
Dim is_patient As pmPA__IR   ' DHE-Input-Variable für den Dienst pmPA
Dim os_patient As pmPA__OR   ' DHE-Output-Variable für den Dienst pmPA
Dim ret As Integer          ' DHE-Service-Aufruf-Variable
Dim i_env As String * DEF_ENVIR  ' DHE-input (String der Laenge 101)
Dim o_servmsg As String * DEF_SERVER_MSG ' DHE-output (Output message)
Dim i As Integer                         ' Hilfsvariable

'Ein ausgewählter Eintrag (ListIndex) aus der Liste Belegungsplan
'(lstBelegungsplan) wird der Patientenauswahl (PatIndex) zugewiesen:

   PatIndex = lstBelegungsplan.ListIndex

   'Falls keine Auswahl seitens des Benutzers getroffen wurde:

   If PatIndex < 0 Then
    'Menüpunkt Anforderung->Anforderung generieren bleibt inaktiv:
    mnuAAnfGen.Enabled = False
   Else
    'Falls der Benutzer einen Patienten aus der Liste ausgewählt hat,
    'wird der Menüpunkt Anforderung->Anforderung generieren aktiviert:
    mnuAAnfGen.Enabled = True
    'Dem Input-Paramter pa__icode (d.h. eindeutige ID einer
    'DHE-Instanz), wird die aus dem Eintrag PatIndex aktuell
    'ausgewählte Patienten-ID (tt_iperm) zugewiesen. PatIndex enthält
    'außer der Patienten-ID noch Name und Vorname des Patienten, sowie
    'Aufenthalts-ID.
    'Weiterhin wird der Sitzungsmanager  über die aktuelle
    'Patientenauswahl informiert und weist der Input-Variable
    'is_patient.pa__icode die Patienten-ID aus PatIndex zu, um diesen
    'Parameter beim Aufruf des DHE-Services pmPA__R zu benutzen. Dadurch
    'können weitere Informationen, die nicht in der Form
    'frmAnforderndeKE dargestellt sind, wie bspw. das
    'Geburtsdatum des Patienten, Informationen die Output-Parameter des
    'entsprechenden DHE-Dienstes sind, in anderen Formulare (wie bspw.
    'frmAnfGen) geladen werden:
    is_patient.pa__icode = osl_list(PatIndex).tt_iperm

   'Aufruf des DHE-Dienstes pmPA__R mit Input-Feld
   'is_patient.pa__icode:
    ret = pmPA__R(i_env, is_patient, os_patient, o_servmsg)

    If ret = SUCCESS Then
       'Label lblName_Eingabe des Formulars frmAnfGen wird mit dem
       'Namen und Vornamen des ausgewählten Patienten geladen:
       frmAnfGen.lblName_Eingabe = ClearString(osl_list(PatIndex).pa_fname)+""+
           ClearString(osl_list(PatIndex).pa_lname)
       If Len(ClearString(os_patient.pa_birth)) > 0 Then
           'Label lblGeburtsdatum_Eingabe des Formulars frmAnfGen wird
           'mit dem als Output des DHE-Aufrufes pmPA__R ausgegebenes
          'Geburtsdatum(pa_birth) geladen und das Datumsformat
          'geändert (Funktion: DatumFormat)
          frmAnfGen.lblGeburtsdatum_Eingabe =
```

```
                                         DatumFormat(ClearString(os_patient.pa_birth))
        End If
        'Inhalt des ComboBoxes aus dem Formular frmAnfGen wird mit Null initialisiert:
        frmAnfGen.cboPatient.Clear
        'Textfeld txtAnfKE wird mit dem Kürzel der anfordernden KE geladen (sp_shnam)
        frmAnfGen.txtAnfKE = ClearString(osl_list(PatIndex).sp_shnam)
        'Für jeden Eintrag aus dem Belegungsplans
        For i = 0 To o_nelem - 1
            'Lade in die ComboBox cboPatient aus dem Formular frmAnfGen
            'Name und Vorname der Patienten aus dem Belegungsplan:
            frmAnfGen.cboPatient.AddItem ClearString(osl_list(i).pa_fname)+""+
              ClearString(osl_list(i).pa_lname)
        Next i
        'Aktueller Eintrag in dem ComboBox cboPatient muß mit dem
        'aktuell ausgewählten Patienten übereinstimmen:
        frmAnfGen.cboPatient.ListIndex = PatIndex
        'Zeige das Formular frmAnfGen an:
        frmAnfGen.Show

     'Falls DHE-Aufruf fehlerhaft:
     Else
         'Fehlermeldung:
         MsgBox ("DHE-Fehler Nr. " + Str(ret) + " Bitte wenden Sie sich an Ihren
                 System-Administrator!")
     End If
    End If
End Sub
```

Darüberhinaus wurden in dem BASIC-Modul *Global.bas* folgende beiden globale Variablen deklariert, die nicht nur in der oben beschriebenen Ereignisprozedur `mnuAAEKG_Click()`, sondern auch in weiteren Prozeduren (vgl. Kapitel E3.2), benutzt werden:

```
Global osl_list() As pmTT__OL
Global o_nelem As Integer
```

Die Objektbezeichnungen, die in dem Quellcode vorkommen (*Beispiel: frmAnfGen.lblName_Eingabe*), können in dem entsprechenden Formular in der Entwurfssicht betrachtet werden. In dem obigen Beispiel handelt es sich um das Formular `frmAnfGen` und das Objekt vom Typ Label `lblName_Eingabe`. Diese Bezeichnungen können in Abbildung 32 wiedergefunden werden.

Die Funktionen `ClearString` und `DatumFormat` wurden ebenfalls global in *Global.bas* definiert. Die Funktion `ClearString` liest ein String und löscht alle leeren Zeichen von Anfang und Ende des String. Die Funktion `DatumFormat` konvertiert das DHE-Datumsformat MM/TT/JJ in das deutsche Datumsformat TT/MM/JJJJ. Die Datentypen `pmPA__IR`, `pmPA__OR`, `pmTT__OL`, `*DEF_ENVIR`, `*DEF_SERVER_MSG` wurden in diesem Kapitel angesprochen und sind in den DHE-BASIC-Modulen `pm_s.bas` und `0dhe_.bas` definiert (vgl. Abbildung 31).

6.4 Validierungstätigkeiten in der Konstruktionsphase

In der Konstruktionsphase verfolgen die Validierungstätigkeiten unter anderem folgende Ziele:

- Feststellen und Bewerten des jeweiligen Zustandes des Entwurfs;
- Aufdecken von Fehlern und Widersprüchlichkeiten (beispielsweise Widerspruch zwischen Spezifikation und Entwurf etc.);
- Überprüfung des Quellcodes auf Übereinstimmung mit dem Entwurf;
- Überprüfung des Quellcodes auf Einhaltung von Programmierrichtlinien.

Daraus und aus den im SPARDAT-Modell beschriebenen Qualitätszielen (vgl. Kapitel E1) ergeben sich die im folgenden Kapitel aufgezählten Qualitätseigenschaften und -merkmale, die in der Konstruktionsphase zu bewerten sind.

6.4.1 Validierungsziele und Auswertungsmethoden

Folgende Qualitätseigenschaften und -merkmale aus dem SPARDAT-Qualitätsmodell werden in der Konstruktionsphase validiert:

Qualitätseigenschaft	Qualitätsmerkmal
Funktionsabdeckung	• Funktionale Vollständigkeit • Vollständigkeit der Dokumentation
Widerspruchsfreiheit	• Widerspruch zwischen Funktionalitäten, die die Funktionen erfüllen • Widerspruch zwischen dynamischem Produktverhalten und Dokumentation
Korrektheit	• Modulbezogene Konsistenz
Sicherheit	• Sicherheitskontrollen / Zugriffsschutz • Besondere Sicherheitsvorkehrungen durch das System • Sicherungsmaßnahmen • Datenschutzmaßnahmen
Robustheit	• Plausibilitätskontrollen und Fehlerbehandlungen
Handhabbarkeit	• Gestaltung der Benutzungsschnittstelle • Rationalität
Effektivität	• Betriebsorganisatorische Aufgabenerfüllung
Einheitlichkeit	• Berücksichtigte Standards, Normen und Richtlinien • Einheitliches Systemverhalten
Änderbarkeit	• Grad der Modularisierung
Erweiterbarkeit	• Flexibilität der Struktur
Portabilität	• Maschinen- und Betriebssystemunabhängigkeit
Wiederverwendbarkeit	• Allgemeingültigkeit

Die zu diesen Qualitätsmerkmalen dazugehörigen Auswertungsmethoden sind eine Teilmenge der im Kapitel E1 beschriebenen Auswertungsmethoden und sind im Kapitel E3.3 für die Konstruktionsphase zusammengefaßt.

6.4.2 Durchführung der Validierung

Die im vorigen Kapitel festgelegten Qualitätseigenschaften und -merkmale werden hier validiert. So werden viele der aufgezählten Qualitätsmerkmale aus zwei Sichten betrachtet: bezogen einerseits auf das OERR-System und andererseits auf die DHE-Software (vgl. auch Kapitel 4.2).

Es wird in diesem Kapitel nicht auf die Validierung aller angeführten Qualitätsmerkmale näher eingegangen, sondern lediglich nur auf diejenigen, die insbesondere die für die Entwicklung des OERR-Systems benutzte DHE-Umgebung berücksichtigen. Die Validierung der gesamten Merkmale der Konstruktionsphase befindet sich im Kapitel E3.3.

Funktionsabdeckung

Ein erstes grundsätzliches Qualitätsmerkmal in der Konstruktionsphase ist die *Funktionale Vollständigkeit*. Dieses wurde teilweise durch die Fragen F_1 bis F_7 auch in der Analysephase validiert. Deshalb wird die Aufzählung der Fragen nicht mit Eins beginnen, sondern mit der aus der Analysephase darauffolgenden Nummer. In der Konstruktionsphase werden für die Validierung dieses Merkmals folgende Prüffragen beantwortet:

F_8: Stimmt der Software-Entwurf mit den funktionalen Anforderungen aus der Spezifikation überein?

F_9: Wurden fehlende Funktionen des Produktes dokumentiert und begründet?

Die Frage F_8 bezieht sich auf das Entwurfsmodell, während F_9 auf das Implementierungsmodell angewendet werden kann. Durch die Auswertung dieser beiden Fragen soll festgestellt werden, ob die in der Analysephase beschriebenen funktionalen Anforderungen des OERR-Systems mit den von DHE zur Verfügung gestellten Diensten erfüllt werden können. Um dies validieren zu können, wird noch einmal auf die charakteristische Eigenschaft der OOSE-Methode, die Nachvollziehbarkeit zwischen den Modellen, verwiesen. So wird von OOSE gefordert, eine Übersetzung der Analyseobjekte in Entwurfsobjekte durchzuführen. Diese Übersetzung zeigte im Kapitel 6.3.1, daß nicht alle Analyseobjekte in Entwurfsobjekten eine Entsprechung finden (vgl. Tabelle 1). Dies ist die Folge der im Entwurfsmodell betrachteten DHE-Umgebung, die eine Vereinfachung in der weiteren Entwicklung des OERR-Systems mit sich bringt: DHE übernimmt einen großen Teil der Datenverwaltung für OERR, so daß viele der Entitätsobjekte aus dem Analysemodell im Entwurfsmodell des OERR-Systems nicht mehr notwendig sind, ohne jedoch die Nachvollziehbarkeit der beiden Modelle zu beeinträchtigen. Es müssen somit keine weiteren Überlegungen angestellt werden, wie die Daten des OERR-Systems gespeichert und konsistent gehalten werden können. Es kann festgestellt werden, daß alle im Analysemodell beschriebenen funktionalen Anforderungen im Entwurfsmodell umgesetzt werden können. DHE stellt somit für eine komplette Implementierung der betrachteten Anwendungsfälle des OERR-Systems alle notwendigen Dienste zur Verfügung. Inwieweit eine korrekte Funktionalität dieser Dienste von DHE gewährleistet wird, wird in der Testphase validiert. Für die im Kapitel 4.4.2 identifizierten charakteristischen OERR-Vorgänge konnten alle Funktionen mit Hilfe der DHE-Dienste umgesetzt werden. Beide Fragen (F_8 und F_9) können mit einer 1 (Zutreffend) beantwortet werden.

Korrektheit

Ein weiteres Merkmal, welches in der Konstruktionsphase zu validieren ist, ist die *Modulbezogene Konsistenz*. Dies ist ein Merkmal der `Korrektheit`. Die `Korrektheit` wird als jene Eigenschaft eines Produktes definiert, mit der dieses seiner Anforderungsspezifikation genügt. Der Begriff „modulbezogen“ bezieht sich im VB-Kontext sowohl auf das Konzept „Formular“ als auch auf die VB-BASIC-Module. Für die Auswertung dieses Merkmals wird die entsprechende Prüfliste aus Kapitel E1.1.3 benutzt. Um jedoch die oben beschriebene Begriffsspezialisierung (Modul = Formular + BASIC-Modul) zu verdeutlichen, werden die entsprechenden Fragen aus der Prüfliste wie folgt an die nun bekannte Entwicklunsgumgebung (VB) angepaßt und beantwortet:

F_1: Entsprechen alle implementierten Formulare- und BASIC-Module den Programmierrichtlinien?

Bei der Entwicklung des OERR-Systems wurden die von der ROKD GmbH festgelegten VB-Programmierrichtlinien für den Entwurf der Bildschirmmasken verwendet. So beginnen beispielsweise alle implementierten Formulare mit dem Schlüsselwort `frm`. Textfelder haben das Schlüsselwort `txt`. Weitere VB-Programmiervereinbarungen sind in [ROKD97] zu finden.

Auch wenn die DHE-BASIC-Module mit der DHE-Software gemeinsam geliefert worden sind, konnte jedoch festgestellt werden, daß diese auch nach bestimmten Programmierrichtlinien implementiert worden sind (vermutlich die Programmierrichtlinien der Firma GESI srl). So beginnen die Deklarationen der Dienste, die zu einem DHE-Manager gehören, mit der Bezeichnung `<manager>`, wobei `<manager>` die Werte `pm`, `am`, `um`, `hm`,`cm` oder `rm` annehmen kann. Frage F_1 kann somit aus beiden, am Anfang des Kapitels erwähnten Sichten (OERR und DHE) mit „Zutreffend“ beantwortet werden.

F_2: Werden alle benutzten Variablen initialisiert oder im Programm berechnet oder werden sie von einer externen Schnittstelle gelesen?

Die in dieser Arbeit betrachtete DHE Version 1.00a verlangt auf der Client-Seite eine Initialisierung derjenigen benutzten Input-Variablen, die als Parameter für die API-Aufrufe übergeben werden. Diese unbedingt notwendige Initialisierung erschien sehr lästig, so daß in der DHE-Version 1.00f diese Initialisierung auf der Client-Seite nicht mehr notwendig sein soll. Sie soll von dem Server, beim Aufruf des entsprechenden Dienstes, automatisch durchgeführt werden.

VB3 gilt als ungeeignet bezüglich der Variableninitialisierung: eine deklarierte Variable, dem noch kein Wert zugewiesen wurde, wird automatisch von VB3 mit Null initialisiert.

Frage F_2 wird bezüglich der DHE-Version 1.00a positiv („Zutreffend“) beantwortet.

F_3: Gibt es fehlende oder nicht benutzte Variablen in den BASIC-Modulen?

Die von DHE gelieferten BASIC-Module enthalten für einen deklarierten Manager (z.B. den Patient Manager) die Deklarationen aller Dienste, die dieser Manager zur Verfügung stellt. Diese Module wurden im `OERR.MAK`-Projekt (vgl. Kapitel 6.3.2) eingebunden, auch wenn nicht alle Dienste benutzt wurden. Dies führte dazu, daß eine Ausführung des Projektes wegen der geringen Arbeitsspeicherkapazität des Rechners, nicht mehr möglich war. Dieses Aspekt wird im Rahmen der Validierung der Eigenschaft `Effizienz` näher beschrieben. Eine Lösung, die von der Firma GESI srl vorgeschlagen wurde, ist, in diesen DHE-BASIC-Module diejenigen Funktions- und Datenstrukturdeklarationen auszukommentieren, die im `OERR.MAK`-Projekt nicht benutzt werden. Dieser Vorschlag erschien jedoch wenig akzeptabel: zum einen würde dem Entwickler einer auf DHE basierten Anwendung die Möglichkeit gegeben, den DHE-Quellcode zu ändern. Zum anderen ist es sehr aufwendig, die nicht benutzten Deklarationen von Funktionen (DHE-Dienste) und Datenstrukturen manuell in allen DHE-BASIC-Modulen zu suchen und auszukommentieren. Nach diesen Überlegungen wurden die DHE-BASIC-Module nicht geändert, so daß das implementierte OERR-System auch Module mit nicht benutzten Variablen enthält. Die Frage F_3 wird mit einer 1 (Zutreffend) bewertet.

F_4: Gibt es falsche oder fehlende Deklarationen in den BASIC-Modulen?

Die Frage F_4 wird mit „Unzutreffend“ beantwortet: falsche oder fehlende Datentypdeklarationen werden von VB direkt nach der Eingabe einer Quellcodezeile, in welcher die entsprechende fehlerhafte Datentypdeklaration vorhanden ist, entdeckt, so daß eine weitere Implementierung mit einem falschen oder fehlendem Datentyp nicht möglich wäre.

F_5: Sind die implementierten Formulare mit den entsprechenden Entwurfsobjekte aus dem Entwurf konsistent?

Die positive Antwort (1) auf die Frage F_5 ist durch den Einsatz der OOSE-Methode mit dessen Haupteigenschaft Nachvollziehbarkeit zwischen dem Entwurfs- und Implementierungsmodell begründet. So konnte fast jedes Entwurfsobjekt in ein VB-Formular übersetzt werden (vgl. Tabelle 4). Ausnahmen waren die Objekte *Sitzungsmanager* und *Patientenauswahl*, die jedoch konsistent im OERR-System mittels Variablendeklarationen umgesetzt wurden.

Sicherheit

Einer der wichtigsten Eigenschaften eines jeden Software-Systems ist die `Sicherheit`. Die `Sicherheit` ist die Eigenschaft eines Systems, innerhalb vorgegebener Grenzen für eine vorgegebene Zeitdauer einer unbefugten Nutzung/Beeinträchtigung von außen zu widerstehen (vgl. auch Kapitel E1.1.5). Die Eigenschaft `Sicherheit` kann durch folgende Merkmale validiert werden:

- *Sicherheitskontrollen / Zugriffsschutz*;
- *Besondere Sicherheitsvorkehrungen durch das System*;
- *Sicherungsmaßnahmen*;
- *Datenschutzmaßnahmen*.

Als sehr wichtig erscheint für das implementierte OERR-System die Validierung der Merkmale *Sicherheitskontrollen / Zugriffsschutz* und *Besondere Sicherheitsvorkehrungen durch das System*, auf die im folgenden näher eingegangen wird. Die Validierung aller Merkmale der Eigenschaft `Sicherheit` kann dem Kapitel E3.3 entnommen werden.

Das Merkmal *Sicherheitskontrollen / Zugriffsschutz* wird mit Hilfe der folgenden Prüfliste bewertet:

F_1: Gibt es Kontrollmechanismen im System?

Die DHE-Software stellt für die Implementierung eines Kontrollmechanismus im OERR-System einen Manager zur Verfügung (*User & Authorisation Manager*), mit Hilfe dessen Authorisierungprofile für verschiedene Benutzer und Benutzergruppen definiert werden können. So gehören beispielsweise Ärzte der Benutzergruppe „Medizinisches Personal“ an, während Schwestern der Benutzergruppe „Pflegepersonal“ angehören. Für Benutzer können auch Rechte für das Einloggen in das OERR-System auf einer oder mehreren Stationen oder Funktionsbereichen angegeben werden. Sie können dann nur auf die Daten der Stationen zugreifen, für die sie Rechte besitzen.
Der Kontrollmechanismus kann im implementierten OERR-System in den Anwendungsfällen `Allge-`

meine Sitzung, Anforderung verifizieren und Ergebnisse bestätigen wiedergefunden werden. So wird in Allgemeine Sitzung unter anderem das Login-Verfahren beschrieben. Ein Benutzer, der sich gegenüber dem OERR-System identifizieren möchte, muß seinen Namen, sein Paßwort sowie die Station, auf der er arbeiten möchte, eingeben. Das OERR-System prüft durch verschiedene DHE-Aufrufe, die als Input-Paramter unter anderem die vom Benutzer eingegebenen Daten haben, ob der Benutzer dem System bekannt ist und ob er über Rechte für die ausgewählte Station verfügt. Erst wenn eine Übereinstimmung mit den in der Datenbank gespeicherten Daten stattfindet, kann der Benutzer mit dem OERR-System arbeiten.

Bei dem Anwendungsfall Anforderung verifizieren und Ergebnisse bestätigen wird zusätzlich die Zugehörigkeit des Benutzers zu der Benutzergruppe „Medizinisches Personal“ kontrolliert. Nur das „Medizinische Personal“ verfügt über Verifikations- und Bestätigungsrechte von Anforderungen bzw. Ergebnisse. Außerdem müssen die Ärzte, die eine Anforderung eines Patienten oder Ergebnisse bestätigen wollen, Rechte in Bezug auf Daten eines Patienten besitzen, der sich auf eine Station aufhält.

Frage F_1 wird, nach den obigen Betrachtungen, positiv bewertet.

F_2: Gibt es Schutz- / Zugriffssysteme (ID, Paßwort) für Daten und Programmfunktionen?

DHE unterstützt nicht nur das Einloggen in das System durch ein Kontrollmechanismus, sondern ermöglicht auch Schutz-/Zugriffsmechanismen für Programmfunktionen und Daten im OERR-System. Ein weiteres Beispiel, welches diesen Aspekt vertieft, ist der Zugriff auf die Arzt-, Schwestern- und Ergebnisablagen (vgl. Formular frmAnforderndeKE in der Schnittstellenbeschreibung im Kapitel E2.3). Während auf die Daten, die sich in der Arztablage befinden, nur Ärzte Zugriff haben und auf Daten aus der Schwesternablage nur Schwestern, können auf Daten in der Ergebnisablage beide Benutzergruppen zugreifen, jedoch das Pflegepersonal nur mit Leserechten. Schreibrechte auf die Ergebnisablage bedeutet das Bestätigen der Ergebnisse durch einen authorisierten Arzt und das Löschen des Ergebnisses aus der Ablage.

Frage F_2 wird positiv bewertet.

F_3: Gibt es einen Mechanismus für die Verschlüsselung der Daten?

DHE stellt kein Verschlüsselungsmechanismus für Daten zur Verfügung und auch im OERR-System wird keines implementiert, so daß die Antwort auf Frage F_3 'Unzutreffend' lautet.

Zusammenfassend kann trotzdem die Schlußfolgerung gezogen werden, daß DHE einen umfangreichen Sicherheitskontroll / Zugriffsschutzmechanismus zur Verfügung stellt, der in DHE-basierten Anwendungen ohne viele Erweiterungen zu benutzen ist. Eine mögliche Erweiterung der von DHE zur Verfügung gestellten Sicherheitsmechanismen wäre die Implementierung eines Datenverschlüsselungskonzeptes.

Das Merkmal *Besondere Sicherheitsvorkehrungen durch das System* wird anhand der folgenden Fragen validiert (vgl. auch Kapitel E1.1.5):

F_1: Gibt es Funktionen, die das Systemverhalten überwachen?

Im OERR ist es nicht notwendig, Funktionen zu implementieren, die das Systemverhalten überwachen. Diesen Aspekt übernimmt DHE. So gibt es beispielsweise auf dem DHE-Server eine Datei dhe-prod.dmp, die alle Informationen (Dienst-Aufrufe, Datum- und Uhrzeit des Aufrufes, Input- und Output-Parameter des Aufrufes) registriert, die innerhalb einer DHE-Sitzung zwischen dem Server und dem Client fließen. Frage F_1 kann positiv ('Zutreffend') ausgewertet werden.

F_2: Gibt es Funktionen, die inkorrekte Funktionen / Störungen erkennen und gegebenfalls Gegenmaßnahmen ergreifen (z.B. Aufruf eines Fehlerbehandlungsmoduls)?

Das OERR-System wurde so implementiert, daß es zwischen Fehlermeldungen, die von DHE stammen und solchen die OERR-basiert sind, unterscheidet. Wenn eine inkorrekte Funktion, bzw. eine Störung beim Aufruf eines DHE-Dienstes entdeckt wird, so wird im OERR-System beispielsweise für den Aufruf pmTT__L folgende Fehlermeldung ausgegeben:

```
MsgBox „DHE-Dienst pmTT_L Fehler-Nr.“ + Str(ret)
```

D.h., daß außer der Meldung, daß der Aufruf des Dienstes pmTT__L fehlerhaft ist, wird auch die Fehler-Nummer ausgegeben (Str(ret)), wobei ret der Dienst-Aufruf ist. Anhand dieser Nummer kann anschließend in der Fehlerbehandlungsdokumentation von DHE gesucht werden (vgl. [Fer96a]), um die Fehlerursache zu finden. Beispiel eines Fehler wäre:

`Fehler-Nr: 10 bedeutet: „Server not active", d.h. daß der DHE-Server nicht hochgefahren wurde, bevor die Client-Anwendung gestartet wurde.`

Die Frage F_2 wird mit 'Zutreffend' bewertet.

F_3: Werden kritische Daten mehrfach gespeichert (Generationen, Auslagerungen nach bestimmten Zeitintervallen)?

Die Daten werden, so wie im Entwurfsmodell beschrieben wurde, nicht von OERR gespeichert, sondern mittels der DHE-API-Aufrufe von DHE in der Sybase-Datenbank. Eine mehrfache Speicherung von kritischen Daten findet sowohl seitens des OERR-Systems als auch seitens der DHE-Software nicht statt. Sybase stellt jedoch einen Backup-Mechanismus in Form eines Backup-Servers zur Verfügung, der parallel zu dem eigentlichen Sybase-Server läuft. So wird automatisch beim Einfügen, Ändern oder Löschen einer Instanz in der Sybase-Datenbank auch die entsprechende Spiegelung durchgeführt.

Die Frage F_3 wird mit 'Zutreffend' bewertet.

F_4: Werden beim Aufruf einer Transaktion Prüfungen auf Zugriffsberechtigung durchgeführt?

Prüfungen auf Zugriffsberechtigung beim Aufruf von Transaktionen werden von DHE ebenfalls mit Hilfe des *User and Authorisation Manager* unterstützt. Ein Beispiel sind die Anwendungsfälle `Anforderung generieren` oder `Ergebnis bestätigen`.

Die Antwort auf die Frage F_4 ist 'Zutreffend'.

F_5: Gibt es eine Datenbankzugriffskontrolle (z.B. Authorisierungstabelle)?

Ebenfalls in dem *User and Authorisation Manager* können Datenbankzugriffe für verschiedene Benutzer oder Programmteile festgelegt werden. Frage F_5 wird positiv bewertet.

F_6: Gibt es Systemfunktionen, die die Zugriffe aufzeichen?

Da DHE einen sehr stabilen und sicheren Zugriffskontrollmechanismus zur Verfügung stellt, werden keine weiteren Systemfunktionen, die den Zugriff auf einzelne Daten oder Programmfunktionen protokollieren, implementiert. Es wird seitens der DHE-Software die Sicherheit gewährleistet, daß jeder unbefugte Zugriff abgebrochen wird. Diese Frage wird mit 'Zutreffend' bewertet.

F_7: Gibt es Funktionen, die unmittelbar die Verletzung der Zugriffsrechte anzeigen?

Im OERR-System wurden Fehlermeldungen implementiert, die die Verletzung der Zugriffsrechte anzeigen (vgl. Frage F_7). So wird beispielsweise einer Stationsschwester, die versucht, eine Anforderung zu verifizieren, folgende Fehlermeldung angezeigt: „`Sie haben keine Rechte, die Anforderung zu verifizieren`". Frage F_7 wird mit 'Zutreffend' bewertet.

F_8: Wird der Benutzer durch das System angehalten, in periodischen Abständen seinen Berechtigungsnachweis zu ändern?

DHE erlaubt das Ändern des Paßwortes eines Benutzers, jedoch besitzt die entsprechende Entität (SP im *Act Manager* - vgl. Abbildung 27) keine Attribute, wo eine Zeitbegrenzung der Benutzung des Paßwortes gespeichert werden könnte. Die Zeitbegrenzung, bzw. die Aufforderung an den Benutzer, seinen Berechtigungsnachweis zu ändern, könnte in das OERR-System implementiert werden. So könnte beispielsweise alle zwei Wochen eine Aufforderung zum Paßwortwechsel erfolgen. Eine solche Funktion wird jedoch nicht im implementierten OERR-System berücksichtigt.

Frage F_8 wird mit 'Unzutreffend' bewertet.

Zusammenfassend lassen sich die Prüffragen des Merkmals *Besondere Sicherheitsvorkehrungen durch das System* folgendermaßen bewerten:

F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8
1	1	1	1	1	1	1	0

Es wird davon ausgegangen, daß sowohl durch das OERR-System und insbesondere durch DHE, besonders umfangreiche und stabile Sicherheitsvorkehrungen getroffen wurden.

Handhabbarkeit

Die `Handhabbarkeit` ist jene Eigenschaft eines Produktes, die die Einfachheit und Angemessenheit der Bedienerführung, des Produktverhaltens in Fehler- und Störfällen, der Lernunterstützung (Hilfe-Funktionen) und der Realisierung der Mensch-Maschine-Schnittstelle bestimmt. Im folgenden wird das Merkmal *Rationalität* näher betrachtet. Die Fragen aus der Prüfliste dieses Merkmals (vgl. Kapitel E1.2.2) betreffen die Benutzungsfreundlichkeit sowohl bezüglich der DHE-Software als auch des OERR-Systems. Der Benutzer ist aus DHE-Sicht der Applikations-Entwickler und aus der Sicht des OERR-Systems das Krankenhauspersonal.

Die Überprüfung der *Rationalität* des OERR-Systems betrifft die Validierung der Schnittstellenbeschreibung seitens des Krankenhauspersonals. Die Schnittstellenbeschreibung aus der Analysephase (vgl. Kapitel 5) wurde mit abstrakten Elementen bzw. Interaktionsformen dargestellt. Der Übergang vom Abstrakten zum Konkreten erfolgt im Entwurfsmodell. Hier werden sowohl Überlegungen in Richtung einer tatsächlichen Implementierung der im Analysemodell benutzten Elemente festgeschrieben, als auch die Verbesserungsvorschläge des Krankenhauspersonals aus dem Mathias-Spital berücksichtigt. Diese Verbesserungsvorschläge entstanden in der Analysephase während der Validierung der Benutzungsschnittstelle im Rahmen des 4. German HANSA-Meetings (vgl. Kapitel 5 - Die Schnittstellenbeschreibung). So führten diese Verbesserungsvorschläge, sowie die Identifizierung der Systemumgebung und somit auch der eingesetzten Programmiersprache VB3 zu Änderungen in den Bildschirmmasken der Schnittstellenbeschreibung. So konnte der *Belegungsplan* aus Abbildung 19 übersichtlicher in Form einer Liste dargestellt werden (vgl. Abbildung 33). Jeder Eintrag in der Liste enthält die *Aufenthalts-ID*, den *Name* und den *Vorname*, sowie die *ID* des Patienten. Weiterhin wurden die Ablage-Symbole geändert.

Eine weitere Änderung der Schnittstellenbeschreibung bezieht sich auf die Bildschirmmaske *Formularmaske* (vgl. Abbildung 20). Diese wurde, wie in Abbildung 34 dargestellt, geändert. Die Textfelder *Blutdruck* und *Vorherige Aufenthalte* wurden weggelassen, da sie für eine Leistungsanforderung zum Zeitpunkt der Anforderunsgenerierung, laut Mathias-Spital-Mitarbeiter, nicht relevant sind. Weiterhin wurde das Feld *Priorität* aus der *Maske_Details* (vgl. Abbildung 21) in die *Formularmaske* eingefügt: es wurde anhand der DHE-Dokumentation festgestellt, daß eine Speicherung der Priorität für jede Einzelleistung in DHE nicht möglich ist, sondern nur für eine gesamte Anforderung die auf einmal verarbeitet wird. Der Hintergrund ist, daß eine Anforderung als eine Aktivität von DHE betrachtet wird, die nur eine Priorität zu einem Zeitpunkt haben kann. Die Suche nach anderen Einzelleistungen im Anforderungskatalog eines bestimmten Anforderungstyps (Röntgen, EKG etc.) wurde ebenfalls vereinfacht: der Anforderungstyp (in Abbildung 34 - EKG) wird schon vor dem Laden der *Formularmaske* festgelegt (beim Aufruf des Menüpunktes `Anforderung->Anforderung generieren->EKG` aus der Funktionsleiste des Belegungsplans). Somit gibt es in der *Formularmaske* nur die Möglichkeit, Einzelleistungen zu dem schon festgelegten Anforderungstyp auszuwählen. Eine Suche einer Einzelleistung ist dann auch nur im betreffenden Anforderungskatalog möglich. Es macht also kein Sinn, dem Benutzer die Möglichkeit zu geben, selbst nach einem Anforderungskatalog zu suchen oder sogar eine beliebige Zeichenfolge im betreffenden Textfeld einzugeben (vgl. Abbildung 21). Durch das Betätigen des Buttons `SUCHE STARTEN` wird die Suche nur im Anforderungskatalog gestartet, der zu dem vorher festgelegten Anforderungstyp gehört. Dadurch wird eine fehlerhafte Eingabe des Benutzers verhindert.

Um die Arbeit des Benutzers zu erleichtern und ihm längere Wege durch Dialogboxen zu ersparen, wurde auf die Bildschirmmaske *Details der selektierten Einzelleistung* aus Abbildung 23 verzichtet. Die ausgewählten und bearbeiteten Einzelleistungen innerhalb einer Anforderung können auch in der *Formularmaske* als Anforderungsprofil angesehen werden.

Letztendlich wurde der Button `SENDE` mit dem Button `ARZTABLAGE` ausgetauscht, damit eine mögliche Verwirrung des Benutzers vermieden wird: die Button-Beschriftung `SENDE` könnte bedeuten, daß das in der *Formularmaske* bearbeitete Anforderungsprofil direkt zu der leistenden KE gesendet wird. Die Beschriftung `ARZTABLAGE` soll daraufhindeuten, daß das Anforderungsprofil in der Arztablage zwecks Verifizierung abgelegt wird.

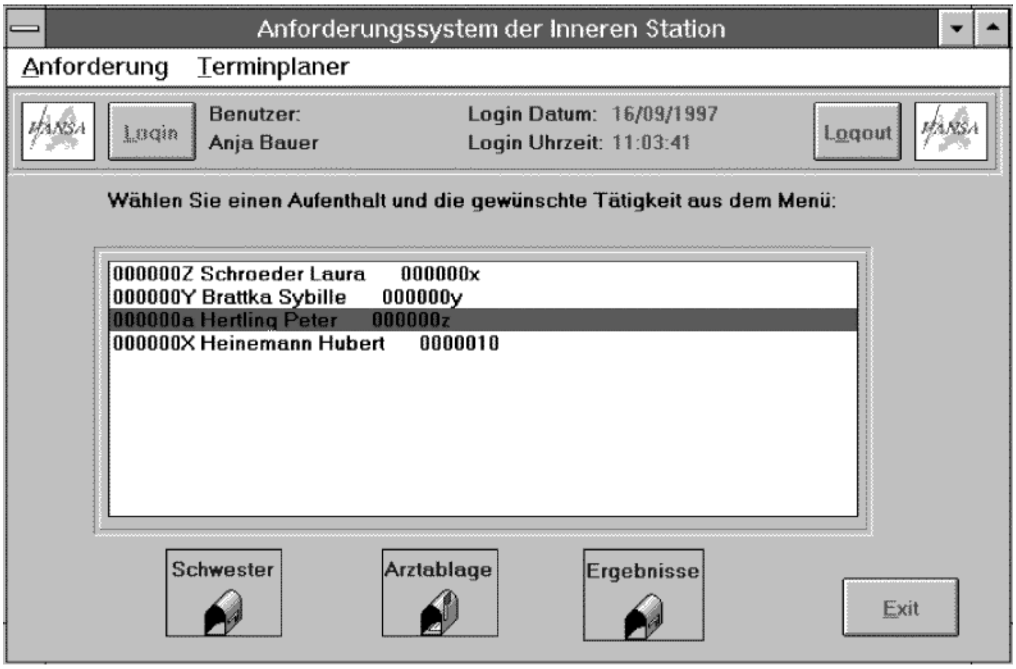

Abbildung 33: Der *Belegungsplan* nach der Validierung durch das Krankenhauspersonal

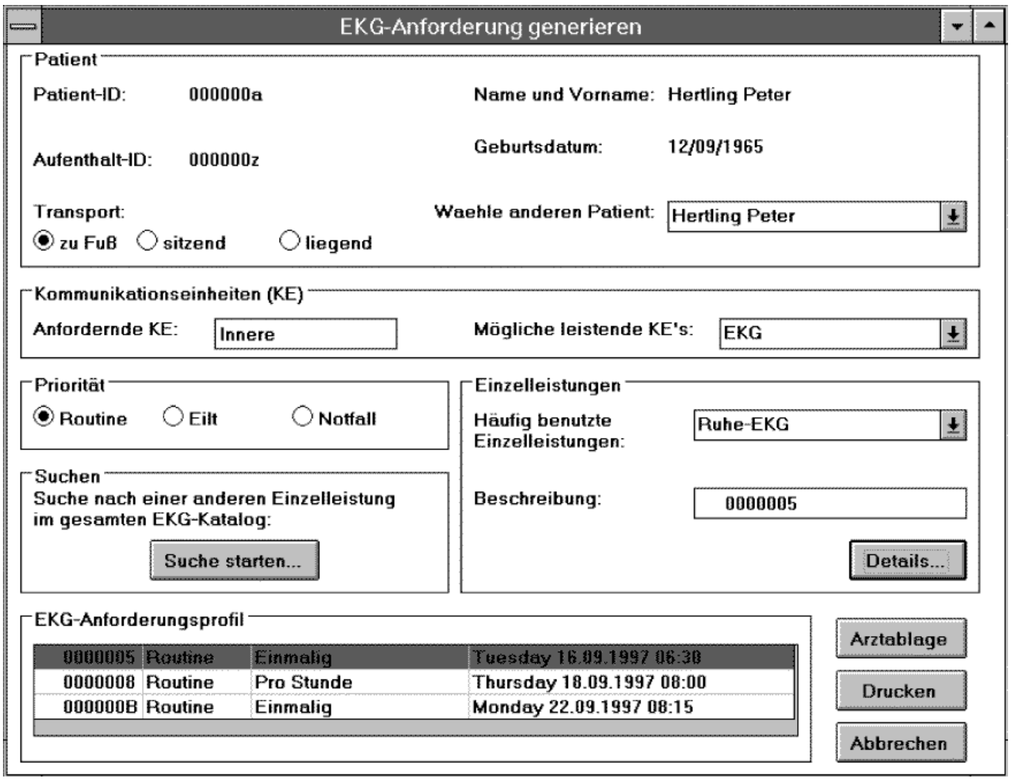

Abbildung 34: Die *Formularmaske* nach der Validierung durch das Krankenhauspersonal

Die letzte Bildschirmmaske, die geändert wurde, ist die *Maske_Details* (vgl. Abbildung 35).

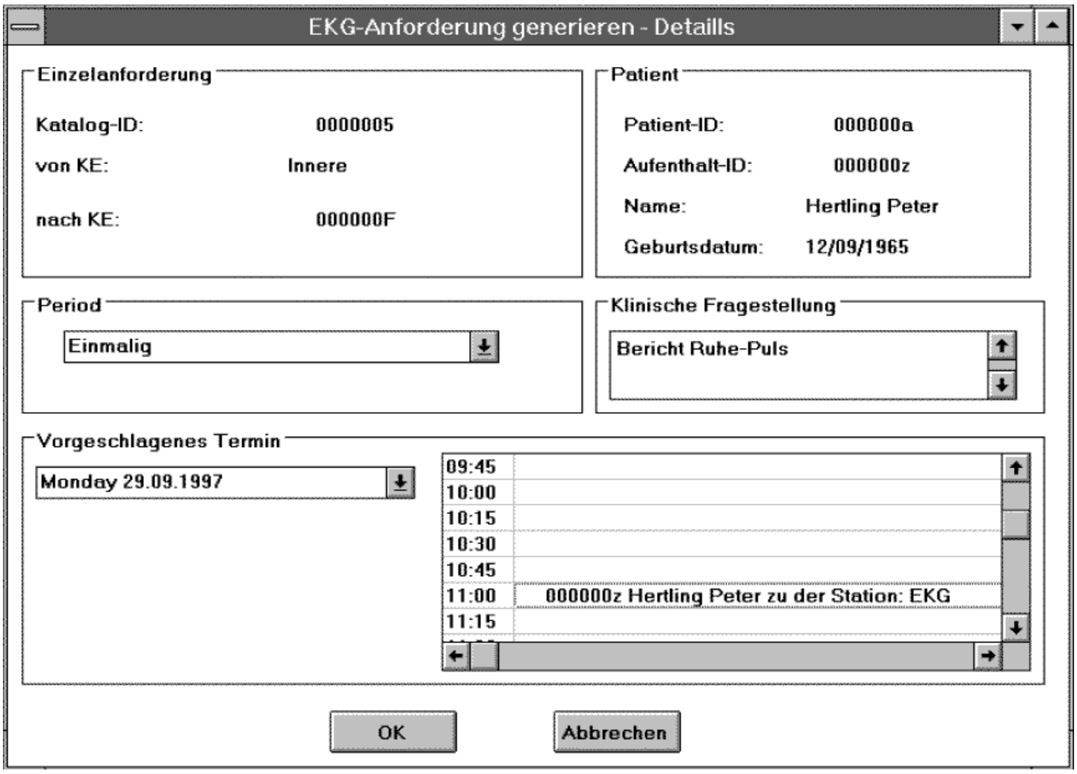

Abbildung 35: Die *Maske_Details* nach der Validierung durch das Krankenhauspersonal

Gegenüber der Maske aus Abbildung 21 wurde das Feld *Priorität* herausgenommen. Nach den Vorschlägen der Mathias-Spital-Mitarbeiter (vgl. Kapitel 5 - Schnittstellenbeschreibung) wurde auch der Terminplaner der leistenden KE herausgenommen. Stattdessen wurde ein konventioneller Kalender eingefügt, mit dem Datum und Uhrzeit der Durchführung der Einzelleistung ausgewählt werden können.

Die gesamten Bildschirmmasken, sowohl des OE- als auch des RR-Teils des OERR-Systems, so wie sie nach der Berücksichtigung der Verbesserungsvorschläge durch das Krankenhauspersonal geändert wurden, können dem Kapitel E2.3.2 entnommen werden.

Wie im Kapitel 5.3.3 erwähnt wurde, bringt eine Änderung der Schnittstellenbeschreibung manchmal auch Änderungen in dem Ablauf der Bildschirmmasken mit sich und somit des in Abbildung 18 dargestellten Dialognetzes. Die neue Version des Dialognetzes des gesamten OERR-Systems, mit der Berücksichtigung dieser Änderungen, wird im Kapitel E2.3.4 dargestellt.

Durch die oben aufgeführten Änderungen der Benutzungsschnittstelle können die Benutzereingaben durch aussagefähige Informationen rasch und zwecksmäßig unterstützt werden. Weiterhin wird durch diese Änderungen die Gliederung der auf dem Rechner abgebildeten Aufgaben für den Benutzer zwecksmäßig, klar und deutig aufgestellt. Somit kann die *Rationalität* des OERR-Systems, nach Berücksichtigung der obigen Änderungen, positiv bewertet werden.

Effektivität

Die `Effektivität` ist die Eigenschaft eines Produktes, in welchem Ausmaß der Benutzer bei seiner Aufgabenerfüllung durch das Produkt unterstützt wird. Die `Effektivität` hat als einziges Merkmal die *Betriebsorganisatorische Aufgabenerfüllung,* welches in der Konstruktionsphase aus zwei Sichten validiert wird: der DHE- und der OERR-Sicht. Aus der Sicht der DHE-Software wird das Merkmal von dem Applikations-Entwickler validiert. Aus der Sicht des OERR-Systems, wird die `Effektivität` durch die Mitarbeiter (Krankenhauspersonal) des Mathias-Spital validiert. Folgende Ergebnisse sind festzuhalten:

F_1: Ist die Schnelligkeit der Informationsbereitstellung angemessen?

Der Entwickler einer auf DHE-basierten Anwendung kann die Schnelligkeit der Informationsbereitstellung in der Datei `dheprod.log` nachlesen, die sich auf dem DHE-Server befindet. So wird beim Laden einer jeden Bildschirmmaske eine gewisse Anzahl an DHE-Dienste aufgerufen, deren Ausführungszeit in dieser Datei von DHE selbst registriert wird. Die Summe der Ausführungszeiten der für das Laden einer Bildschirmmaske aufgerufenen DHE-Dienste, gibt Aufschluß über die Schnelligkeit der Informationsbereitstellung. Folgende Dienst-Ausführungszeiten werden für das Laden der Bildschirmmaske *Formularmaske* (vgl. auch Abbildung 34) vom Belegungsplan in die Datei `dheprod.log` dargestellt:

```
amAS__L |sec      0.786
amFA__L |sec      0.546
pmPA__R |sec      0.314
pmTT__L |sec      0.601
```

Die Summe dieser Ausführungszeiten ergibt im Durchschnitt ca. 2,25 Sekunden für das Laden dieser Bildschirmmaske. Die hier dargestellte Sichtweise auf die Schnelligkeit der Informationsbereitstellung ist objektiv zu betrachten. Subjektiv erscheint die Dauer für die Informationsbereitstellung auch für die Mathias-Spital-Mitarbeiter beim Vorführen der Anwendung akzeptabel.

Frage F_1 kann aus objektiver und subjektiver Sicht positiv bewertet werden.

F_2: Ist die Aktualität der Informationen ausreichend?

Die Aktualität der Informationen wird aus zwei Sichten betrachtet, bezüglich der OERR-Benutzungsoberfläche und der DHE-Datenbank. Es wird analysiert, ob DHE aktiv Änderungen in der Datenbank erkennt und die Ausgaben auf der OERR-Benutzungsoberfläche entsprechend aktualisiert, oder ob dafür zusätzlich Quellcode auf der Client-Seite implementiert werden muß.

DHE ist eine passive Komponente. D.h., wenn neue Datensätze in der relationalen integrierten Sybase-Datenbank gespeichert werden, können diese erst nach dem entsprechenden DHE-Dienstaufruf sichtbar auf der Benutzungsoberfläche gemacht werden. Eine automatische Aktualisierung des Datenbestandes, welches auf der Benutzungsoberfläche der OERR-Anwendung sichtbar ist, findet nicht statt. Folgendes Beispiel soll das passive Verhalten der DHE-Software besser erläutern:

Es wird das Anwendungsfall `Anforderung versenden` (vgl. Kapitel E2.2) betrachtet. In diesem Anwendungsfall wird beschrieben, wie eine schon generierte Einzelleistung, nachdem sie verifiziert wurde, von einem ✍Stationsarzt an die leistende KE versendet wird. Eine Einzelleistung wird unter anderem mit Hilfe des DHE-Dienstes `amReqAct` versendet. Sie bekommt eine eindeutige Identifikationsnummer und den Status „angefordert“.

Der ResultReporting (RR)-Teil der OERR-Anwendung auf der leistenden KE stellt dem dortigen Benutzer nach dem Einloggen eine Liste zur Verfügung, in der alle angeforderten Einzelleistungen mit deren Stati enthalten sind. Sobald eine Einzelleistung von der anfordernden KE gesendet wurde, wird ein neuer Datensatz in der Datenbank angelegt. Dieser Datensatz muß in der obigen Liste als neuer Eintrag erscheinen. Die Aktualisierung dieser Liste mit den neu angeforderten Einzelleistungen erfolgt nicht automatisch, sobald der neue Datensatz in der Datenbank geschrieben wurde. Um regelmäßig diese Liste zu aktualisieren, wurde in dem `OERR.MAK`-Projekt das VB-Timer benutzt, der in regelmäßigen Zeitabständen (alle fünf Sekunden) den DHE-Dienst `amAA__L` mit den entsprechenden Parameter aufruft. Nach dem Aufruf wird die Liste um die neu eingetroffenen Einzelleistungen aktualisiert.

Auf der Windows NT Plattform wird es jedoch mit der DHE-Version 1.00f demnächst möglich sein, DHE auf eine anderen Art und Weise aktiv zu machen: durch den Einsatz von Triggern in der Datenbank, können diese an Anwendungen gesendet werden, wenn sich Daten geändert haben, die diese Anwendung benutzt (vgl. [Has97]). Am Beispiel der OERR-Anwendung könnte der Triggermechanismus folgendermaßen funktionieren: falls neu angeforderte Einzelleistungen durch die OE-Teilanwendung generiert wurden (d.h. es wurden Daten in die Datenbank eingefügt), so schickt diese Teilanwendung ein Trigger an die RR-Teilanwendung, der als Folge das Aufrufen des DHE-Dienstes `amAA__L` haben wird. So wird die Liste, die durch diesen Dienstaufruf erzeugt wird, immer die aktuelle Sicht auf die Datenbank haben. Somit wird ohne Verzögerung die komplette Liste, inklusiv der neu eingetroffenen Einzelleistungen, angezeigt.

Der zweite Aspekt der Aktualität der Informationen aus DHE-Sicht ist die Änderung (Aktualisierung) von schon vorhandenen Datensätze. Beispielsweise wird ein Patient aus einem Zimmer auf ein anderes Zimmer auf der Station verlegt. Dafür wird nicht ein neuer Datensatz erzeugt, sondern der schon vorhandene aktualisiert. DHE stellt für das Updaten von Daten spezielle Dienste vom Typ __M zur Verfügung, die als i_action - Parameter das „M“ (Modified) übergeben bekommen. Pflichtfelder für diese Dienste sind die Angaben der eindeutigen Identifikationsnummer des entsprechenden Datensatzes sowie der sogenannte „Timestamp“ des Datensatzes. DHE stellt also Rückgabetypen für Dienste zur Verfügung (__M), die Aussagen über die Aktualisierung von Daten in der Datenbank machen.

Frage F_2 wird mit 'Zutreffend' bewertet.

F_3: Ist die Genauigkeit der Informationen ausreichend?

Die Genauigkeit der Informationen bezieht sich im Kontext dieser Arbeit auf die Genauigkeit von DHE mit der es erlaubt, Informationen zu speichern. Ein Beispiel, welches dieses Aspekt verdeutlicht, wird im folgenden beschrieben:

Auf der leistenden KE gibt es beispielsweise eine Regelung, daß die Durchführung der Einzelleistungen in der Reihenfolge des Ankommens gewährleistet wird. In DHE muß es also möglich sein, diese Ankommenszeit einer Einzelleistung speichern zu können. Andererseits muß die RR-Anwendung auf der leistenden KE eine entsprechende Benutzungsoberfläche dem Benutzer zur Verfügung stellen, damit dieser feststellen kann, welche Einzelleistung als nächste durchgeführt werden muß (d.h. daß die Benutzungsoberfläche ein Feld für die Ankommenszeit enthalten muß).

Für die ausgewählten charakteristischen OERR-Vorgänge konnte eine ausreichende Genauigkeit der Informationen, die DHE zur Verfügung stellt, festgestellt werden. Frage F_3 wird deshalb positiv (1) ausgewertet.

F_4: Ist die Anpassungsfähigkeit des Produktes gegenüber Veränderungen in der Organisation, im Datenvolumen und bei Sonderfällen gegeben?

In der Konstruktionsphase wurde festgelegt, daß das entwickelte OERR-System nicht die Daten verwalten muß, da dieser Aspekt von DHE übernommen wird. Eine Veränderung in der Organisation des Krankenhauses oder im Datenvolumen bringt keine Änderungen der OERR-Anwendung mit sich. Die Veränderungen werden nur innerhalb von DHE vorgenommen.

Wenn eine leistende KE beispielsweise die Durchführung einer neuen Einzelleistung anbietet, so muß diese Einzelleistung in der *Formularmaske* auf jeder anfordernden KE erscheinen. Im OERR-System ist dafür der Aufruf des Dienstes amAS__L vorgesehen, der eine Liste aller Einzelleistungen, die eine ausgewählte leistende KE durchführt, ausgibt. Die neu angebotene Einzelleistung muß somit nur als ein neuer Datensatz in DHE eingefügt werden. So wird die Ausgabe des Dienstes amAS__L aus OERR auch diesen Datensatz enthalten. Wie neue Datensätze in DHE eingefügt werden, wird in der Testphase beschrieben.

Änderungen im Datenvolumen könnten auftreten, wenn beispielsweise durch das GSG verlangt wird, neue Daten zu speichern und zu verwalten, die bislang nicht berücksichtigt worden sind. Falls die Speicherung bzw. Verwaltung dieser Daten von DHE unterstützt wird, so müssen diese durch die OERR-Benutzungsoberfläche und die entsprechenden DHE-Dienste dem Benutzer sichtbar gemacht werden. Falls jedoch DHE die entsprechenden Daten nicht unterstützt, so muß eine Erweiterung von DHE durchgeführt werden. DHE stellt dafür sogenannte *Extended Services* zur Verfügung, die an spezifischen Notwendigkeiten der einzelnen Länder neu definiert und angepaßt werden können, ohne von der Firma GESI abhängig zu sein.

Sonderfälle können dann auftreten, wenn vorübergehend oder sehr selten bestimmte Veränderungen in der Organisation des Krankenhauses durchzuführen sind. In diesen Fällen kann ebenfalls eine Erweiterung des OERR-Systems oder von DHE (siehe oben) in Betracht gezogen werden.

Frage F_4 wird mit „Zutreffend“ beantwortet.

F_5: Sind für Fehler, die der Benutzer macht, Korrekturmöglichkeiten vorhanden und ist der Aufwand für die Durchführung dieser Korrekturen gering?

Es wurde bei der Validierung der Eigenschaft Handhabbarkeit eine Änderung der Schnittstellenbeschreibung vorgenommen. Diese Änderung hatte unter anderem auch der Zweck, die Eingaben, die der Benutzer vornimmt, so einfach wie möglich zu gestalten, um eventuelle Eingabefelder (Schreibfehler etc.) zu verhindern. Ein Beispiel war die Eingabe des Stichwortes für die Suche im Katalog der Einzelleistungen (vgl. Abbildung 20). Dort konnten leicht Eingabefehler seitens des Benutzers gemacht werden. Die Suche wurde erleichtert, in dem der Benutzer nichts mehr eingeben muß. Die Suche im

Anforderungskatalog wird vom System aus beschränkt: der Anforderungstyp wird aus dem Menü des Belegungsplans von der ✍Stationsschwester ausgewählt (vgl. Abbildung 34). Weiterhin wurden viele Benutzereingaben durch Selektieren aus *ListBoxen* und *ComboBoxen* geändert, indem *default*-Werte eingetragen wurden. Ein Beispiel für mögliche Benutzerfehler ist in der Bildschirmmaske mit dem Belegungsplan der anfordernden KE (vgl. Abbildung 33). Das Krankenhauspersonal versuchte des öfteren, den Menüpunkt `Anforderung->Anforderung generieren` auszuwählen, bevor es einen Patienten aus dem Belegungsplan selektiert hat. Es erschien auf dem Bildschirm für den Benutzer die Fehlermeldung „`Bitte wählen Sie einen Patienten aus!`“. Erst nachdem der Benutzer durch Klicken des `OK`-Buttons diesen Dialogbox beendet hat, kehrte er in dem Anfangszustand zurück und konnte einen Patienten auswählen. Um diese häufig aufgetretene Fehlermeldung zu vermeiden, um somit auch die Effektivität des Systems zu erhöhen, wird nun der obige Menüeintrag erst dann aktiviert, nachdem der Benutzer ein Patient aus dem Belegungsplan selektiert hat.

Letztendlich blieb eine minimale Freitext-Benutzereingabe im OERR-System vorhanden. Lediglich nur der Eingabefeld „Klinische Fragestellung“ aus Abbildung 35 verlangt, optional, die Eingabe eines Freitextes seitens des Benutzers.

So konnten Benutzereingabefehler und somit umfangreiche Beschreibungen für mögliche Korrekturen die der Benutzer selbst vornehmen könnte, drastisch reduziert werden. Dadurch können auch die Standard-Ergonomiekriterien eingehalten werden (vgl. [DIN88] und [ISO90]).

Weiterhin ist jede Dialogbox mit einem Button `ABBRECHEN` versehen, nach dessen Betätigung alle Eingaben in der Dialogbox storniert werden und der Systemzustand vor dem Öffnen des Dialogboxes wiederhergestellt wird (vgl. auch Kapitel 5 - Schnittstellenbeschreibung).

Frage F_5 wird positiv validiert.

Erweiterbarkeit

Im Rahmen der Qualitätseigenschaft `Erweiterbarkeit` werden auch in der Konstruktionsphase weitere Fragen aus der Prüfliste des Merkmals *Flexibilität der Struktur* beantwortet. Die Fragen F_1 und F_2 zu diesem Merkmal wurden in der Analysephase betrachtet (vgl. Kapitel 5.4). In der Konstruktionsphase werden für die Validierung des Merkmals *Flexibilität der Struktur* folgende Prüffragen beantwortet:

F_3: Wurden bereits beim Entwurf mögliche Erweiterungen vorgesehen?

Diese Frage bezieht sich einerseits auf Erweiterungen des OERR-Systems und andererseits auf Erweiterungen der DHE-Software. Da es sich in dieser Arbeit um die Konstruktion und Implementierung von charakteristischen OERR-Vorgänge handelt und nicht die Entwicklung eines gesamten OERR-Systems angestrebt wird, wird sowohl das Analyse- als auch das Entwurfs- und Implementierungsmodell auf diese Vorgänge beschränkt und keine Erweiterungen des OERR-Systems vorgesehen.

Wie im Rahmen der Eigenschaft `Effektivität` erwähnt wurde, erlaubt die DHE-Software auch Erweiterungen in Form von erweiterten Diensten (*Extended Services*). Diese Dienste können von dem Entwickler selbst definiert werden, falls die DHE-Standard-Dienste nicht die gesamte Funktionalität der geplanten Anwendung unterstützen. Erweiterte Dienste werden bei der Entwicklung der OERR-Anwendung nicht benutzt, da die Funktionalität der ausgewählten charakteristischen Vorgänge durch die vorhandenen DHE-Standard-Dienste unterstützt wird.

Frage F_3 kann aus beiden betrachteten Sichten mit „Unzutreffend“ bewertet werden.

F_4: Können Updates der DHE-Software durchgeführt werden, ohne daß die OERR-Anwendung in Mitleidenschaft gezogen wird?

Die wohl wichtigste Schnittstelle der entwickelten OERR-Anwendung ist die Schnittstelle zu DHE. DHE wurde zuerst als Version 1.00a installiert. Im Laufe der Durchführung der Konstruktionsphase, nachdem verschiedene Fehler in der DHE-Software der Firma GESI gemeldet wurden, haben die Mitarbeiter von GESI eine neue DHE-Version, 1.00f, den HANSA-Projektpartner zur Verfügung gestellt. Die im Laufe der OERR-Entwicklung aufgetretenen DHE-Fehler wurden in einer informellen Art und Weise dokumentiert und den Mitarbeitern der Firma GESI gesendet. Diese intensive Kommunikation erschien sinnvoll, um sicherzustellen, daß die aufgetretenen Fehler tatsächlich DHE-bedingt und nicht durch die Implementierung des OERR-Systems eingefügt wurden.

Die neue DHE-Version wurde versucht in ROKD zu installieren, dies erschien jedoch sehr zeitaufwendig: die von GESI gesendete komprimierte Datei erwies sich nach der Dekomprimierung als fehlerhaft, was zu einer Beschädigung der Datenbank führte. Nach einer Wiederherstellung der Datenbank wurde auf eine Aktualisierung der DHE-Version wegen Zeitproblemen verzichtet. Die Dokumentation der neuen DHE-Version verspricht jedoch, daß die gemeldeten Fehler nun behoben sind.

Die Frage F_4 kann demzufolge nicht beantwortet werden, da keine Schnittstellenänderungen durchgeführt wurden.

Portabilität

Unter `Portabilität` eines Programmsystems wird jene Eigenschaft eines Systems verstanden, auf unterschiedlichste Rechnersysteme übertragen werden zu können. Die Auswertung der `Portabilität` bezieht sich in dieser Arbeit auf die Portierbarkeit einerseits der DHE-Software auf verschiedene Maschinen und Betriebssysteme und andererseits des OERR-Systems in verschiedene Umgebungen.

Die DHE-Portabilität wurde in dieser Diplomarbeit nicht praktisch getestet, weil die Umgebung, in der die DHE-Software installiert worden ist (vgl. Kapitel 6.3.1), während der Validierung nicht geändert wurde. Es besteht jedoch die Möglichkeit den DHE-Server auch auf den Plattformen Hewlett Packard, IBM oder Tandem zu installieren.

Als die DLL für Windows 95 von der Firma GESI zur Verfügung gestellt wurde, war schon ein Teil der OERR-Anwendung unter Windows 3.1 in VB3 implementiert. Es wurde versucht, diesen Teil in die Windows 95-Umgebung zu portieren um weiter in VB4, nicht zuletzt wegen den besseren Performance (insbesondere der Speicherkapazität), zu arbeiten. Aus dieser Sicht wurde das Merkmal der `Portabilität`, die *Maschinen- und Betriebssytemunabhängigkeit*, in der Konstruktionsphase mittels folgender Prüfliste validiert:

F_1: Sind hardwareabhängige Details in eigenen Modulen gekapselt?

F_2: Wird im Code auf Dienstprogramme des Betriebssystems zurückgegriffen, bzw. werden Routinen aus der Systembibliothek verwendet?

Frage F_1 wird mit „Zutreffend“ (1), wärend Frage F_2 mit „Unzutreffend“ beantwortet werden kann. Im OERR-System wurden keine hardwareabhängige Details implementiert. Diese befinden sich in der Datei `services.gas`, die von DHE zur Verfügung gestellt wird (vgl. Kapitel 6.3.2.1). Der Quellcode von OERR greift auch nicht auf Dienstprogramme des Betriebssystems zurück. Es werden nur Routinen aus der Systembibliothek in OERR verwendet (*Beispiel: MsgBox, Custom Control Objekte etc.*). Eine Portierung auf Windows 95 ist nicht möglich. Dies liegt nicht an Windows 95 sondern an VB4. Die schon mit VB3 implementierten Bestandteile des OERR-Systems können nicht in VB4 übernommen werden um weiter mit VB4 zu implementieren. Einige der benutzen Objekte aus VB3 sind in VB4 nicht benutzbar. Ein Beispiel dafür ist das „Grid“- Custom Control Objekt für die Darstellung einer Tabelle aus der Bildschirmmaske *Formularmaske*. Folgende Fehlermeldungen erscheinen beim Laden des Projektes `OERR.MAK` in VB4 auf dem Bildschirm:

```
VB hat eine Instanz von „VBXGrid“ in einer binären Formdatei gefunden.
VBX-Dateien können in der 32-bit-Version nicht aus binären Formdateien
umgewandelt werden.
Fehler beim Laden von FRMANFGE.FRM. Ein Steuerelement konnte aufgrund
eines Ladefehlers nicht geladen werden.
```

Ähnliche Fehlermeldungen erscheinen auch für Instanzen von „VBXSSPanel“ und „VBXSSCommand“. Eine Anpassung dieser Objekte an die VB4-Konzepte wird wegen des zu großen Aufwands nicht durchgeführt. Auf eine Portierung der schon teilweise implementierten OERR-Anwendung von VB3 unter Windows 3.1 auf VB4 unter Windows 95 wird deshalb verzichtet. Es ist möglich, eine Anwendung mit DHE direkt in einer 32-bit Umgebung zu implementieren. Die dafür notwendige `dhe.dll` - Datei wurde jedoch zu spät ausgeliefert, so daß sie eine Portierung der OERR-Anwendung auf eine 32-bit Umgebung in dieser Arbeit nicht berücksichtigt wird. Damit ist die `Portabilität` der OERR-Anwendung nicht ein Problem von DHE sondern der verwendeten MS-Windows Version.

Wiederverwendbarkeit

Die `Wiederverwendbarkeit` wird in diesem Kapitel als letzte Eigenschaft validiert. Unter `Wiederverwendbarkeit` wird die Eignung des Produktes oder einzelner Teile zur Übertragung in ein anderes Anwendungsgebiet bei gleichbleibender Systemumgebung verstanden. In dieser Arbeit bezieht sich die Wiederverwendbarkeit auf die Übernahme verschiedener Teile der OERR-Anwendung in andere Anwendungen, die mit DHE entwickelt werden.

Es gibt zwei Arten von Auswertungsmethoden für die Bewertung der `Wiederverwendbarkeit`: ein Bewertungsmaß und eine Prüfliste. Das Bewertungsmaß ist die *Allgemeingültigkeitskenngröße* einer zu wiederverwendbaren Komponente und wird mittels folgender Formel berechnet:

$$AG = \frac{M_A}{M_G}$$

wobei gilt:

AG = die Kenngröße einer Komponente, von verschiedenen Applikationen unverändert benutzt werden zu können;
M_A = Anzahl der Module, die von anderen Applikationen unverändert verwendet werden können;
M_G = Gesamtanzahl der Module der Applikation.

Hier erscheint erneut der Begriff *Modul*, welches in VB ein Formular oder ein BASIC-Modul sein kann. Die BASIC-Module stammen von DHE und können ohne jegliche Änderungen auch in anderen VB-Anwendungen wiederverwendet werden.

Es konnte jedoch auch festgestellt werden, daß ebenfalls ein in der OE-Anwendung implementiertes Formular in der RR-Anwendung wiederverwendet werden kann. Es handelt sich um die Login-Maske `frmLogin.frm` der beiden Teilanwendungen OE und RR. Da beide dieser Teilanwendungen das Rollenmechanismus benötigen, konnte u.a. ein Login-Vorgang implementiert werden, der sowohl die Anmeldung des Krankenhauspersonals gegenüber dem OERR-System auf der anfordernden KE als auch auf der leistenden KE unterstützt. Dieser Vorgang, bzw. dieses Formular, könnte auch in einer anderen DHE-basierten Anwendung (beispielsweise in einem Pflegesystem) ohne jegliche Änderung wiederverwendet werden.

Die *Allgemeingültigkeitskenngröße* kann nun folgendermaßen berechnet werden:

M_G = 28 Module: dies entspricht der Gesamtanzahl der Module aus dem `OERR.MAK`-Projekt (Tabelle 3). Davon sind jedoch nur elf Formulare und ein OERR-BASIC-Modul für die OERR-Anwendung implementiert worde, wohingegen 16 BASIC-Module von DHE stammen. M_G erhält demnach den Wert 12.

M_A = 1 Modul: aus den 11 Formularen der OERR-Anwendung konnte eine einzige in einer anderen Teilanwendung wiederverwendet werden.

Daraus folgt:

$$AG = \frac{M_A}{M_G} = \frac{1}{12} = 0{,}08$$

Dieser niedrige Wert zeigt, daß OERR doch eine spezielle Anwendung ist, deren Teile nur sehr gering wiederverwendet werden können. Es besteht aber die Möglichkeit der Wiederverwendung aller von DHE mitgelieferten BASIC-Module sowohl in der OERR-Anwendung als auch in weiteren Anwendungen, auch wenn dies sich als Nachteil auf die `Effizienz` auswirkt (siehe nächstes Kapitel).

Außer der *Allgemeingültigkeitskenngröße* kann die `Wiederverwendbarkeit` zusätzlich durch folgende Prüfliste bewertet werden:

F_1: Wurden beim Entwurf Datenkapselungen oder abstrakte Datentypen verwendet?

F_2: Sind Ein-/Ausgabe-Funktionen in speziell dafür vorgesehenen Modulen implementiert?

F_3: Sind anwendungs- und hardwarebezogene Funktionen in voneinander unabhängigen Modulen implementiert?

Alle drei Fragen können aus DHE-Sicht positiv (1) beantwortet werden. Die Ein-/Ausgabe-Funktionen werden ebenfalls in den DHE-BASIC-Moduln definiert. Hardwarebezogene Funktionen werden in OERR nicht implementiert und sind in DHE für den Entwickler transparent (vgl. die technologische Plattform der DHE-Struktur in Abbildung 7).

6.4.3 Auswertung der Ergebnisse

Die Eigenschaften, die in der Konstruktionsphase validiert wurden, beziehen sich sowohl auf den Entwicklungsprozeß als auch auf die DHE-Software und das implementierte OERR-System. Was den Entwicklungsprozeß anbetrifft, kann erneut, so wie in der Analysephase auch, die positive Bewertung der Durchführung der Konstruktionsphase durch den Einsatz der OOSE-Methode festgestellt werden (vgl. die Bewertung der *Funktionalen Vollständigkeit*).

Die Validierung des Entwurfs und der Implementierung der OERR-Anwendung kann grundsätzlich positiv bewertet werden. Nachteile ergeben sich bei der Portierung eines Teils des damals schon implementierten OERR-Systems in eine 32-bit Umgebung. Große Vorteile in der Benutzung der DHE-Software für die Entwicklung der OERR-Anwendung ergeben sich aus der `Sicherheit`, `Effektivität` und `Wiederverwendbarkeit`. Außerdem stellt DHE eine Vielzahl von Diensten zur Verfügung, so daß die Implementierung aller charakteristischen OERR-Vorgänge mit Hilfe dieser Dienste möglich ist (Merkmal: *Funktionale Vollständigkeit*).

6.5 Dauer der Tätigkeiten der Konstruktionsphase

Die Konstruktionsphase ist der Kern eines jeden Entwicklungsprozesses, in den der größte Aufwand auf Seiten der Entwickler liegt. Durch den Einsatz der OOSE-Methode kann jedoch eine Verminderung der Entwicklungszeit erzielt werden. Weiterhin ergibt die Benutzung der DHE-Software eine wesentliche Vereinfachung der Aufstellung des Entwurfs- und des Implementierungsmodells, indem die Datenbankverwaltung sowie die Herstellung der Datenbankkonsistenz von DHE übernommen wird und nicht in der OERR-Anwendung entworfen und implementiert werden muß.

Die Konstruktion von Software-Systemen, die auf DHE basieren, bringt demnach eine wesentliche Zeitverminderung gegenüber konventionelle Software-Systemen, in denen insbesondere die Datenbankanbindung sehr aufwendig ist, mit sich.

7 Die Testphase

Die Testphase ist die letzte Phase des Entwicklungsprozesses eines Software-Systems. Diese Phase beinhaltet auch die Wartung und Erweiterung. Letztere werden in dieser Arbeit nicht betrachtet. In diesem Kapitel wird, wie auch bei den vorigen zwei Phasen, das Validierungskonzept aus Kapitel 4.3 auf die Phase *Test* angewendet.

7.1 Beschreibung der Entwicklungsphase *Test*

Während der gesamten Konstruktionsphase ist der Anwender nur unmittelbar am Fortgang des Projektes beteiligt. Dieses kann zwar, wie erwähnt, durch die Aufstellung von verschiedenen Modellen verbessert werden, jedoch ist eine abschließende Testphase notwendig.

7.2 Input- und Output-Parameter der Entwicklungsphase *Test*

Als Eingabe für die Durchführung der Testphase des entwickelten OERR-Systems dient neben dem Entwurfs- und dem Implementierungsmodell auch das Anforderungsmodell der Analysephase (vgl. Abbildung 36). Die Grundlage hierfür bilden die vom Anwender formulierten Anwendungsfälle aus dem Anforderungsmodell. Jeder dieser Anwendungsfälle muß sich mit, aus Anwendersicht vertretbarem Aufwand mit dem System realisieren lassen. Es sei darauf hingewiesen, daß sich an dieser Stelle auch Hinweise auf fehlende Funktionalitäten ergeben können. In diesem Fall ist ein erneuter Durchlauf der Methode zur Erweiterung des bestehenden Systems notwendig (vgl. [JCJ+93]).

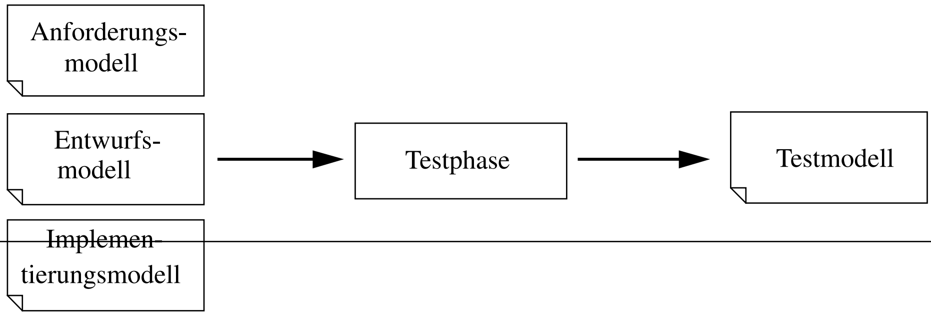

Abbildung 36: Input- und Output-Modelle der Testphase

Die Testphase wird bei Jacobson et al. (vgl. [JCJ+93]) in drei Schritten durchgeführt (*Modultest*, *Integrationstest* und *Systemtest*) und hat als Ergebnis das *Testmodell*.

7.3 Aufstellung des Phasen-Enddokumentes

In diesem Kapitel wird die Vorgehensweise bei der Aufstellung des Testmodells erläutert. Weiterhin wird das Testmodell für den Anwendungsfall `Anforderung generieren` aufgestellt.

Folgende Vorgehensweise wird beim Testen der entwickelten OERR-Anwendung angewendet:

1. Eingabe von Testfalldaten in DHE;
2. Modultest;
3. Integrationstest;
4. Systemtest.

Diese vier Schritte werden nun einzeln erläutert:

7.3.1 Eingabe von Testfalldaten in DHE

Testen ist ein dynamisches Prüfverfahren, in dem das Prüfobjekt ausgeführt wird. Für die entwickelte OERR-Anwendung bedeutet Testen die Überprüfung des Programms mit einer stichprobenartig ausgewählten Menge von Eingabewerten. Dabei wird geprüft, ob sich das Programm so verhält, wie es in der Spezifikation gefordert wird.

Um die Funktionalität der Formulare aus der OERR-Anwendung testen zu können ist es notwendig, eine Menge von Testdaten mit Hilfe der DHE-Dienste in die Sybase-Datenbank einzufügen. DHE bietet zwei Möglichkeiten Daten als Testdaten in die Datenbank einzufügen:

a.) Laden von Daten aus einer externen Datei;

b.) Laden von Daten mittels des DHE „Test“-Clients.

Diese beiden Alternativen werden konkret anhand der entwickelten OERR-Anwendung im folgenden beschrieben:

a.) Laden von Daten aus einer externen Datei:

Eine Möglichkeit, Daten in DHE bzw. in der integrierten Datenbank von DHE zu importieren, besteht in der Implementierung einer C-Funktion, die Daten aus einer ASCII-Datei einliest und die entsprechenden DHE-Dienste für das Einfügen dieser Daten in der Datenbank aufruft. Dieser Aspekt führt zu Überlegungen in Richtung einer Verknüpfung der entwickelten OERR-Anwendung mit dem bestehenden proCOM-Stationssystem der Firma ROKD GmbH, zumal diese Anwendung im Mathias-Spital im Einsatz ist. So könnten beispielsweise die in proCOM eingetragenen Aufenthalte in den Belegungsplan der OERR-Anwendung (vgl. Abbildung 33) geladen werden.

Probleme traten jedoch auf, als festgestellt wurde, daß DHE-Dienste Pflichtfelder besitzen, für die proCOM keine Werte generiert. Diese Felder müssen gefüllt werden, sonst liefert DHE eine Fehlermeldung. Weiterhin wurde festgestellt, daß manche Feldformate (Länge etc.) aus proCOM nicht mit denen aus DHE übereinstimmt. Beispielsweise hat in proCOM das Feld Wohnort eine Länge von 32 Zeichen, während DHE für dieses Feld nur 31 Zeichen zur Verfügung stellt. Änderungen wären auch in den Feldern *Geburtsdatum* und *Geschlecht* notwendig gewesen.

Ein weiteres Hindernis in der Übernahme von Daten aus proCOM besteht in dem großen Aufwand der Übernahme einer Anzahl von proCOM-Katalogen (wie beispielsweise der Postleitzahlen-Katalog, der Ort-Katalog, der Stationen-Katalog etc.) in DHE. Dies wäre notwendig gewesen, um beispielsweise mit eindeutigen Identifikationsummern der Stationen aus proCOM weiter in DHE bzw. in der OERR-Anwendung arbeiten zu können.

Prinzipiell ist es möglich, Daten aus einer schon vorhandenen externen Datei in DHE zu importieren. Wegen dem relativ großen Aufwand, machte dies jedoch nur für die Entwicklung charakteristischer OERR-Vorgänge wenig Sinn.

b.) Laden von Daten mittels des DHE „Test“-Clients:

Eine zweite Möglichkeit, Daten in DHE zu laden, ist die manuelle Eingabe mittels des von DHE zur Verfügung gestellten „Test“-Client-Programms `dhe_T` (vgl. [Fer96a]). Dieses Programm erlaubt die Ausführung der DHE-Dienste interaktiv auf dem Server, ohne dafür eine Anwendung auf der Client-Seite schreiben zu müssen. So können durch direkten Aufruf der entsprechenden DHE-Dienste die Testdaten in die Datenbank manuell eingefügt werden.

Um festlegen zu können, welche Daten für das Testen der Funktionalität der entwickelten OERR-Anwendung in DHE eingegeben werden müssen, soll das in Abbildung 27 dargestellte Diagramm als Orientierung dienen. Um beispielsweise das Formular `frmAnfGen` („Anforderung generieren“ - vgl. Abbildung 34) auf seine Funktionalität testen zu können, ist es notwendig u.a. Daten mittels folgender DHE-Dienst-Aufrufe in DHE einzufügen:

`pmPA__M`: Patient einfügen; *Beispiel: Hertling Peter, Brattka Sabine*;
`pmCC__M`: Aufenthaltstyp einfügen; *Beispiel: stationär, ambulant*;
`pmTT__M`: Aufenthalte einfügen; *Beispiel: Hertling Peter auf die Innere Station*;
`amTA__M`: Aktivitätsklassen einfügen; *Beispiel: EKG, Röntgen, Labor*;
`amAY__M`: Aktivitätstyp einfügen; *Beispiel: Ruhe-EKG, Röntgen-Zervikal*;
`amAS__M`: Aktivitätstyp durchgeführt von Agent; *Beispiel: Ruhe-EKG wird von dem EKG-Funktionsbereich durchgeführt*.

Im Kapitel E4.1 befinden sich Angaben zu weiteren Testdaten, die in DHE eingefügt worden sind, um die Funktionalität aller entwickelten Formulare testen zu können.

Im folgenden wird exemplarisch beschrieben, wie das Formular `frmAnfGen` getestet wird. Das gesamte Testmodell (Modultest, Integrationstest und Systemtest) mit den Testergebnissen befindet sich im Kapitel E4.1.

7.3.2 Modultest

Unter *Modultest* wird der Test der kleinsten Programmiereinheiten, der Module, verstanden (vgl. [Wal90]). Er findet vor der Integration der Module zu größeren Einheiten statt. Typische Testaspekte, die für die Aufstellung von Testfällen relevant sind, umfassen:

- Funktionen des Moduls;
- die Modulstruktur;
- Ausnahmebedingungen, Sonderfälle etc.;
- Performance.

Für die entwickelte OERR-Anwendung bedeutet Modultest das Testen der einzelnen Formulare, die in VB implementiert wurden, anhand der oben genannten Aspekte.

Nach der manuellen Eingabe der im Kapitel 7.3.1 erwähnten Testdaten ist es möglich, das Formular `frmAnfGen` aus dem *Belegungsplan* zu laden, mittels des Menüpunktes `Anforderung->Anforderung generieren->EKG`. Nun sollen die in diesem Formular vorgesehenen Interaktionsformen (*ListBox*, *ComboBox*, *Labels* etc.) die entsprechenden DHE-Daten ausgeben. Beispielsweise soll die *ComboBox* `cboLeistKE` aus Abbildung 32 die Liste der KE's darstellen, die eine EKG-Anforderung durchführen. Diese wären laut eingefügten Testfalldaten die Innere Station, die Chirurgie, die Geburtshilfe und die Gynäkologie sowie der EKG-Funktionsbereich. Das Ergebnis dieses durchgeführten Tests ist negativ. Die Begründung wird im folgenden erläutert: in DHE werden als Testdaten die Aktivitätsklasse EKG mit 17 dazugehörigen Aktivitätstypen eingegeben. Beispiele für Aktivitätstypen der Klasse EKG sind: Ruhe-EKG, Rhythmus-EKG, Belastungs-EKG etc. Jedem Aktivitätstyp wird in DHE eine leistende KE zugeordnet, die diesen Aktivitätstyp durchführen kann. So wird auch in diesem Fall eingegeben, daß jeder der 17 Aktivitätstypen von folgenden KE's durchgeführt werden können: Innere Station, Chirurgie, Geburtshilfe und Gynäkologie, EKG-Funktionsbereich. Als Folge werden in die *ComboBox* `cboLeistKE` die aufgezählten Stationen und Funktionsbereiche jeweils 17mal, einmal für jeden Aktivitätstyp, ausgegeben, statt jeweils nur ein Mal. Es ist demzufolge in DHE nicht möglich, einer Aktivitätsklasse die dazugehörenden leistenden KE's zuzuordnen, sondern nur den Aktivitätstypen. Die eingegebene Klassifizierung der Aktivitätstypen nach Klassen bzw. nach leistenden KE's kann nicht in derselben Form von DHE für die Ausgabe verwendet werden.

Das Formular `frmAnfGen` ist nicht das einzige, das Einschränkungen in seiner Funktionalität aufweist. Weitere Fehler wurden in dem Formular `frmDetails` („EKG-Anforderung generieren-Details“), `frmArztablage` („Arztablage“) und `frmAnfVer` („Anforderung verifizieren“) beim Testen entdeckt. Es handelt sich dabei um den Aufruf folgender DHE-Dienste:

- `amAA__M`: Einfügen einer Einzelleistung in DHE;
- `amAA__L`: Ausgabe einer Liste mit allen Einzelleistungen, die den Status „vorgeschlagen“ haben;
- `amAA__R`: Ausgaben einer Einzelleistung zwecks Verifizierung;
- `amReqAct`: Anfordern einer Einzelleistung.

Zusammenfassend kann die Schlußfolgerung gezogen werden, daß eine eingeschränkte Funtionalität der Module beim Modultest aufgewiesen werden konnte. Die mäßigen Ergebnisse des Modultests wird auf fehlerhafte Dienste der DHE-Version 1.00a zurückgeführt.

7.3.3 Integrationstest

Nachdem die ersten Formulare freigegeben sind, kann mit der Integration der Formulare zu größeren Einheiten begonnen werden. Beim *Integrationstest* wird davon ausgegangen, daß die Formulare einzeln sehr gut getestet sind und nun im wesentlichen das Testen der Formularschnittstellen und das Zusammenwirken der Formulare (Formularkommunikation) durchgeführt wird (vgl. auch [Wal90]).

Eng mit der Problematik des Integrationstests verbunden ist die Frage, in welcher Weise die verschiedenen Formulare integriert werden. Die Reihenfolge der Integration der Formulare kann aus dem in Abbildung 37 aufgestellte Dialognetz abgeleitet werden. Dieses stellt eine graphische Beschreibung der Dialog- bzw. Formularabläufe in dem entwickelten OERR-System dar. Zusätzlich sind in dieser Abbildung diejenigen DHE-Dienste dargestellt, die in der DHE-Version 1.00a fehlerhaft sind und somit zu einer Einschränkung in der Integration der Formulare des entwickelten OERR-Systems führen.

Kurz zusammengefaßt können wegen fehlerhafter DHE-Dienste in der DHE-Version 1.00a folgende OERR-Funktionalitäten zwar implementiert werden, der Integrationstest ergibt jedoch negative Ergebnisse: die *Maske_Details* im System der anfordernden KE enthält einen Button `OK,` dessen Betätigung die Details einer Einzelleistung mittels des Dienstes `amAA__M` in DHE speichern soll. Die *Maske_Arztablage*, ebenfalls im OE-System der anfordernden KE, soll, durch Betätigung des Buttons `ARZTABLAGE` vom Belegungsplan aus, eine Liste mit allen Einzelleistungen ausgeben, die der Stationsarzt verifizieren muß, ausgeben. Dafür wird der DHE-Dienst `amAA__L` aufgerufen. Derselbe Dienst wird auch für die Auflistung der schon durchgeführten und dokumentierten Einzelleistungsergebnisse, durch die Betätigung des Buttons `ERGEBNISSE` vom Belegungsplan und durch den Aufruf der Schwesternablage (Button `SCHWESTER`), aufgerufen. Eine weitere Einschränkung der OERR-Funktionalität befindet sich in dem fehlerhaften Dienst `amReqAct`, welches für die Anforderung einer Aktivität zuständig ist (in diesem Fall die Anforderung von Einzelleistungen).

Generell kann aus Abbildung 37 festgestellt werden, daß die DHE-Dienste `amAA` (`amAA__L` und `amAA__M`), sowie `amReqAct` fehlerhaft sind. Der Dienst `hmXA` (`hmXA__M`) ist zwar nicht fehlerhaft, kann aber wegen Speichereinschränkungen von VB3 nicht in dem `OERR.MAK`-Projekt eingefügt werden.

7.3.4 Systemtest

Beim *Systemtest* geht es darum, das Gesamtsystem einerseits hinsichtlich seiner funktionalen Leistungen und andererseits hinsichtlich der Grenzen seiner Leistungsfähigkeiten zu testen. Beim Systemtesten werden auch Aspekte wie die Belastung der verfügbaren Hardware-Einsatzumgebung und Integration in die Benutzerorganisation in Betracht gezogen.

Das OERR-System wird hinsichtlich seiner funktionalen Leistungen getestet. Das Zusammenwirken der Formulare ist durch Anklicken der verschiedenen Buttons auf der Formularoberfläche nachvollziehbar. So bewirkt beispielsweise das Klicken auf den Button `DETAILS` in dem Formular `frmAnfGen` (vgl. Abbildung 34) das Öffnen des Formulars `frmAnfDe` („EKG-Anforderung generieren-Details“) (vgl. Abbildung 35). Die im Kapitel 7.3.2 festgestellten Fehler beim Modultesten bewirkt sich auch auf das Systemtest. Um jedoch weiter testen zu können, wurde der Quellcode an die entsprechenden Stellen geändert und einige Variablen mit Konstanten belegt. Dies erkennt man im Quellcode durch den Kommentar „Zwischenlösung“.

Außer den funktionalen Leistungen werden beim Systemtest auch die Grenzen der Leistungsfähigkeit der entwickelten Anwendung getestet. Um dies möglich zu machen, sind oft riesige Testmengen bereitzustellen. In DHE wurde aber nur eine kleine Menge an Testdaten eingefügt (vgl. Kapitel 7.3.1). Für die Bereitsstellung umfangreicherer Testdatenmengen eignen sich Simulatoren oder Testdatengeneratoren.

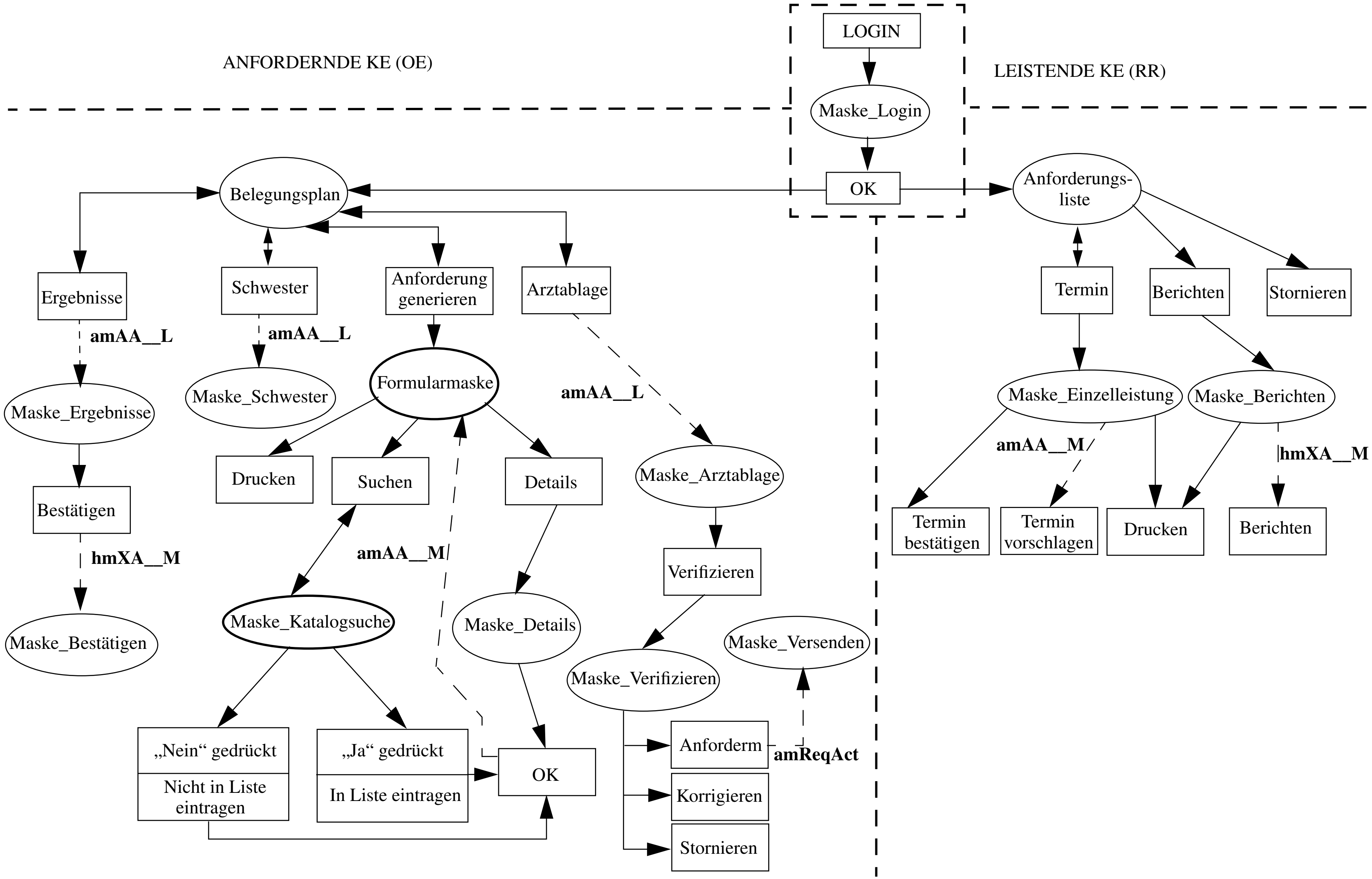

Abbildung 37: Integrationstest-Ergebnisse anhand des aufgestellten Dialognetzes

7.4 Validierungstätigkeiten in der Testphase

Die Validierung der Testphase umfaßt sowohl die Bewertung der eingesetzten Methode in dieser Phase, als auch die Auswertung des Gesamtsystems. Die Bewertung der Testphase erlaubt zum ersten Mal im Entwicklungsprozeß Aussagen einerseits über die gesamten Leistungen und Leistungsfähigkeiten der entwickelten OERR-Anwendung und andererseits über mögliche Hilfen, die DHE und VB beim Testen zur Verfügung stellen.

7.4.1 Validierungsziele und Auswertungsmethoden

Für das Testen sind folgende Prüfziele zu berücksichtigen (vgl. [Wal90]:

Qualitätseigenschaft	Qualitätsmerkmal
Funktionsabdeckung	• Funktionale Vollständigkeit
Korrektheit	• Erreichter Testgrad
Effizienz	• Speichereffizienz
	• Ausführungseffizienz
Erlernbarkeit	• Lernzeit für die Nutzung der DHE-Software
	• Voraussetzungen für die Erlernbarkeit von DHE
	• Bewertung der Qualität der DHE-Schulungs- / -Lernunterlagen
	• DHE-Hilfe- / -Tutorialsysteme
Erweiterbarkeit	• Flexibilität der Struktur
Korrigierbarkeit	• Fehlerfindung
Prüfbarkeit	• Zugänglichkeit des Produktes (DHE und/oder OERR)
Verknüpfbarkeit	• Systemtest unter realistischen Ausführungsbedingungen
	• Schnittstellentest
Zuverlässigkeit	• Einsatz des Software-Produktes (DHE und OERR)
	• Verfügbarkeit des Software-Produktes (DHE und OERR)

Die Definition dieser SPARDAT-Qualitätseigenschaften können zusammen mit den dazugehörigen Auswertungsmethoden dem Kapitel E4.2 entnommen werden.

7.4.2 Durchführung der Validierung

In diesem Kapitel werden exemplarisch fünf der acht aufgelisteten Qualitätseigenschaften validiert. Diese sind: `Funktionsabdeckung`, `Effizienz`, `Korrigierbarkeit`, `Verknüpfbarkeit` und `Zuverlässigkeit` und wurden ausgewählt, da deren Validierung sowohl die Implementierungsumgebung des OERR-Systems als auch direkt die DHE-Software berücksichtigt. Die Validierung aller in der Testphase ausgewerteten Eigenschaften befindet sich im Kapitel E4.2.

Funktionsabdeckung

Das Merkmal *Funktionale Vollständigkeit* der Eigenschaft `Funktionsabdeckung` wird in der Testphase mit Hilfe der folgenden Frage validiert (die Fragen F_1 bis F_9 wurden in vorigen Phasen beantwortet):

F_{10}: Wurde jede Funktion (DHE-Dienst oder VB3-Ereignisprozedur) mit mindestens einem Testfall getestet?

Bezogen auf die entwickelte OERR-Anwendung bedeutet der Funktionstest das Testen jedes DHE-Aufrufes, welcher innerhalb eines VB-Formulars benutzt wird. Dafür werden entsprechende Testdaten in DHE mittels des DHE-Testprogramms `dhe_T` eingefügt (vgl. Kapitel 7.3.1). Für das Testen der Ausgaben der aufgerufenen DHE-Dienste kann leicht festgestellt werden, ob die aufgetretenen Fehler DHE-bedingt sind oder ob die entwickelte OERR-Anwendung auf der Client-Seite von DHE diese Fehler versursacht hat. Dieses Aspekt der Fehlerfindung wird im Rahmen der Eigenschaft *Korrigierbarkeit* beschrieben.

Mit den von DHE und VB zur Verfügung gestellten Werkzeuge, kann jede implementierte Funktion aus der OERR-Anwendung mit mindestens einem Testfall getestet werden. Die Antwort auf die Frage F_{10} ist „Zutreffend“ (1).

Effizienz

Im Rahmen der Validierung der `Effizienz` werden in der Testphase die Merkmale *Speichereffizienz* und *Ausführungseffizienz* betrachtet. Die Validierung der *Speichereffizienz* erfolgt anhand folgender Prüfliste:

F_1: Findet eine Speicherplatzoptimierung statt?

F_2: Sind zur Laufzeit möglichst wenige Teile des Objektcodes zur selben Zeit im Hauptspeicher?

Das Speicherproblem, welches bei der Eigenschaft `Korrektheit` im Kapitel 6.4.2 kurz erwähnt wurde, wird hier näher betrachtet: es wurde festgestellt, daß nicht alle notwendigen Funktions- und Datenstrukturdeklarationen (d.h. die DHE-BASIC-Module - vgl. Kapitel 6.3.2.1) im VB-Projekt `OERR.MAK` geladen werden konnten. Der Grund ist die geringe Speicherkapazität von VB3 in einer 16-bit-Umgebung. Es können in einem VB3-Projekt nicht beliebig viele Deklarationen definiert werden. Auch die Einzelleistungsergebnisse eines Patienten, die ebenfalls als Gesundheitsdaten von DHE gespeichert werden, können mit den entsprechenden DHE-Dienst-Aufrufen wegen geringer Speicherkapazität nicht verwaltet werden. Der Teil des entsprechenden Quellcodes (vgl. Kapitel E3.2 - Ereignis `cmdArztablage_Click()` und `cmdErgebnisablage_Click()`) ist deshalb auskommentiert.

In VB3 findet keine Speicherplatzoptimierung statt. Frage F_1 wird mit „Unzutreffend“ beantwortet. Es werden nach den obigen Betrachtungen viele Teile des Objektcodes zur selben Zeit im Hauptspeicher gehalten, so daß Frage F_2 auch mit „Unzutreffend“ beantwortet werden kann.

Es kann die Schlußfolgerung gezogen werden, daß eine Entwicklung von DHE-basierten Anwendungen unter VB3 in einer 16-bit Umgebung, so wie diese im Rahmen dieser Arbeit durchgeführt wurde, eingeschränkt ist. Die aktuelle Version von Visual Basic (5.0) in einer 32-bit Umgebung führt nicht mehr zu diesen Einschränkungen.

Die *Ausführungseffizienz* der entwickelten OERR-Anwendung kann mittels der Ausführungszeit der DHE-API-Aufrufe bewertet werden. Die Ausführungszeit eines jeden aus der Anwendung aufgerufenen DHE-Dienstes wird von DHE auf der Server-Seite in der Datei `dheprod.log` registriert. Dieser Aspekt wurde im Kapitel 6.4.2 erläutert. Ein wesentlicher Aspekt der *Ausführungseffizienz* des OERR-Systems besteht in der Ausführungszeit der Dienste `dhCM` und `dhSE`, welche eine Kommunikation zwischen der anfordernden und leistenden KE initiieren. Hier ist es wichtig, daß eine auf der anfordernden KE generierte und gesendete Anforderung schnell für die leistende KE zur Verfügung steht.

Es soll noch hinzugefügt werden, daß die entwickelte OERR-Anwendung in einer nicht-realen Krankenhausumgebung getestet wurde, wo ein relativ kleiner Datenbestand und nur zwei Clients vorhanden sind. Es ist schwer nachzuvollziehen, wie sich die Ausführungszeiten ändern würden, wenn mehr als ein Client zur selben Zeit Zugriffe auf die Datenbank ausübt und große Datenbestände vorhanden sind. Die Bewertung der Ausführungszeiten in einer solchen realen Krankenhausumgebung wird in dieser Arbeit nicht durchgeführt.

Korrigierbarkeit

Die SPARDAT-Qualitätseigenschaft `Korrigierbarkeit` kann zum ersten Mal in der Testphase validiert werden. Die `Korrigierbarkeit` ist die Eigenschaft, möglichst einfach Fehler in einem Produkt zu finden und zu beheben. Die `Korrigierbarkeit` läßt sich durch das Merkmal *Fehlerfindung* charakterisieren. Folgende Prüfliste wird für die Validierung dieses Merkmals aufgestellt:

F_1: Sind Fehlermeldungen einer eindeutigen Fehlersituation zuordbar?

Die OERR-Anwendung wurde so implementiert, daß sie zwischen Fehlern aufgrund der DHE-Software und Benutzerfehlern unterscheiden werden kann (vgl. auch Kapitel 6.4 - Eigenschaft `Sicherheit`). Jede von OERR ausgegebene Fehlermeldung enthält Hinweise auf mögliche Fehlerursachen. Falls ein Fehler infolge eines DHE-Dienstaufrufes zur Laufzeit der Anwendung auftritt, so kann in VB eine entsprechende Fehlermeldung implementiert werden, die auch die DHE-Fehlernummer bzgl. eines aufgerufenen Dienstes enthält. Diese Nummer kann dann in der DHE-Dokumentation nachgeschlagen werden, um die möglichen Ursachen festzustellen.

Es gibt Fehlersituationen, in denen Eingabefehler des Benutzers der OERR-Anwendung die Ursache sind. Diese Situationen werden durch entsprechend ausführlich beschriebene Hinweise für den Benutzer gelöst, damit eine möglichst schnelle Aufhebung des entstandenen Benutzerfehlers erfolgen kann. Die erste Frage des Merkmals *Fehlerfindung* kann somit mit „Zutreffend“ (1) bewertet werden.

F_2: Gibt es Zusatzinformationen, die im Fehlerfall zur Analyse herangezogen werden können (Analyseprotokolle)?

Zusatzinformationen zu einem aufgetretenen Fehler können einerseits aus den von der Anwendung ausgegebenen Fehlermeldung bestimmt werden (DHE-Fehlernummer). Eine zweite Möglichkeit, Fehler schnell zu finden, bietet jedoch auch der DHE-Server: in der Datei `dheprod.dmp` werden die aus der OERR-Anwendung aufgerufenen DHE-Dienste mit Ein- und Ausgabeparametern von DHE protokolliert. Dieses Analyseprotokoll gibt Aufschluß darüber, ob aufgetretene Fehler DHE-bedingt sind oder in der entwickelten Anwendung auf der Client-Seite auftreten. Folgendes Beispiel wird betrachtet:

In Abbildung 38 ist das Formular `frmAnfVer`, die dem Anwendungsfall `Anforderung verifizieren` zuzuordnen ist, dargestellt. In diesem Beispiel handelt es sich um die Verifizierung der Einzelleistung *Ruhe-EKG* die für den Patienten *Hertling Peter* generiert wurde. Beim Laden dieses Formulars wird der DHE-Dienst `amAA__R` mit dem Eingabeparamter

`is_act.aa__icode = „ 0000001" + Chr$(0)`

aufgerufen. Dieser Dienst hat als Ausgabe alle Informationen, die für die generierte Einzelleistung *Ruhe-EKG* (`aa_icode= „ 0000001"`) in DHE gespeichert wurden. Diese Informationen beinhalten Namen und Vornamen des Patienten, für den die Einzelleistung generiert wurde, sein Geburtsdatum, die Aufenthalts-ID, die Priorität der Durchführung der Einzelleistung, den vorgeschlagenen Termin etc. Diese Informationen werden mittels der entsprechenden VB-Interaktionsformen (*Textbox*, *Labels* etc.) im obigen Formular ausgegeben. Bei einer genaueren Betrachtung des Formulars kann festgestellt werden, daß die ersten drei Zeichen in den Ausgabe-Interaktionsformen fehlen. Zum Beispiel müßte das *Label* für die Ausgabe des Namens und Vornamens, den Namen des Patienten (*Hertling*) enthalten. Fehlende Zeichen gibt es auch im Gruppenfeld *Kommunikationseinheiten* sowie *Anforderung*.

Um feststellen zu können, ob dies ein DHE-bedingter Fehler ist, oder ob die Ursache dieses Fehlers auf der Client-Seite zu suchen ist, wird der Inhalt der Datei `dheprod.dmp` untersucht. Ein Ausschnitt aus dieser Datei, die die Ein- und Ausgabeparameter des analysierten DHE-Dienstes `amAA__R` enthält, kann in Abbildung 39 betrachtet werden.

Im ersten Teil der Datei sind die Parameter zu sehen, die der Client durch die OERR-Anwendung beim Aufruf des Dienstes `amAA__R` dem DHE-Server gesendet hat (`Dump of buffer transmitted by the client)`. Einer dieser Parameter ist die ID der Einzelleistung, für die weitere Informationen von der Anwendung verlangt werden (ID = 0000001, in Abbildung 39 fett markiert). Das Backslash-Zeichen „\“ trennt die einzelne Aufrufparameter dieses Dienstes.

Im zweiten Teil der Datei `dheprod.dmp` (`Dump of buffer transmitted by the server`) werden nun die Ausgabeparameter des DHE-Servers aufgelistet. Interessant sind die Felder, in denen der Name des Patienten und die Beschreibung der Einzelleistung aufgelistet sind (in Abbildung 39 ebenfalls fett markiert). Man sieht, daß sowohl der Name des Patienten (Hertling) als auch die Beschreibung der Einzelleistung (Ruhe-EKG) von DHE korrekt ausgegeben werden. Der DHE-Dienst `amAA__R` funktioniert also korrekt.

Da sich die Datei `dheprod.dmp` auf dem DHE-Server befindet, kann der im Formular `frmAnfVer` aus Abbildung 38 aufgetretene Fehler entweder auf die Kommunikation zwischen dem Server und dem Client oder aber auf die Client-Seite eingegrenzt werden. Eine Kommunikation zwischen dem DHE-Server und -Client findet bei jedem DHE-Dienst-Aufruf statt. Es ist also unwahrscheinlich, daß der Fehler beim Aufruf bzw. Ausgabe des Dienstes `amAA__R` in der Server-Client-Kommunikation liegt (es würde sonst bei jedem DHE-Aufruf auftreten). Eine mögliche Ursache des aufgetretenen Fehlers könnte in dem schon erwähnten Speicherproblem liegen: es ist nicht mehr genügend Speicherplatz vorhanden, um alle Daten im Formular darzustellen.

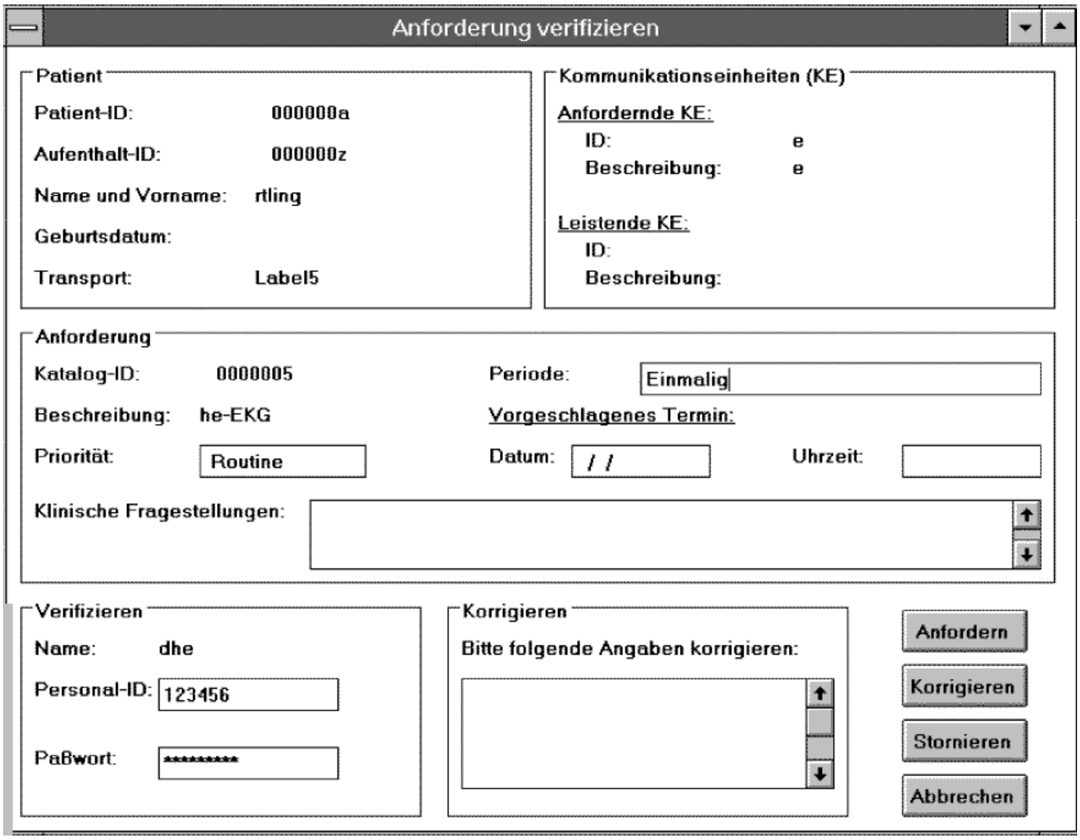

Abbildung 38: Das Formular `frmAnfVer` („Anforderung verifizieren“)

```
1997/08/20 17:11:53---- Service  amAA__R.  Dump of buffer transmitted by the client
Header. id [] length:[       142]  client:[79781ec14697]  action:[0] trimmed:[0] msg:[0]
------ dump of user buffer --------
\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
\\      0000001\
------ end of dump --------

1997/08/20 17:11:53---- Service  amAA__R.  Dump of buffer transmitted by the server
Header. lenght:[      2405] action:[0] trimmed:[0] msg:[0]
------ dump of user buffer --------
0000001\
N0001000000090291\
1997/01/29\
14:43:11\
dhe        \
dhe_unit    \
1997/02/25\
10:16:57\
dhe        \
dhe_unit    \
N \
522133279\ \
0000005\
10010       \
Ruhe-EKG        \
Ruhe-EKG                                          \
LN      000000a\
Hertling                              \
Peter                                    \
M1965/09/12\
000000z\
1     0000001\
S           \
Stationaer      \
Stationaeres Aufenthalt                        \
dhe        \
dhe_unit    \
------ end of dump --------
```

Abbildung 39: Ausschnitt aus der Datei `dheprod.dmp`

Solche Fehleranalysen werden also sehr gut von DHE unterstützt und helfen dem Entwickler möglichst schnell die Fehlersuche einzugrenzen. Frage F_2 kann ebenfalls mit „Zutreffend“ (1) beantwortet werden.

F_3: Gibt es Hilfsmittel für die temporäre Fehlerbeseitigung?

Die temporäre Beseitigung von Fehlern wird auch in der entwickelten OERR-Anwendung angewendet, um die weitere Entwicklung des Systems beim Auftreten von DHE-bedingten Fehler nicht zu verzögern. Wie erwähnt, wurden Zwischenlösungen implementiert, in dem Aufrufparamter (Variablen) mit Konstanten belegt wurden.

Spezielle Hilfsmittel für temporäre Fehlerbeseitigung bietet sowohl DHE als auch VB3 nicht, im Vergleich zu VB5, welches hinsichtlich dieses Aspektes mehr Möglichkeiten bietet.

Die Frage F_3 wird mit „Unzutreffend“ (0) bewertet.

F_4: Gibt es Hilfsmittel bei der Kompilierung, die mögliche Syntax-, Datentyp- und Schnittstellenfehler prüfen?

Visual Basic bietet das sogenannte *Debuggen im Einzelschritt* an, in dem das Ergebnis der Ausführung einer jeden Code-Zeile sequentiell betrachtet werden kann. Die Validierung bezieht sich hier weniger auf OERR oder DHE, sondern eher auf die Entwicklungsumgebung.

Die Frage F_4 wird positiv (1) beantwortet.

F_5: Gibt es eine Testumgebung, damit einzelne Module isoliert prüfbar sind?

In Visual Basic können ohne viel Aufwand Testumgebungen entwickelt werden, um einzelne Formulare isoliert prüfen zu können. DHE bietet auch durch das beschriebene `dhe_T` - Programm die Möglichkeit, DHE-Dienst-Aufrufe einzeln zu prüfen. Frage F_5 kann positiv (1) bewertet werden.

Verknüpfbarkeit

Eine weitere Qualitätseigenschaft, die in der Testphase validiert wird, ist die `Verknüpfbarkeit`. Die `Verknüpfbarkeit` ist die Eigenschaft eines Produktes, dieses mit anderen Produkten über Schnittstellen zu verbinden bzw. zu betreiben. Durch die Auswertung der `Verknüpfbarkeit` soll u.a. festgestellt werden, in welchem Ausmaß die Verträglichkeit der Schnittstellen gewährleistet ist, bzw. ob Konversionsprozeduren zum Transfer von Daten in andere Systeme vorhanden sind.

Im Hinblick auf diese Arbeit wird die `Verknüpfbarkeit` aus interner und externer Sicht betrachtet: aus der internen Sicht wird die Kommunikation zwischen den zwei Teilanwendungen von OERR (OE und RR) innerhalb von DHE ausgewertet. Die externe Sicht, aus der die `Verknüpfbarkeit` betrachtet wird, ist der im Kapitel 7.3.1 beschriebene Datenaustausch zwischen einem bereits existierenden Sytem, das nicht auf DHE basiert (das proCOM-Stationssystem der ROKD GmbH) und der unter DHE entwickelten OERR-Anwendung. Diese Verknüpfung wurde aus den dort genannten Gründen in dieser Arbeit nicht weiter vertieft.

Die Eigenschaft `Verknüpfbarkeit` läßt sich, so in [Wal90], durch zwei Merkmale charakterisieren: *Systemtest unter realistischen Ausführungsbedingungen* und *Schnittstellentest*. Das erste Merkmal wird durch folgende Frage validiert :

F_1: Sind alle Teilprogramme des entsprechenden Produktes ohne erkennbare Probleme gelaufen?

Diese Frage bezieht sich auf die Verknüpfung der zwei entwickelten Teilanwendungen *Order Entry* und *Result Reporting*, also auf die interne Sicht der `Verknüpfbarkeit`. In Abbildung 12 wurde beispielhaft die Verknüpfung von OE und RR mittels des DHE-Servers dargestellt. Beide Teilanwendungen stellen zwei unterschiedliche DHE-Clients dar, die auf dieselbe integrierte DHE-Datenbank durch die DHE-Dienste Zugriff haben: für jede von Client_1 angeforderte Leistung wird eine Leistungs-Identifikationsnummer mittels des TCP/IP-Kommunikationsprotokolls an den DHE-Server gesendet. Dieser initiiert eine Kommunikation zwischen der anfordernden und der von Client_1 angegebenen leistenden KE, die für die weitere Versendung von Nachrichten zwischen den beiden angegebenen KE's benutzt werden kann.

Die Verknüpfung der beiden Teilanwendungen zeigt jedoch eine Einschränkung in der Funktionsfähigkeit des Kommunikationsmechanismus der DHE Version 1.00a: die von der anfordernden KE angegebene Identifikationsnummer der leistenden KE kann nicht von DHE gespeichert werden. Dieses außergewöhnliche Verhalten, welches zu einer wesentlichen Einschränkung des gesamten Act Managements in DHE führt (vgl. auch Abbildung 26), wurde den GESI-Mitarbeitern mitgeteilt. Die Antwort darauf bestätigte diesen DHE-bedingten Fehler. Mit einem von Hand eingetragenen Wert für die Identifikationsnummer im Quellcode kann jedoch eine stabile und störungsfreie Kommunikation innerhalb von DHE Version 1.00a beobachtet werden. Dieser Fehler wurde, so die GESI-Mitarbeiter, in der DHE-Version 1.00f behoben. Die Frage F_1 muß aber für die Version 1.00a mit einer 0 („Unzutreffend“) beantwortet werden.

Das Merkmal *Schnittstellentest* der Eigenschaft `Verknüpfbarkeit` wird mittels folgender Prüfliste validiert:

F_1: Existieren Schnittstellen zu anderen Systemen? (Wenn ja, welche?)

Diese Frage bezieht sich auf die externe Sicht der `Verknüpfbarkeit` und zwar auf dem Datenaustausch zwischen dem bestehenden proCOM-Stationssystem und der DHE-basierten Anwendung. Eine durchgeführte Analyse, dessen theoretischen Ansätze im Kapitel 9.2 zusammengefaßt sind, hat ergeben, daß dieser Datenaustausch prinzipiell möglich ist.

Frage F_1 wird, unter praktischem Vorbehalt, mit „Zutreffend“ beantwortet.

F_2: Sind die Protokolle für die Kommunikation über diese Schnittstellen standardisiert?

Aus der internen Sicht der `Verknüpfbarkeit` kommunizieren die beiden Teilanwendungen OE und RR über das standardisierte TCP/IP-Protokoll. Mögliche Ansätze für die Verknüpfung existierender Systeme mit DHE-basierten Anwendungen (externe Sicht) wurden vorher geschildert.

Aus beiden Sichten kann die Frage F_2 mit 'Zutreffend' beantwortet werden.

Zuverlässigkeit

Eine letzte Eigenschaft, deren Validierung in diesem Kapitel beschrieben wird, ist die `Zuverlässigkeit` eines Produktes. Eine praxisgerechte Bewertung der `Zuverlässigkeit` orientiert sich an der Anzahl von Systemstörungen, die in einer Anwendung zur Laufzeit auftreten. Merkmale der `Zuverlässigkeit` sind der *Einsatz des Software-Produktes* sowie dessen *Verfügbarkeit*. Beide Merkmale lassen sich durch jeweils einen Bewertungsmaß auswerten.

Der *Einsatz des Software-Produktes* wird durch die *Einsatzkenngröße* ausgewertet:

$$E = \frac{S_t}{1000LOC}$$

Hierbei gilt:
E = Einsatzkenngröße;
S_t = Anzahl der Störungen pro Zeiteinheit t (t wird bei [Wal90] auf einen Monat festgelegt);
LOC = Anzahl der Quellcodezeilen (Lines of Code).

Die entwickelte OERR-Anwendung wurde nicht in einer realen Krankenhausumgebung auf `Zuverlässigkeit` getestet. Es wurden nur Tests am Entwicklungsort durchgeführt . Der DHE-Server sowie die OERR-Anwendung wurden hier über einen Zeitraum von 72 Stunden auf `Zuverlässigkeit` getestet. Während dieses Zeitraums, nach Betätigung verschiedener Buttons aus der OERR-Anwendung, die jeweils einen oder mehrere DHE-Dienstaufrufe zur Folge haben, wurde die Client-Server-Verbindung abgebrochen (DHE-Fehlermeldung 910: „`Error while closing network connection`“) und der Sybase-Server war nicht mehr aktiv. Es wurde festgestellt, daß dieses Verhalten unregelmäßig nicht immer beim selben DHE-Aufruf auftritt und somit nicht nachvollziehbar ist. Dieser Aspekt wurde ebenfalls der Firma GESI gemeldet. Es ergaben sich also drei Störungen in einem Zeitraum von 72 Stunden. Die *Einsatzkenngröße* kann nun folgendermaßen berechnet werden:

Erfahrungswerte nach [Wal90] liegen für E bei ein bis drei Fehler pro 1000 LOC in einem Zeitraum von einem Monat. In diesem Fall wären es drei Fehler in 72 Stunden, d.h. ca. 30 Fehler in einem Monat. Dies kann demnach nicht als ein gutes Ergebnis betrachtet werden. Dieser gravierende Fehler weckt Bedenken über die `Zuverlässigkeit` des OERR-Systems und der DHE-Version 1.00a.

Die *Verfügbarkeit* des Systems wird mittels der *Verfügbarkeitskenngröße* ausgewertet:

wobei:
V = Verfügbarkeitskenngröße;
T_S = Sollzeit = Zeit in der das System verfügbar sein muß;
T_D = Zeit, in der das System während der festgelegten Sollzeit nicht verfügbar ist.

Sowohl DHE als auch das OERR-System müssen rund um die Uhr im Krankenhaus dem Krankenhauspersonal zur Verfügung stehen. Da das System aber nicht in einer realen Umgebung getestet wurde, kann dieses Merkmal nicht ausgewertet werden. Es wird jedoch davon ausgegangen, daß wegen der aufgetretenen Funktionsstörungen, auch dieses Merkmal nicht zufriedenstellend ausgewertet werden kann.

Beide Bewertungsmaße haben also einen negativen Ergebnis.

7.4.3 Auswertung der Ergebnisse

Abschließend kann die Schlußfolgerung gezogen werden, daß die Validierung der Testphase nicht die erwarteten positiven Ergebnisse gebracht hat. Es können einfach und schnell Fehlerursachen gefunden werden (Eigenschaft `Korrigierbarkeit`), es ist möglich, auch wenn mit viel Aufwand, DHE mit anderen schon existierenden Systemen zu verknüpfen (Eigenschaft `Verknüpfbarkeit`), das Hauptziel eines jeden Software-Systems bleibt aber immer noch dessen `Zuverlässigkeit`, und dieses Ziel wird in diesem Fall nur teilweise erfüllt.

7.5 Dauer der Tätigkeiten der Testphase

Die Durchführung der Testphase ist von großer Bedeutung im Entwicklungsprozeß. In der Testphase wurden Daten in der Datenbank manuell eingefügt, und es wurde auch versucht, die durch Testen aufgetretenen Fehler zu analysieren und aufzuheben. Bevor ein Fehler den GESI-Mitarbeiter gemeldet wurde, wurde solange getestet, bis mit hoher Wahrscheinlichkeit sicher war, daß der aufgetretene Fehler DHE-bedingt ist und nicht von der Entwicklung des OERR-Systems abhängt. Sowohl DHE als auch die von der ROKD GmbH ausgewählte VB-Programmierumgebung, bieten jedoch Möglichkeiten an, Fehler im OERR-System schnell zu lokalisieren und zu beheben. Aufgrund detailierter Fehlermeldungen, die in DHE während der Durchführung der Testphase aufgetreten sind, konnten die DHE-bedingten Fehler nur von den GESI-Mitarbeitern aufgehoben werden und wurden in die DHE-Version 1.00f berücksichtigt.

8 Auswertung und Interpretation der Validierung

Die Validierung der SPARDAT-Qualitätseigenschaften erfolgte in den Kapiteln 5 bis 7 phasenweise: viele Eigenschaften und Merkmale ließen sich in mehr als eine Entwicklungsphase validieren (*Beispiel: das Merkmal Funktionale Vollständigkeit der Eigenschaft* `Funktionsabdeckung` *wurde in allen drei Entwicklungsphasen validiert*). Die Teilergebnisse der Validierung eines Merkmals differierte manchmal auch von Phase zu Phase. In diesem Kapitel werden die Ergebnisse der Validierung zusammengefaßt und diskutiert.

Im Kapitel 8.1 wird das Ergebnis eines jeden Validierungsziels analysiert und interpretiert. Kapitel 8.2 erläutert noch einmal in einer Zusammenfassung die Änderungen, die nach der Validierung der DHE-Version 1.00a von den GESI-Mitarbeitern vorgenommen wurden. Nicht zuletzt wird im Kapitel 8.3, auch durch die in Abbildung 40 dargestellten Eigenschaften, eine Interpretation der Validierung des Entwicklungsprozesses und somit der eingesetzten konstruktiven Qualitätssicherungselemente (V-Modell, OOSE-Methode, Dialognetze als Formalismus und VB3 als Programmierumgebung) durchgeführt. Im Kapitel 8.4 wird analysiert, inwieweit die DHE-Software Teile einer Standardarchitektur implementiert.

8.1 Zusammenfassung und Interpretation der Qualitätseigenschaften

In diesem Kapitel werden die validierten SPARDAT-Qualitätseigenschaften einzeln ausgewertet und das Ergebnis analysiert. Um eine Übersicht zu erhalten, wird an dieser Stelle nocheinmal ein Gesamtüberblick über die Eigenschaften gegeben, die im Rahmen dieser Diplomarbeit validiert wurden (vgl. Abbildung 40).

- Gebrauchstauglichkeit
  - Funktionstüchtigkeit
    - Funktionsabdeckung
    - Widerspruchsfreiheit
  - Betriebstüchtigkeit
    - Korrektheit
    - Zuverlässigkeit
    - Sicherheit
    - Robustheit
    - Effizienz
    - Verknüpfbarkeit
  - Benutzungsfreundlichkeit
    - Erlernbarkeit
    - Handhabbarkeit
    - Effektivität
    - Einheitlichkeit
- Pflegbarkeit
  - Wartbarkeit
    - Änderbarkeit
    - Korrigierbarkeit
    - Erweiterbarkeit
  - Prüfbarkeit
- Anpaßbarkeit
  - Portabilität
  - Wiederverwendbarkeit

Abbildung 40: Gesamtüberblick über die SPARDAT-Qualitätseigenschaft (vgl. [Wal90])

8.1.1 Funktionsabdeckung

Alle zehn Fragen aus der Prüfliste der *Funktionalen Vollständigkeit*, als ein Merkmal der `Funktionsabdeckung`, konnten im Laufe des Entwicklungsprozesses positiv bewertet werden. Es konnte damit festgestellt werden, daß alle in der Analysephase spezifizierten Anforderungen eines OERR-Systems mittels der benutzten DHE-Umgebung und der Programmiersprache VB3 umgesetzt werden konnten. DHE stellt alle nötigen Dienste zur Verfügung, um die ausgewählten charakteristischen OERR-Vorgänge implementieren zu können.

Weniger erfolgreich konnte das Merkmal *Vollständigkeit der DHE-Dokumentation*, im Rahmen der Eigenschaft `Funktionsabdeckung` bewertet werden. Von sechs Fragen wurden vier als 'Unzutreffend' bewertet. Die DHE-Dokumentation erwies sich als unvollständig und die Sachverhalte waren nicht immer konkret und wirklichkeitsgetreu dargestellt. Da diese Dokumentation als Kern für die Entwicklung einer DHE-basierten Anwendung betrachtet wird, brachten die oben erwähnten Nachteile eine Verz ögerung der Aneignung des Funktionsumfangs mit DHE mit sich.

8.1.2 Widerspruchsfreiheit

Das Merkmale *Widerspruch zwischen Funktionalitäten, die die Funktionen erfüllen* erhielt dank der eingesetzten OOSE-Methode eine positive Bewertung. Durch dessen charakteristische Eigenschaft - die Nachvollziehbarkeit zwischen den Modellen - konnten Widersprüche zwischen den einzelnen Entwicklungsphasen und Entwicklungsmodellen verhindert werden. Ohne die Nachvollziehbarkeit der OOSE-Methode wären weitere Validierungsmaßnahmen dieses Merkmals notwendig gewesen. Das Merkmal *Widerspruch zwischen dynamischen Produktverhalten und Dokumentation* erzielte, bezogen auf die DHE-Software, negative Validierungsergebnisse.

8.1.3 Korrektheit

Grundsätzlich konnte die Validierung der `Korrektheit` mittels der verwendeten Bewertungsmaße und den Prüflisten positiv ausgewertet werden. Das entwickelte OERR-System erfüllt die von ihm erwartete Funktionalität in großen Teilen. Die beim Testen aufgetretenen Fehler, die eine zeitweilige Einschränkung der Funktionalität des OERR-Systems zur Folge hatten, konnten, so die GESI-Mitarbeiter, in der DHE-Version 1.00f behoben werden. Eine neue Validierung der Eigenschaft `Korrektheit` dürfte mit dieser Version positiver ausfallen.

8.1.4 Zuverlässigkeit

Über einen Beobachtungszeitraum von 72 Stunden konnte kein fehlerfreies Funktionieren des OERR-Systems festgestellt werden. Ein Kommunikationsfehler zwischen dem Client und dem Server führte dazu, daß an unregelmäßigen Stellen in der Anwendung beim Aufruf verschiedener DHE-Dienste die Client-Server-Kommunikation unterbrochen und auch der Sybase-Server inaktiv wurde. Dieser gravierende Fehler führte zu sehr schlechten Validierungsergebnissen bezüglich der `Zuverlässigkeit` des Systems und wurde auch von den GESI-Mitarbeitern bestätigt.

8.1.5 Sicherheit

Aus der Sicht der `Sicherheit` ist einer der wichtigsten Teile von DHE der *User&Authorisation Manager* (vgl. Abbildung 7). Der darin implementierte Sicherheitsmechanismus verlangt, daß für jeden Benutzer ein Authorisierungsprofil definiert wird. Jedes Authorisierungsprofil spezifiziert, welche Funktionalität benutzt werden darf und auf welchen Daten zugegriffen werden kann. Die Bestätigung (*authentification*) ist in DHE auf die Paßwort-Prüfung beschränkt. Jeder Benutzer kann über ein individuelles Paßwort verfügen. Jede Anwendung, die auf DHE zugreift, gibt durch dessen ID und Paßwort, Informationen über ihre Handhabung.

Der *User&Authorisation Manager* von DHE entspricht dem ORB-Zugriffskontrollteil von CORBA (Common Object Request Broker Architecture, vgl. [BH97]). DHE erwartet, daß grundlegende Sicherheitsdienste, wie beispielsweise die Verschlüsselung, von anderen, darunterliegenden Schichten (vgl. Bitschicht aus Abbildung 7) durchgeführt und erfüllt werden. Die Abwesenheit solcher Dienste und, damit verbunden, der eingeschränkte Zugriffskontrollmechanismus, wird in [BH97] stark in Frage gestellt. Dort wird deshalb eine mögliche Erweiterung des DHE-Sicherheitskonzeptes mit Konzepten aus CORBA und umgekehrt vorgeschlagen.

Die Validierung der Eigenschaft `Sicherheit` durch zielgerechte Implementierung von charakteristischen OERR-Vorgänge ergab positive Ergebnisse. Es ist zwar in DHE in den Versionen 1.00a und 1.00f kein Verschlüsselungsmechanismus vorhanden, wir gehen jedoch davon aus, daß mit dem in dieser Arbeit validierten DHE-Sicherheitskonzept arbeitsprozeßbezogene Zugriffsschutzmaßnahmen, sowie Datenschutzregelungen in der Entwicklung DHE-basierter Anwendungen berücksichtigt werden können. Der Verschlüsselungsmechanismus sollte, falls notwendig, in den Anwendungen implementiert werden.

Eine weitere Sicherungsmaßnahme, die in DHE noch nicht implementiert wurde, welche jedoch für die nächste DHE-Version geplant ist, ist die Einführung der elektronischen Unterschrift. Durch entsprechende Programme (wie beispielsweise `pgp`) soll sichergestellt werden, daß eine von einer Person gesendete Nachricht tatsächlich von dieser Person gesendet wurde und nicht zwischendurch von jemand anderen geändert wurde. Dieser Aspekt soll in Zukunft bei der Benutzung der DHE-Sicherheitskontroll-und Zugriffsschutzmechanismen dem Entwickler von DHE-basierten Anwendungen noch mehr Erleichterung bringen.

8.1.6 Robustheit

Die Auswertung der Prüfliste zu dem Merkmal der `Robustheit` ergab positive Ergebnisse bezüglich dieser Eigenschaft: um möglichst wenige Benutzungseingabefehler zu ermöglichen, wurde weitgehend auf direkte Eingabefelder auf der Benutzungsoberfläche verzichtet. Stattdessen konnten mit Hilfe der von VB3 zur Verfügung gestellten Interaktionsformen (*ListBox*, *ComboBox* etc.) Vorgaben für den Benutzer angegeben werden. Für dennoch mögliche Fehler wurden entsprechende Fehlermeldungen mit ausführlichen Hinweise für den Benutzer, sowie Stornierfunktionen implementiert, so daß die Robustheit des Systems nicht beeinträchtigt sein sollte.

8.1.7 Effizienz

Die Auswertung der *Speichereffizienz*, als ein Merkmal der Eigenschaft `Effizienz`, ergab als Nachteil die verwendete 16-Bit-Entwicklungsumgebung der Client-Anwendung. Diese Einschränkungen führten zu einer Beeinträchtigung der Funktionalität des entwickelten OERR-Systems. So konnten zwei wichtige OERR-Aspekte nicht berücksichtigt werden: die Mobilität des Patienten und die Bearbeitung der Einzelleistungsergebnisse (Result-Reporting). Die für die Implementierung dieser Aspekte notwendigen DHE-Dienst- und Datenstrukturdeklarationen konnten in dem entwickelten Projekt, wegen geringer Speicherkapazität nicht geladen werden. Die von den GESI-Mitarbeiter vorgeschlagene Zwischenlösung erschien jedoch nicht zufriedenstellend, da sie eine Änderung des DHE-Quellcodes bedeutet hätte.

Es soll an dieser Stelle noch einmal betont werden, daß dieses Speicherproblem nicht durch DHE entstanden ist, sondern durch die Microsoft Visual Basic - Version 3.0. Weiterhin sind die GESI-Mitarbeiter der Meinung, daß die von ihnen zur Verfügung gestellten DHE-BASIC-Module zwar mit DHE, bzw. mit dem sogenannten GAS Werkzeug von DHE (vgl. [Fer96]), erzeugt wurden, aber nur eine Hilfe für den Entwickler darstellen sollen, diese aber nicht unbedingt benutzt werden müssen.

Die Bewertung der *Ausführungseffizienz*, als zweites Merkmal der Eigenschaft `Effizienz`, konnte durch die Analyse der von DHE zur Verfügung gestellten Datei `dheprod.log` weitgehend vereinfacht werden. Die dort aufgelisteten Ausführungszeiten konnten Hinweise über die Dauer der aufgeruften DHE-Dienste geben. Die Validierung der *Ausführungseffizienz* für zwei Clients und einen relativ kleinen DHE-Datenbestand ergab gute Ergebnisse. Die Validierung in einer realen Krankenhausumgebung, in der große Datenbestände vorhanden sind und mehr als nur ein Client zur selben Zeit Zugriffe auf diese Datenbestände ausübt, konnte in dieser Arbeit nicht durchgeführt werden.

8.1.8 Verknüpfbarkeit

Die `Verknüpfbarkeit` wurde in der vorliegenden Diplomarbeit aus zwei Sichten validiert. Die interne Sicht bezog sich auf die Verknüpfung der beiden implementierten DHE-basierten Teilanwendungen *Order Entry* (OE) und *Result Reporting* (RR). Die Betrachtung der `Verknüpfbarkeit` aus der externen Sicht brachte Überlegungen hinsichtlich einer Verknüpfung des entstandenen OERR-Systems mit einem bereits existierenden System, welches nicht auf DHE basiert: dem proCOM-Stationssystem der Firma ROKD.

Die Verknüpfung der beiden Teilanwendungen erwies sich als nicht fehlerfrei. Um eine Anforderung von der anfordernden KE an eine ausgewählte leistende KE senden zu können, muß die Identifikationsnummer der leistenden KE in DHE gespeichert werden. Dieses Feld exisitiert in DHE, der Wert wird jedoch in der DHE Version 1.00a nicht gespeichert. Die neue DHE-Version 1.00f verspricht eine korrekte und funktionsfähige Verknüpfung der beiden Teilanwendungen.

Die externe Verknüpfung des OERR-Systems mit dem proCOM-Stationssytem wurde analysiert, jedoch nicht durchgeführt. Prinzipiell erlaubt DHE die Verknüpfung mit externen Systeme auf drei Arten: Datenaustausch mittels einer ASCII-Datei, durch Replikation und mit einem Kommunkationsserver. Alle drei Möglichkeiten sind mit zusätzlichen Implementierungsarbeiten verbunden. Große Vorteile bietet jedoch die Verknüpfung durch einen Kommunkationsserver (vgl. Kapitel 9.2).

Abschließend zu diesen Betrachtungen kann die Implementierung neuer Anwendungen in DHE, gegenüber einer Integration bestehender Systeme, als weniger zeitaufwendig bewertet werden.

8.1.9 Erlernbarkeit

Die `Erlernbarkeit` wurde in dieser Arbeit bezüglich der DHE-Software und der OERR-Anwendung validiert. DHE bietet, wenn auch nicht vollständig (vgl. Kapitel 8.1.1), umfangreiche Schulungs- bzw. Lernunterlagen, sowie ein rechnergestütztes Hilfesystem, die zu einer raschen Aneignung des Umgangs mit den DHE-Benutzungsschnittstellen (Lernzeit ca. 2 Wochen) führen.

Die Aneignung des Umgangs mit den Benutzungsschnittstellen der entwickelten OERR-Anwendung erwies sich für die Mathias-Spital-Mitarbeiter als einfach und leicht. Sowohl durch DHE, aber nicht zuletzt wegen der Benutzung der Programmiersprache VB für die Konstruktion der OERR-Benutzungsschnittstelle, konnten positive Ergebnisse bezüglich der Erlernbarkeit der OERR-Anwendung erzielt werden. Die Eigenschaft der `Erlernbarkeit` kann positiv bewertet werden.

8.1.10 Handhabbarkeit

Die Benutzungsschnittstelle der entwickelten OERR-Anwendung wurde nach den standardisierten Ergonomiekriterien konstruiert. Durch den Einsatz der Programmiersprache VB und den von dieser zur Verfügung gestellten Interaktionsformen, konnten ausführliche Fehlerhinweise, eine angemessene Bedienerführung sowie eine benutzungsfreundliche Mensch-Maschine-Schnittstelle für die OERR-Anwendung realisiert werden. Die Validierung der `Handhabbarkeit` führt somit zu positiven Ergebnissen.

8.1.11 Effektivität

Die Validierung der Eigenschaft `Effektivität` erfolgte anhand einer Prüfliste bestehend aus fünf Fragen, die sowohl die Schnelligkeit der Informationsbereitstellung sowie die Aktualität und Genauigkeit der Informationen in DHE und OERR berücksichtigen, aber auch deren Anpassungsfähigkeit gegenüber Veränderungen in der Organisation, im Datenvolumen und bei Sonderfällen.

Die DHE-Version 1.00f auf einer Windows NT - Plattform bringt auch hinsichtlich der `Effektivität`, im Spezialfall der Aktualität der Informationen in OERR, gewisse Neuigkeiten: den Triggermechanismus, der die passive DHE-Komponente zu einer aktiven Komponente umwandelt (vgl. [Has97]).

Alle Fragen konnten mit 'Zutreffend' beantwortet werden, was zu einem positiven Ergebnis der Validierung der `Effektivität` führt.

8.1.12 Einheitlichkeit

Bei der Entwicklung der OERR-Anwendung wurden die ROKD-firmenspezifischen Standards sowie VB-Programmierrichtlinien berücksichtigt. Dabei wurde auch auf ein einheitliches Systemverhalten und einen einheitlichen Bildschirmmaskenaufbau geachtet. Damit soll dem Benutzer eine Hilfe beim Arbeiten mit dem System gewährleistet werden. Die `Einheitlichkeit` kann somit positiv bewertet werden, auch wenn sich die Validierung dieser Eigenschaft mehr auf die Implementierungsebene bzw. auf die eingesetzte Programmiersprache und weniger auf DHE bezieht.

8.1.13 Änderbarkeit

In der Validierung dieser Eigenschaft spielte die DHE-Datenbankabstraktion eine wichtige Rolle. So sind direkte Zugriffe auf die DHE-Datenbanken, Dateien, Tabellen und sonstige Datenstrukturen für den Entwickler einer auf DHE basierten Anwendung transparent gehalten. Die Zugriffe auf die Datenbank erfolgt nur über die von DHE zur Verfügung gestellten Dienste. Dies erwies sich als ein großer Vorteil bei der Entwicklung der OERR-Anwendung gegenüber konventionellen KIS-Anwendungen, bei deren Entwicklung insbesondere die Datenbankanbindung sehr zeitaufwendig ist. Durch die in DHE vorliegende Datenkapselung besteht die Möglichkeit einer korrekten und raschen Durchführung von Änderungen in jeder DHE-basierten Anwendung, so auch in OERR. Die Datenbanktransparenz aus DHE ist eine sehr bequeme Art und Weise, DHE-basierte Anwendungen zu entwickeln. Es kann aber manchmal auch der Fall sein, daß länderspezifische Erweiterungen der Datenbank notwendig sind. Eine Abhängigkeit von GESI ist somit nicht auszuschließen.

Deshalb wird in [Has97] ein Vorschlag gemacht, zusätzlich zu dem bestehenden Datenbanktransparenz-Mechanismus eine SQL-Schnittstelle für die Applikationsprogrammierer einzufügen. So könnten mögliche länderspezifische Änderungen in der Datenbank direkt von dem Entwickler durchgeführt werden, weiterhin bestünde jedoch die Möglichkeit, Daten zu manipulieren, ohne unbedingt die physikalische Struktur der Datenbank zu kennen.

Die Validierung der `Änderbarkeit` kann für die Bedürfnisse der entwickelten OERR-Anwendung positiv bewertet werden.

8.1.14 Korrigierbarkeit

Die Validierung dieser Eigenschaft bezog sich auf zwei Sichten: zum einen wurde validiert, ob DHE Hilfsmittel für eine möglichst einfache Fehlerfindung und -behebung dem Entwickler einer DHE-basierten Anwendung zur Verfügung stellt. Die Validierung dieses Aspektes ergibt positive Ergebnisse: auf dem DHE-Server protokolliert DHE alle Ein-/Ausgabeparameter, die bei einem DHE-Dienstaufruf zwischen den Server und dem Client fließen, so daß es nachvollziehbar ist, ob der auftretene Fehler DHE- oder anwendungsbedingt ist.

Aus einer anderen Perspektive konnte validiert werden, ob die Programmiersprache VB, mit Hilfe dessen die OERR-Anwendung implementiert wurde, Hilfsmittel zur Fehleranalyse und -beseitigung zur Verfügung stellt. Auch dieser Aspekt kann positiv bewertet werden: in VB3 war es möglich, Fehlermeldungen zu implementieren, die im Falle von DHE-bedingten Fehlern eine Fehlernummer ausgeben, die in der DHE-Dokumentation nachgeschlagen werden kann, um mögliche und genaue Fehlerursachen zur Begrenzung einer aufgetretenen Fehlersituation ausfindig zu machen. Weiterhin ermöglicht VB3, so wie viele andere Programmiersprachen auch, eine Einzelschrittkompilierung, die mögliche Syntax-, Datentyp- und Schnittstellenfehler prüft. Es konnten auch Zwischenlösungen für die temporäre Fehlerbeseitigung implementiert werden. Die `Korrigierbarkeit` kann aus beiden erwähnten Sichten positiv validiert werden.

8.1.15 Erweiterbarkeit

Einige der in der Prüfliste der Merkmale der `Erweiterbarkeit` gestellten Fragen konnten im Rahmen dieser Arbeit nicht validiert werden, da eine Erweiterung der entwickelten OERR-Anwendung nicht angestrebt wurde. Es wurden lediglich nur charakteristische OERR-Vorgänge entwickelt, ohne ein Gesamtsystem anzustreben. Es konnte jedoch festgestellt werden, ob die eingesetzten konstruktiven Qualitätssicherungsmaßnahmen Erweiterungen von auf DHE-basierenden Anwendungen ermöglichen. Insbesondere die eingesetzte Programmiersprache brachte positive Ergebnisse bei der Validierung der `Erweiterbarkeit`.

Überlegungen in dieser Richtung brachten das Ergebnis, daß wegen der in DHE bestehenden Datenkapselung und dem relativ großen Modularisierungsgrad der Anwendungen, die nicht zuletzt dank der eingesetzten Programmiersprache VB erzielt wurden, eine Erweiterung des OERR-Systems möglich ist. DHE erlaubt auch eine anwendungsspezifische Erweiterung der Dienste, durch das Einfügen von neuen Dienste (sogenannte *Extends*) seitens des Entwicklers.

8.1.16 Prüfbarkeit

Die *Zugänglichkeit* als Merkmal der `Prüfbarkeit` wurde in dieser Arbeit hinsichtlich der DHE-Dokumentation validiert. Die neun Fragen aus der entsprechenden Prüfliste befassten sich insbesondere mit der DHE-Beschreibung (Benutzerfunktionen, Bedienung, Installation, Wartung etc.), so daß eine positive Bewertung dieser Eigenschaften festgestellt werden kann.

8.1.17 Portabiliät

Die `Portabilität` wurde nur auf der Client-Seite betrachtet. Die Maschinen- und Betriebssystemumgebung des DHE-Servers wurde im laufe der Durchführung der Validierung nicht geändert.

Eine Portierung der implementierten OERR-Anwendung von einer 16-Bit Umgebung (Windows 3.1 und VB3) in eine 32-Bit Umgebung (Windows 95 und VB4) erschien Sinn zu machen, nachdem eine schlechte *Speichereffizienz* in der 16-Bit Umgebung validert wurde. Dieses Ergebnis hatte als Folge eine Beeinträchtigung der weiteren Entwicklung der Anwendung (vgl. Kapitel 8.1.7). Die Portierung erwies sich jedoch als erfolgslos: wegen der in VB4 fehlenden Unterstützung für verschiedene VB3-Objekten, die in der Anwendung benutzt wurden, sowie der Existenz neuer VB4-Konzepte, konnte eine direkte Portierung nicht durchgeführt werden. Es wäre möglich gewesen, die schon in VB3 implementierten Teile der Anwendung so umzuschreiben, daß sie auch in einer 32-bit Umgebung funktionsfähig sind, der Aufwand dafür erwies sich jedoch als zu groß.

8.1.18 Wiederverwendbarkeit

Die *Allgemeingültigkeit* als Merkmal der `Wiederverwendbarkeit` konnte durch beide betrachteten Auswertungsmethoden, sowohl das Bewertungsmaß als auch die Prüfliste, aus DHE-Sicht positiv und OERR-Sicht negativ validiert werden. Die Wiederverwendung aller von DHE mitgelieferten BASIC-Module in den beiden entwickelten DHE-basierten Teilanwendungen (OE und RR) der OERR-Anwendung erwies sich als ein großer Vorteil der benutzten DHE-Software. Die `Wiederverwendbarkeit` stellt somit eine wichtige Eigenschaft der DHE-Software dar.

8.2 Die DHE-Version 1.00f

In diesem Abschnitt soll die im Laufe dieser Arbeit des öfteren erwähnten DHE-Version 1.00f mit ihren Änderungen gegenüber der Version 1.00a zusammengefaßt werden.

Es wurde in den vorigen Kapiteln beschrieben, daß viele der aufgetretenen Fehler in der DHE-Version 1.00a, die durch die Durchführung der Validierung zum Vorschein gekommen sind, den GESI-Mitarbeiter mitgeteilt wurde. Anhand dieser Fehler und Vorschläge weiterer HANSA-Projektpartner hat GESI im September 1997 die DHE-Version 1.00f zur Verfügung gestellt, in der diese Fehler und Vorschläge berücksichtigt worden sind.

Das Update-Verfahren wurde in einem entsprechenden Dokument, ebenfalls von GESI beschrieben, zur Verfügung gestellt und verspricht eine einfache Installation der DHE-Version 1.00f. Da jedoch Fehler in der gelieferten komprimierten Datei vorhanden waren und die Validierung der Version 1.00a fast abgeschlossen war, wurde auf ein Update im Rahmen dieser Arbeit verzichtet.

Theoretisch, anhand der Dokumentation der Version 1.00f, konnte jedoch festgestellt werden, daß folgende Fehler aus der Version 1.00a entfernt wurden:

Zuverlässigkeit: Die Validierung dieser Eigenschaft deckt Kommunikationsfehler zwischen dem DHE-Client und -Server auf. Nach Angaben der Firma GESI ist dieser Fehler in der Version 1.00f behoben.

Verknüpfbarkeit: Die DHE-Version 1.00f bietet eine korrekte und funktionsfähige Verknüpfung der OE- und RR-Teilanwendungen innerhalb von DHE. Die Identifikationsnummer der leistenden KE, die bei der Generierung einer Anforderung von dem Benutzer eingegeben wird, kann nun in der DHE-Datenbank gespeichert werden (vgl. Kapitel 7.4.1).

Vollständigkeit der DHE-Dokumentation: Die vor kurzem zur Verfügung gestellten Dokumentation zu der DHE-Version 1.00f erwies sich umfassender als die Dokumentation der Version 1.00a. Die neue Dokumentation ist als HTML-Dokument mittels eines WWW-Browser verfügbar.

Ungelöst bleibt das Problem der *Speichereffizienz*: nach Angaben der Firma GESI sind die von DHE mit Hilfe des *GAS Tools* generierten BASIC-Module vom Entwickler manuell zu bearbeiten (vgl. Kapitel 8.1.7). Eine Durchführung dieses Verfahrens erwies sich als zeitintensiv.

Das beschriebene Portabilitätsproblem (vgl. Kapitel 8.1.17), das eine Portierung der teilweise entwickelten OERR-Anwendung von der 16-Bit Umgebung (VB3) in einer 32-Bit Umgebung (VB4) verhindert, wurde von der Firma GESI als ein Microsoft-bedingtes Problem dargestellt.

Eine Erweiterung, die die DHE-Version 1.00f gegenüber der Version 1.00a bietet, ist die im Kapitel 6.4.2 beschriebene Initialisierung einiger Input-Variablen. Diese DHE-bedingte Initialisierung, die vorher von dem Applikationsprogrammierer manuell durchgeführt wurde, wird in der Version 1.00f durch den DHE-Server automatisch vorgenommen.

Die meisten bei der Validierung der DHE-Version 1.00a entdeckten Probleme konnten in der Version 1.00f behoben werden. Dies wurde mittels Dokumentation dargelegt, jedoch entfiel eine praktische Prüfung. Basierend darauf, sollte eine weitere Validierung der betroffenen Qualitätseigenschaften positiv ausfallen.

Wäre zu Beginn der Validierung der Einsatz der DHE-Version 1.00f, sowie die Entwicklung der OERR-Anwendung in einer 32-bit Umgebung möglich gewesen, wären die Validierungsergebnisse positiver ausgefallen.

8.3 Bewertung der eingesetzten konstruktiven Qualitätssicherungselemente

Generell können die in dieser Arbeit eingesetzten konstruktiven Qualitätssicherungselemente als angemessen erachtet werden. Die vier Hilfsmittel für die Entwicklung des OERR-Systems mit DHE werden nun in diesem Kapitel einzeln betrachtet und kurz bewertet. Diese Bewertung soll Aussagen darüber erlauben, ob diese konstruktiven Elemente teilweise oder ganz dazu beigetragen haben, die Validierungsergebnisse der entwickelten OERR-Anwendung und somit der DHE-Software zu verbessern.

8.3.1 Das Vorgehensmodell

Das in dieser Arbeit verwendete Vorgehensmodell ist das *V-Modell* von Boehm. Dieses Modell erwies sich für die Durchführung dieser Arbeit als sehr angemessen, zumal es eine besondere Bedeutung für die Sicherstellung der Qualitätssicherung hat. Durch die Gegenüberstellung korrespondierender Phasen konnten konstruktive und analysierende Tätigkeiten parallel durchgeführt werden (vgl. Abbildung 3). Durch entsprechende Prüfmaßnahmen in jeder konstruktiven Phase konnten früh eventuell aufgetretene Fehler behoben werden.

8.3.2 Die Methode

Die in dieser Arbeit eingesetzten Methode zur Unterstützung der Entwicklung der OERR-Anwendung ist *OOSE*. Sie unterstützt das oben beschriebene Vorgehensmodell durch folgende Phasenmodelle: *Problembereichsmodell*, *Anwendungsfallmodell*, *Schnittstellenbeschreibung* und *Analysemodell* in der Analysephase, das *Entwurfs*- und *Implementierungsmodell* in der Konstruktionsphase und das *Testmodell* in der Testphase. Eine der bedeutendsten Eigenschaft, die die OOSE-Methode charakterisiert, ist die *Nachvollziehbarkeit* zwischen den Modellen. Jedes aufgestellte Modell kann aus dem davorliegenden nachvollzogen werden.

Der Einsatz der OOSE-Methode brachte auch Vorteile bezüglich der Vereinheitlichung der Dokumentation der Arbeitsergebnisse und der systematischen und zielstrebigen Vorgehensweise. Die OOSE-Methode wirkte sich somit auch auf die in dieser Arbeit durchgeführten Validierung und Validierungsergebnisse positiv aus: es wurde nicht nur eine produkt- sondern auch eine phasenbezogene Validierung durchgeführt, in dem der Entwicklungsprozeß bewertet wurde. Die positiven Ergebnisse der Validierung des Entwicklungsprozesses, die daraus entstanden sind, bewirkten auch gute Ergebnisse bei der Validierung des betrachteten Software-Produktes. So konnte beispielsweise die Bewertung des Merkmals *Funktionale Vollständigkeit* des OERR-Systems, die in allen drei Entwicklungsphasen ausgewertet wurde, durch den Einsatz der OOSE-Methode positive Ergebnisse erzielen. Dieser Aspekt führte auch zu einer einfacheren Durchführung der Validierung, da viele zu einem Merkmal gehörenden Prüffragen durch die Eigenschaft der Nachvollziehbarkeit keine weiteren ausführlicheren Begründungen benötigten.

Zusammenfassend kann die Schlußfolgerung gezogen werden, daß die in dieser Arbeit angewendeten OOSE-Methode nicht nur zu gute Validierungsergebnisse geführt hat, sondern auch die Zeit für die Durchführung der Entwicklung und Validierung verringert hat. Diese positiven Ergebnisse könnten mit anderen Methoden, bei denen die Qualitätssicherung von Software-Produkten weniger im Vordergrund steht (vgl. Kapitel 2.2.2), nicht erzielt werden.

8.3.3 Die Formalismen

Für den Aufbau der Analyse- und Entwurfsmodelle bietet die oben erwähnte OOSE-Methode als Formalismen verschiedene textuelle und graphische Notationen, die die Aufstellung und Verständlichkeit dieser Modelle erleichtert. Beispiele für solche Notationen sind die graphischen Symbole der verwendeten Analyseobjekte (Schnittstellenobjekte, Entitätsobjekte, Kontrollobjekte) oder der Entwurfsobjekte.

Für die Konstruktion von Oberflächenprototypen stellt jedoch OOSE keinen Formalismus zur Verfügung. Daher wurden *Dialognetze* eingesetzt, die die Beschreibung von Dialogabläufen in graphisch-interaktiven Systemen unterstützten. Der Einsatz von Dialognetzen in dieser Arbeit kann als angemessen bewertet werden. Die Einfachheit des Formalismus führt dazu, daß in frühen Phasen des Entwicklungsprozesses, in dem der zukünftige Benutzer noch aktiv bei der Analyse des zu entwickelten Systems beiträgt, der Ablauf der Dialoge mit Dialognetze beschrieben werden konnte: die Schnittstellenbeschreibung aus der Analysephase (vgl. Kapitel 5) enthält somit nicht nur die Beschreibung der Bildschirmmasken, sondern auch eine formalisierte, jedoch leicht verständliche Beschreibung der Abläufe.

8.3.4 Die Programmiersprache

Die verwendete Programmiersprache zur Implementierung des OERR-Systems ist *VB3*. DHE erlaubt auch die Verwendung der Programmiersprache C für die Entwicklung von Anwendungen. VB3 wurde jedoch von der Firma ROKD bevorzugt.

Ein großer Vorteil der Programmiersprache VB3 erwies sich in der Konstruktion der Benutzungsoberfläche, da dafür keinen Quellcode geschrieben werden mußte. Dies führte zu einer relativ geringen Anzahl an zu schreibenden Quellcodezeilen für die Entwicklung des OERR-Systems.

Nachteile der Programmiersprache VB3 bezüglich der OERR-Entwicklung konnten in der Speicheroptimierung und Variableninitialisierung beobachtet werden. VB3 stellt keine Konzepte zur Verfügung, um für Variablen Speicherplatz zu allokieren bzw. freizugeben. Wegen dieses Nachteils, der sogar die Validierungsergebnisse bezüglich des Merkmals *Speichereffizienz* negativ beeinflußt hat (vgl. Kapitel 8.1.7), könnte für die Implementierung von weiteren DHE-basierten Anwendungen die Programmiersprache C bevorzugt werden, welche mit den `malloc()`- und `free()`-Funktionen, sowie das Pointer-Konzept, die Speicheroptimierung unterstützt. Bezüglich der Variableninitialisierung weist VB jeder deklarierten Variable, die noch kein Wert hat, automatisch den Wert Null zu.

8.4 Abschließende Bewertung

Die ersten Ergebnisse, der in dieser Arbeit durchgeführten Validierung der DHE-Software, wurden im Rahmen der Ulmer Konferenz „New Technologies in Hospital Information Systems“ im September 1997 präsentiert (vgl. [PSL97]). Die im Kapitel 8.1 vorgestellten Gesamtergebnisse zeigen, daß es aus der Sicht der entwickelten OERR-Anwendung denkbar wäre, DHE für die Implementierung einer Standard-Architektur einzusetzen, vorausgesetzt, die noch bestehenden Fehlern würden komplett beseitigt sein.

DHE stellt Dienste zur Verfügung, mit denen alle betrachteten OERR-Vorgänge implementiert werden konnten. Die Nutzung von DHE als Middleware des CEN-Architekturgerüstes führt zu einer Kostenreduzierung in der Entwicklung neuer KIS-Anwendungen. Durch die mögliche Migration bestehender Systeme zu DHE (vgl. Kapitel 9) werden mit den Einsatz der DHE-Software auch die bereits getätigten Investitionen in bestehende KIS-Lösungen gesichert.

Bezogen auf die in dieser Arbeit entwickelten OERR-Anwendung und die betrachteten Qualitätsziele, bietet DHE nicht nur eine Lösung für die Ist-Situation der EDV in Krankenhäusern sondern, aufgrund der Konzeption als offene Client-Server-Architektur, eine plattformunabhängige und technologisch zukunftsorientierte Basis für den Aufbau eines verteilten krankenhausweiten Informationssystems.

9 Migration zu DHE

Das vorrangige Ziel des EU-HANSA-Projektes ist es, die DHE-Software im Hinblick auf ihre Einsetzbarkeit in verschiedenen Krankenhäusern zu validieren. Daran nehmen neun EU-Länder, darunter auch Deutschland, teil. Die nachfolgenden Betrachtungen beziehen sich auf einem möglichen Einsatz der DHE-Software in deutschen Krankenhäusern.

Im Kapitel 9.1 wird zuerst ein Vergleich der drei Middleware-Architekturen CORBA, DHE und HL7 vorgestellt. Im Kapitel 9.2 werden theoretische Ansätze der Migration von bestehenden Systemen zu DHE analysiert. Kapitel 9.3 schließt die Betrachtungen dieses Kapitels mit den Einsatzmöglichkeiten der DHE-Software in deutschen Krankenhäuser, ab.

9.1 Middleware-Architekturen im Vergleich

KIS-Anwendungen benötigen für ihre Kommunikation und Kooperation die Nutzung gemeinsamer Dienste. Diese Dienste sind die Inhalte und Zielstellungen von Middleware-Systemen. Damit ermöglichen standardisierte Middleware-Systeme die Integration von Anwendungssystemen unabhängig von ihrer Zugehörigkeit zu bestimmten organisatorischen und technologischen Domänen (Anwendungsgebiet, Herstellerbezug, Plattform).

„Auf der Basis von EDI (Electronic Data Interchange)-Standards wird eine offene Kommunikation unterstützt“ ([Blo97]). Dazu sind jedoch spezialisierte Kommunikationsserver oder standardisierte Middleware-Produkte erforderlich, die die eigentliche Interoperabilität realisieren. Für die interne Kommunikation in Krankenhäusern erlangte *HL7*, als ein EDI-Standard, eine besondere Bedeutung. Beispiele für Middleware, die eine Kooperativität verteilter Systeme sichert, sind *CORBA* und *DHE*. In diesem Zusammenhang sollte auch *OLE/(D)COM* (Object Linking Embedding / (Distributed) Component Object Model) von Microsoft erwähnt werden. Im folgenden werden die Middleware-Architekturen HL7, CORBA und DHE, anhand der in [BH97a] vorgestellten Vergleich, kurz bewertet. Abbildung 41 gilt dazu als Grundlage.

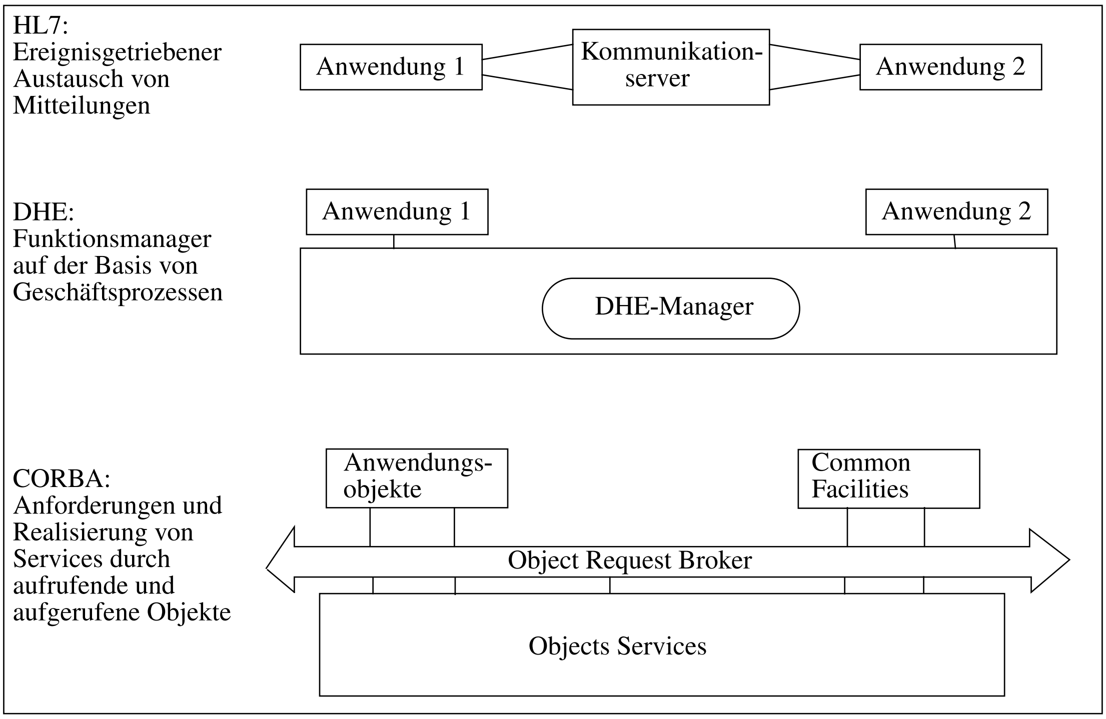

Abbildung 41: HL7, DHE und CORBA im Vergleich ([Blo97])

Die *HL7-Architektur* ermöglicht die offene Kommunikation zwischen Anwendungssystemen durch Standardisierung von Syntax und Semantik ereignisgetrieben ausgetauschter Mitteilungen. Die Kommunikation wird durch einen Kommunikationsserver bzw. durch ein standardisiertes Middleware-Produkt, die nicht Bestandteil des HL7-Kommunikationsstandards sind, verwaltet.

Die *CORBA-Architektur* der „Object Management Group“ (OMG) ist ein Middleware-Standard, der ein offenes, generisches Konzept (einschließlich der Implementierungswerkzeuge) sowohl für eine allgemeine als auch eine objektorientierte Interoperabilität verteilter Anwendungen liefert (vgl. [Blo97]). CORBA realisiert alle Funktionen unterhalb der Anwendungsschicht und fordert nur wenige Grundvoraussetzungen von einfachen Transportprotokollen (vgl. Abbildung 41).

Die *DHE-Architektur* wurde ausführlich im Rahmen dieser Arbeit beschrieben.

Ebenso wie die Beschreibung, kann auch die Bewertung und Integration der vorgestellten Middleware-Architekturen hier nur sehr grob vorgenommen werden. Jede der diskutierten Architekturen integriert Komponenten von verteilten Informationssystemen des Gesundheitswesens. Das Ausmaß der Integration ist jedoch unterschiedlich. HL7 spezifiziert standardisierte Mitteilungen auf der Anwendungsebene. DHE liefert gesundheitswesenspezifische Dienste, die die kooperierenden Anwendungen transparent unterstützen. Diese Dienste entsprechen „Vertical Common Facilities“ im CORBA-Konzept. Sie können als Geschäftobjekte in Geschäftsprozeßmodellen betrachtet werden (vgl. [Blo97]). Neben diese „Facilities“ definiert CORBA allgemeine Middleware-Dienste und Implementierungswerkzeuge, die transparent eine Kooperativität zwischen beliebigen Anwendungssystemen (auch nicht objektorientierten) ermöglichen.

Neben den unterschiedlichen Betrachtungen der drei Middleware-Architekturen wurde von Blobel et. al in [BH97a] ein komponentbasiertes Metamodell entwickelt, welches eine Evaluierung der sehr unterschiedlichen Architekturansätze sowie Ansätze zur Harmonisierung der drei betrachteten Systeme ermöglicht (vgl. Tabelle 6). Im Kapitel 8.1.5 wurde einen Ansatz für die Erweiterung des DHE-Sicherheitskonzeptes mit Konzepten aus CORBA erwähnt. Auch eine Harmonisierung der DHE-Architektur mit HL7 ist, so wie im nächsten Kapitel vorgestellt wird, nicht auszuschließen. Es kann zusammengefaßt werden, daß, obwohl die drei betrachteten Middleware-Architekturen wesentliche Unterschiede vorweisen, diese auf keinen Fall als konkurrierende Ansätze betrachtet werden können.

Merkmal	**HL7**	**CORBA**	**DHE**
Paradigma	Nachrichtenkonzept	• Objekt-Orientierung; • Verwaltung von verteilten Objekten	• Schichtarchitektur; • basiert auf gesundheitswesenspezifischem Datenmodell
Architekturkonzept	Nein	Ja	Ja
Middleware von Standard-Dienste	Nein	Ja, generisch	Ja, gesundheitswesenspezifisch; generisch, wenn nötig
Verknüpfbarkeit	Nein	Ja	Ja
Ebene der Standardisierung	Standardisierte Nachrichten	Standardisierte Architektur und Objektdienste	Europäische Prestandard Architektur (CEN/TC251)
Einfügen neuer Anwendungen	Unabhängige Entwicklung mit Benutzung standardisierter Nachrichten.	Teilentwicklung mit Benutzung von vorhandenen Objekten.	Teilentwicklung mit Benutzung der vorhandenen Dienste und darunterliegenden Modell.

Tabelle 6. Die Merkmale der Middleware-Architekturen ([BH97a])

9.2 Lösungsansätze für die Migration zu DHE

Die Mehrheit der deutschen Krankenhäuser verfügt über eine große Vielfalt von unterschiedlichen Hardware- und Software-Komponenten. Um diese getätigten Investitionen seitens der Krankenhäuser zu schützen, könnte DHE nur zusammen mit diesen Komponenten in den Krankenhäuser akzeptiert und eingesetzt werden. Deshalb werden an dieser Stelle mögliche theoretische Ansätze beschrieben, die eine Integration von schon bestehenden Systemen mit der auf DHE-basierten OERR-Anwendung erlauben. Als schon bestehendes System wird das proCOM-Stationssystem betrachtet.

Ein erster Ansatz wurde bereits im Kapitel 7.3.1 erwähnt und beschreibt die Verknüpfung des proCOM-Stationssystems mit der entwickelten OERR-Anwendung mittels einer ASCII-Datei.

Ein zweiter Ansatz basiert auf dem Konzept der Replikation. In [Tan94] werden drei Replikationsarten vorgestellt: explizite Replikation, träge Replikation und Replikation auf der Basis von Gruppen. Es wird im folgenden das Konzept der Replikation auf der Basis von Gruppen beschrieben.

Bei dieser Replikationsart werden Kopien von Daten zeitgleich an alle Server geschickt. In Abbildung 42 ist dieser Mechanismus anhand des folgenden Beispiels dargestellt: das proCOM-Stationssystem ($Client_1$) erzeugt eine Neuaufnahme. Die generierten Daten werden gleichzeitig sowohl an den proCOM-Server ($Server_1$) als auch an den DHE-Server ($Server_2$) gesendet, so daß die OERR-Anwendung ($Client_2$), ohne Zeitverzögerung, Zugriff auf diese Daten haben wird. Der im proCOM-Stationssystem neu aufgenommene Patient wird gleichzeitig auch im Belegungsplan der OERR-Anwendung geladen. So unterscheidet sich diese Replikationsart, durch den Vorteil der gleichzeitigen Replizierung, positiv von den anderen beiden in [Tan94] beschrieben Arten.

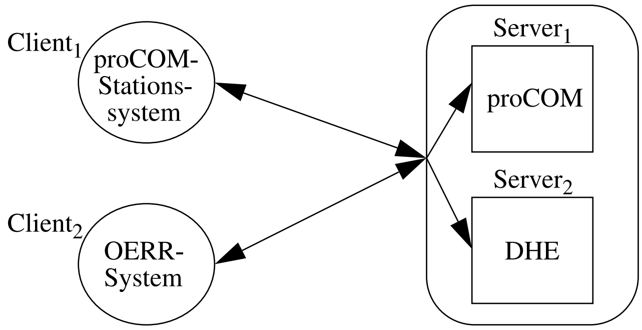

Abbildung 42: Verknüpfung durch Gruppenreplikation

Ein dritter und letzter Ansatz für die Verknüpfung von existierenden Systemen mit DHE-basierten Anwendungen ist die Nutzung eines Kommunikationsservers. Die Firma ROKD GmbH verfügt über das *pro7-System* (vgl. [SL97]), dessen zentrale Komponente ein Kommunikationsserver ist. Das pro7-System wurde in einem gemeinsamen Projekt von ICD, Dortmund, proDV GmbH, Dortmund und ROKD GmbH konzipiert und entwickelt. Der pro7-Kommunikationsserver verarbeitet die an ihn von den Subsystemen als Nachrichten gesendeten Informationen, führt eine syntaktische Analyse durch und leitet die Nachrichten in Abhängigkeit vom Sender und vom Nachrichtentyp an eine Menge von Empfängern weiter (Routing). Grundlage für eine einfache Integration von Anwendungen ist eine einheitliche Schnittstelle zum Austausch von Daten. Aus diesem Grund wird vom Kommunikationsserver ein standardisiertes Protokoll zum Austausch medizinisch-technischer und klinisch-administrativen Daten verwendet. Es handelt sich dabei um eine Erweiterung des standardisierten HL7-Protokolls (vgl. [GS97]).

Zur Zeit unterstützt das pro7-System u.a. auch die Integration des proCOM-Stationssytsems mittels eines Adapters und eines Konverters (vgl. Abbildung 43). Während der Kommunikationsserver in erster Linie den Informationsfluß im System verwaltet und steuert, wird die Offenheit durch die zusätzlichen Komponenten verwirklicht. Bei diesen Komponenten handelt es sich entweder um Adapter, die primär der physikalischen Integration dienen, oder um Komponenten zur logischen Integration, den sogenannten Konvertern (die physikalische und logische Integration wurden im Kapitel 3.1.1 definiert). Ausführliche Informationen über die Aufgaben von Adaptern und Konvertern im pro7-System können [GS97] entnommen werden.

So könnte als eine neue Kopplung im pro7-System DHE integriert werden, ebenfalls mit Hilfe eines Konverters und/oder Adapters (vgl. Abbildung 43). Auf die Adapter-Komponente könnte in diesem Fall verzichtet werden, falls DHE folgende Voraussetzungen erfüllt: Unterstützung von DCE (Distributed Common Environment)-Base Services zu DCE Version 1.1 und Unterstützung des HL7-Acknowledgements-Mechanismus. Die DCE-Base Services werden von der Plattform, auf der die DHE-Software installiert ist, unterstützt. Die aktuelle Plattform (Solaris 2.5) stellt diese Services zur Verfügung. Die DHE-Software sollte demnach um das HL7-Acknowledgemt-Mechanismus erweitert werden, damit die

Integration mit dem pro7-Kommunikationsserver ohne Adapter-Komponente hergestellt werden kann. Für weitere Details wird an dieser Stelle auf [SL97] verwiesen.

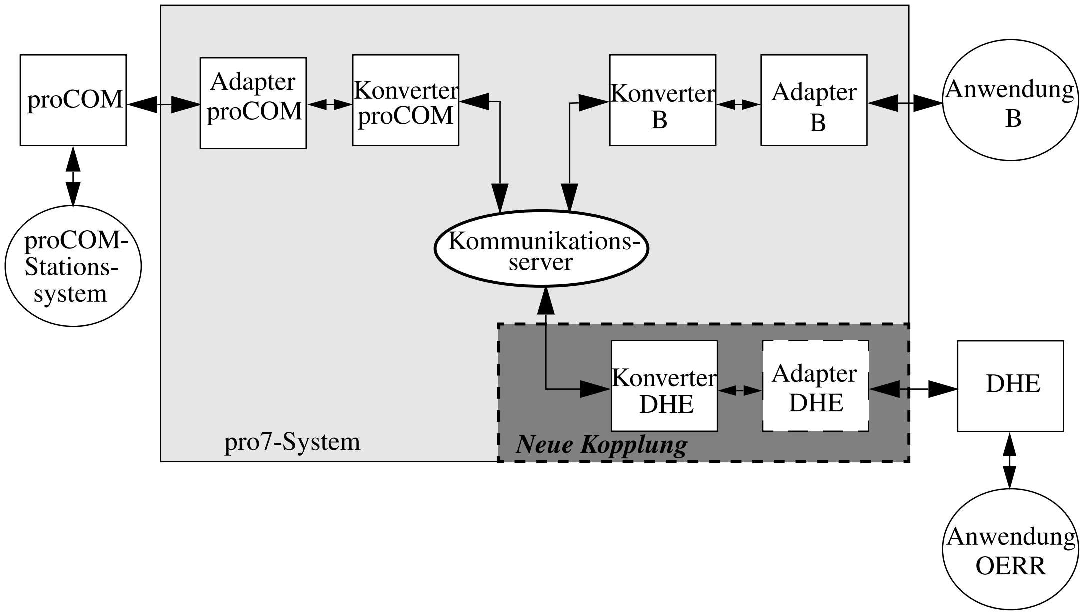

Abbildung 43: Verknüpfung über das pro7-Kommunikationsserver

9.3 Einsatzmöglichkeiten der DHE-Software

Die beschriebene Ist-Analyse in zwei deutschen Krankenhäusern in [BCF+97], sowie die ebenfalls darin enthaltene Marktanalyse gibt zu verstehen, daß in Deutschland eine Vielzahl von Krankenhäusern über EDV-Lösungen in verschiedenen Organisationsbereichen, wie beispielsweise Stationen, Funktionsbereiche, Verwaltung etc., verfügen. Insbesondere diejenigen Krankenhäuser, die nur eine geringe EDV-Infrastruktur besitzen, tendieren zu einer raschen und kostengünstigen Anschaffung von neuen Anwendungen, die sich leicht mit den vorhandenen integrieren lassen. Dazu lassen sich zwei unterschiedliche Perspektiven aufzeigen, die den Einsatz der DHE-Software in den deutschen Krankenhäusern beeinflußen:

Zum einen kann DHE, für Krankenhäuser mit wenig EDV-Lösungen, die eine Erweiterung ihrer EDV-Landschaft anstreben, zusammen mit einem Kommunikationsserver eingesetzt werden. Die Integration der bestehenden Anwendungen in DHE müßte über diesen Kommunikationsserver durchgeführt werden. Hierzu werden dann neue Anwendungen direkt auf DHE-Basis entwickelt. Die zeitsparende Entwicklung von DHE-basierten Anwendungen kann somit zu Kostenreduzierung der Erweiterungen der EDV-Lösungen führen.

Für Krankenhäuser, die zum anderen keinen großen Bedarf an neuen Anwendungen haben, sondern nur eine Integration der bestehenden EDV-Insellösungen anstreben, würde der Einsatz der DHE-Software nicht die oben genannten Vorteile erbringen. Die Stärke von DHE liegt, nach den Ergebnissen der durchgeführten Validierung, weniger in der Integration von bestehenden, nicht DHE-basierten Anwendungen, sondern viel mehr in der Entwicklung neuer Anwendungen. Aus diesem Grund ist für die dargestellte Krankenhaussituation ein Kommunikationsserver DHE vorzuziehen.

Ein Krankenhaus in Deutschland, in dem nur DHE eingesetzt wird, scheint wegen den dort bereits vorhandenen EDV-Komponenten, ein unrealistisches Ziel zu sein.

Ein weiteres Einsatzgebiet von DHE wird in den Osteuropäischen Staaten vermutet, in denen insbesonders große Krankenhäuser erstmals EDV-Lösungen einsetzen. Aus diesem Grund wurde das HANSA-EAST-Projekt unter Leitung der Firma GESI und der EU im Juli 1997 gestartet.

10 Zusammenfassung und Ausblick

Das *Distributed Healthcare Environment* ist eine Middleware, die die Integration von autonomen Anwendungen in einem Krankenhausinformationssystem unterstützt. Aus einer anderen Perspektive kann DHE als ein hochentwickeltes Management-System betrachtet werden, das alle Daten aus einem KIS speichert, überträgt und auf Integrität und Konsistenz überprüft. DHE liefert gesundheitswesenspezifische Dienste, die die kooperierenden Anwendungen transparent unterstützen. Die DHE-Architektur ist an unterschiedliche organisatorische Bedingungen anpaßbar und liefert Werkzeuge zur System-Anpassung und dynamischen Fortschreibung auch im Sinne zukünftiger Migrationsstrategien.

Mit dieser Diplomarbeit wurde ein OERR-System mit Hilfe der DHE-Software realisiert und auf verschiedene Qualitätseigenschaften validiert. Dafür wurde ein Validierungskonzept aufgestellt, indem die Validierungsziele und die entsprechenden Auswertungsmethoden identifiziert wurden, um dann das Format eines Validierungsplans für jede Entwicklungsphase festzulegen. Anschließend wurden charakteristische OERR-Vorgänge ausgewählt, mittels denen die DHE-Software anhand der festgelegten Qualitätseigenschaften validiert werden konnten. Die Durchführung der Validierung und die Interpretation der Ergebnisse schlossen diese Diplomarbeit ab.

Die in dieser Arbeit vorgestellten Ergebnisse bieten unter anderem folgende Erweiterungsmöglichkeit an:

Erweiterung des OERR-Systems: Die in dieser Diplomarbeit entwickelten charakteristischen OERR-Vorgänge könnten um weitere erweitert werden, um ein Gesamtsystem zu entwickeln (*Beispiel: Terminplaner*).

Verknüpfung mit externen Systemen: Wie in dieser Diplomarbeit vorgeschlagen, könnte das entwikkelte OERR-System um die Verknüpfung mit externen Systemen ergänzt werden. Denkbar wäre die Verknüpfung mit dem proCOM-Stationssystem mittels des pro7-Systems.

Validierung der DHE-Version 1.00f: Viele der in dieser Arbeit festgestellten Fehler in der DHE-Software Version 1.00a wurden von den GESI-Mitarbeitern in der Version 1.00f beseitigt. So könnte die aktuelle Validierung um eine Validierung der Version 1.00f von DHE hinsichtlich der Eigenschaften ergänzt werden, die negativ bewertet wurden.

Portierung in einer 32-bit Umgebung: In dieser Arbeit wurden charakteristische OERR-Vorgänge in einer 16-bit Umgebung entwickelt. Wegen den Fehlern, die bei der Validierung festgestellt wurden, ist es nötig, das OERR-System um die Teile zu ergänzen bzw. zu ändern, die das Funktionieren in einer 32-bit Umgebung erlauben.

Literaturverzeichnis

[BCF+97] I. Basusta, S.A. Cengiz, T. Frenzel, M. Gronek, H. Heinecke, J. Hilker, M. Lange, M. Patoka, R. Privighitorita, T. Proske, C. Strötmann, A. Witte: *Spezifikation eines Systems zum Behandlungsmanagement im Krankenhaus*. Endbericht der Projektgruppe 263 - PROMETHEUS. Interner Bericht, Universität Dortmund, Fachbereich Informatik, Lehrstuhl für Software-Technologie, 1996.

[BGZ92] U. Bürkle, G. Gryczan und H. Züllighoven: *Erfahrungen mit der objektorientierten Vorgehensweise bei einem Bankenprojekt*. In: Informatikspektrum Nr.15 pp. 273-281, 1992.

[BH97] B. Blobel and M. Holena: *Security Threats and Solutions in Distributed, Interoperable Health Information Systems Using Middleware*. In: New Technologies in Hospital Information Systems. Proc. Conference on Architectural Concepts for Hospital Information Systems, IOS Press, Ulm, 20.-22.9.1997.

[BH97a] B. Blobel and M. Holena: *Comparison, Evaluation and Possible Harmonisation of the HL7, DHE and CORBA Middleware*. In: New Technologies in Hospital Information Systems. Proc. Conference on Architectural Concepts for Hospital Information Systems, IOS Press, Ulm, 20.-22.9.1997.

[Blo97] B. Blobel: *Objektorientierte Middleware-Architekturen*. In: Forum der Medizin_Informatik, Heft 2/1997.

[Boe79] B. Boehm: *Guidelines for Verifying and Validating Software Requirements and Design Specification*. Euro IFIP, pp. 711-719, 1979.

[CEN93] Comitee Europeen de Normalisation: *CEN/TC251/PT010 - Conceptual Architectural Framework*. First Working Document - Draft V2.0 - pp.8, 1993.

[CR84] L.A. Clarke, D.J. Richardson: *Symbolic Evaluation - An Aid to Testing and Verification.* In: Software Validation, Ed. H.-L. Hausen, North-Holland, pp. 141-166, 1984.

[DF89] E.-E. Doberkat, D. Fox: *Software Prototyping mit SETL*. B.G. Teubner Verlag Stuttgart, 1989.

[DIN88] DIN 66234 Teil 8: *Bildschirmarbeitsplätze, Grundsätze der Dialoggestaltung*. Beuth, Berlin, 1988.

[DR93] R.G. Dromey and K. Ryan: *PASS-C: Program Analysis and Style System User Manual*. Software Quality Institute, Griffith University, 1993.

[Dum92] R. Dumke: *Softwareentwicklung nach Maß*. Vieweg Verlag, Braunschweig/Wiesbaden, 1992.

[EDI93] Consorzio EDITH: *The Hospital Framework*. In: UC-11/02-001, Version 1.0. Rome, 1993.

[EM87] M.W. Evans, J.J. Marciniak: *Software Quality Assurance and Management*. Wiley & Sons, 1987.

[Fer95] F.M. Ferrara: *HANSA Project Highlights-Version 1.0*. Internal Report, Firma GESI srl, Rome, 1995.

[Fer95a] F.M. Ferrara: *Specification of the DHE middleware-Version 1.0*. Internal Report, Firma GESI srl, Rome, 1995.

[Fer96] F.M. Ferrara: *The DHE middleware information view-Version 1.1*. Internal Report, Firma GESI srl, Rome, 1996.

[Fer96a] F.M. Ferrara: *The DHE middleware functional view-Version 1.1*. Internal Report, Firma GESI srl, Rome, 1996.

[Fer96b] F.M. Ferrara: *The DHE middleware API services-Version 1.1*. Internal Report, Firma GESI srl, Rome, 1996.

[FFH+97] W. Franke, T. Frenzel, J. Hilker, M. Patoka, R. Privighitorita, T. Proske, H.-G. Sobottka, C. Stroetmann: *PROMETHEUS - Spezifikation eines rechnergestützten Behandlungsmanagementsystems im Krankenhaus*. Software-Technik Memo Nr. 94 des Lehrstuhls für Software-Technologie, Fachbereich Informatik, Universität Dortmund, September 1997.

[GS97] B. Gesell, T. Schmal: *Das pro7-System als Basis für den Aufbau eines verteilten krankenhausweiten Informationssystem*. In: W. Hasselbring (Ed.): Erfolgsfaktor Softwaretechnik für die Entwicklung von Krankenhausinformationssystemen. Krehl Verlag, Münster, 1997.

[Has97] W. Hasselbring: *Softwaretechnik für die Entwicklung von Krankenhausinformationssystemen*. Folien zum lehrstuhlinternen Seminarvortrag, 1997.

[ISO90] ISO 9241 Part 10: *Ergonomic Dialog Design Criteria, Version 3*. Committee Draft, December 1990.

[Jan93] C. Janssen: *Dialognetze zur Beschreibung von Dialogabläufen in graphisch-interaktiven Systemen*. In: Proc. Software-Ergonomie, Teubner-Verlag, 1993.

[JCJ+93] I. Jacobson, M. Christerson, P. Jonsson and G. Övergaard: *Object-Oriented Software Engineering*. Addison-Wesley, 1993.

[Kit96] B. Kitchenham: *Software Metrics - Measurement for Software Process Improvment*. NCC Blackwell Publishers, Cambridge, USA, 1996.

[Lor96] W. Lordieck: *EU-Projekt HANSA (HealthCare Advanced Networked System Architecture) - Stand und weitere Vorgehensweise*. Folien, August 1996.

[Lor96a] W. Lordieck: *Objectives, Expectations and Results Measurement foreseen at ROKD Bielefeld and Mathias-Spital Rheine*. In: HANSA-ROKD-02-001, Vers.1.0, March 1996.

[MR93] G. Mutsch, M. Reitinger: *Professionelles Applikationsdesign mit Microsoft Visual Basic*. Trainings-Seminar-Unterlagen. Microsoft Institut GmbH, München, 1993.

[MRW77] J.A. McCall, P.K. Richards, G.F. Walters: *Factors in Software Quality,* Vol. I-III. Rome Air Development Center, 1977.

[MS94] Microsoft Institut GmbH: Winword 2.0. München, 1994.

[Mye79] G.J. Myers: *The Art of Software Testing*. Wiley & Sons, 1979.

[OD94] B. Oliver and R.G. Dromey: *SAFE: a programming language for software quality.* First International (Asia-Pacific) Conference on Software Quality and Productivity, Hong Kong, 1994.

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

[Pri97] R. Privighitorita: *HANSA-Tätigkeitsreport*. Ausarbeitung zu dem Vortrag im Rahmen des 4. German HANSA-Meetings. Interner Bericht, Firma ROKD GmbH, Bielefeld, 1997.

[PSL97] R. Privighitorita, H.-G. Sobottka and W. Lordieck: *Experiences with DHE: An Order Entry and Result Reporting Case Study.* In: New Technologies in Hospital Information Systems. Proc. Conference on Architectural Concepts for Hospital Information Systems, IOS Press, Ulm, 20.-22.9.1997.

[Püt95] O. Pütter: *Entwurf und Implementation einer Bibliothek graphischer Dialoge für die Ein- und Ausgabe von* PROSET. Diplomarbeit am Fachbereich Informatik der Universität Dortmund, 1995.

[ROKD95] ROKD: *Open Hospital System*. Kunden-Broschüre, Firma ROKD GmbH, Bielefeld, Februar 1995.

[ROKD96] ROKD: *proCOM Krankenhaus*. Produktinformation, Firma ROKD GmbH, Bielefeld, Juni 1996.

[ROKD97] ROKD: *Visual Basic Entwicklungsrichtlinien - Version 1.00*. Interner Bericht, Firma ROKD GmbH, Bielefeld, 1997.

[Roy70] W. Royce: *Managing the development of Large Software Systems - Concepts and Techniques*. In: Proc. WESCON, 1970.

[SL97] M. Schulte and W. Lordieck: *Experiences with DCE: The pro7 Communication Server Based on OSF-DCE Functionality.* In: New Technologies in Hospital Information Systems. Proc. Conference on Architectural Concepts for Hospital Information Systems, IOS Press, Ulm, 20.-22.9.1997.

[Sta94] C. Stary: *Interaktive Systeme - Software-Entwicklung und Software-Ergonomie*. Vieweg Verlag, 1994.

[Tan94] A. S. Tanenbaum: *Moderne Betriebssysteme*. Carl Hanser und Prentice-Hall International Verlag. München, Wien 1994.

[Wal90] E. Wallmüller: *Software-Qualitätssicherung in der Praxis*. Carl Hanser Verlag, München Wien, 1990.

[You89] Edward Yourdon: *Modern Structure Analysis*. YOURDON Press Computing Series, 1989.