Diplomarbeit

Eine Taxonomie und ein Anforderungskatalog für Kommunikationsserver im Krankenhaus

Matthias Lange

Diplomarbeit
am Fachbereich Informatik
der Universität Dortmund

17. April 1997

Gutachter:

Dr. W. Hasselbring
Prof. Dr. E.-E. Doberkat

Betreuer:

Dr. N. Osada
Prof. Dr. H.U. Prokosch
Institut für Medizinische Informatik und Biomathematik,
Westfälische Wilhelms-Universität Münster

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

CPU	Central Programming Unit
DBMS	Datenbankmanagementsystem
DV	Datenverarbeitung
EDV	elektronische Datenverarbeitung
HDD	High Density Disk (Festplatte)
HL7	Health Level 7
ISO	International Organization of Standardization
KKS	Krankenhauskommunikationssystem
KIS	Krankenhausinformationssystem
KS	Kommunikationsserver
OMT	Object-Oriented Modeling Technique [RBP+91]
OSI	Open Systems Interconnection
PDV-System	Patientendatenverwaltungssystem
SQL	*Structured Query Language*, relationale Datenbankabfragesprache

Kapitel 1

Einleitung

1.1 Motivation und Zielsetzung

Die in den letzten Jahrzehnten gewachsene DV-Landschaft in den deutschen Krankenhäusern ist dadurch geprägt, daß nur in einigen wenigen Bereichen Systeme zur elektronischen Datenverarbeitung (EDV-Systeme) eingesetzt werden. Vor allem in den traditionellen Bereichen der Verwaltung (Buchhaltung, Abrechnung, etc.) und in einigen wenigen Funktionsbereichen, wie in der Radiologie und den klinisch-chemischen Laboratorien, wurden in sich abgeschlossene Software-Systeme installiert. Ein Datenaustausch hat zwischen diesen Systemen bisher kaum und auf direktem elektronischem Wege noch seltener stattgefunden.

Mit der weiteren Verbreitung der EDV-Systeme änderte sich dieses Bild. Vor allem in den großen Universitätskliniken wurden in immer mehr Funktionsbereichen EDV-Systeme eingesetzt. Der Wunsch nach einem elektronischen Datenaustausch zwischen diesen Systemen wuchs. Bisher wurden für die Kopplung der Systeme Punkt-zu-Punkt-Verbindungen eingesetzt. Auf diese Weise entstand ein Verbund der einzelnen EDV-Systeme des Krankenhauses mit vielen individuellen Schnittstellen (siehe Abbildung 1.1).

Die aktuelle Situation ist durch eine sich immer weiter ausbreitende EDV geprägt. In immer neuen Bereichen im Krankenhaus werden EDV-Systeme installiert. Neben den ursprünglich administrativen und funktionsorientierten Bereichen sind nun in zunehmenden Maße auch die stationären und ambulanten Abteilungen betroffen. Eine Kopplung dieser Systeme über einen Ausbau des Rechnerverbundes durch weitere Punkt-zu-Punkt-Verbindungen ist nicht mehr praktikabel. Jedes neue System bringt eine Vielzahl neuer Schnittstellen mit sich.

Kommunikationsserver sollen hier eine Abhilfe schaffen. Die einzelnen EDV-Systeme in den Abteilungen werden nun nicht mehr direkt miteinander verbunden, sondern alle Kommunikationsverbindungen werden über den Kommunikationsserver geführt. Somit wandelt sich das Bild der Vernetzung der DV-Systeme entscheidend. Statt des traditionellen, fast vollständigen Verbundes kommt nun eine sternförmige Vernetzung der Systeme zum Einsatz (siehe Abbildung 1.2).

Jedes DV-System im Krankenhaus wird nun direkt mit dem Kommunikationsserver verbunden. Somit wird für jedes System für den Anschluß an den Kommunikationsverbund nur noch eine physikalische Verbindung benötigt. Zusätzlich stellt der Kommunikationsserver eine einheitliche Implementierungsumgebung für die Schnittstellen zwischen den Systemen zur Verfügung. Somit lassen sich alle Schnittstellen an einem Ort bündeln und auf eine einheitliche Weise verwalten. Sie sind so wesentlich einfacher zu warten, als wenn sie im Netzwerk verstreut installiert wären.

Die EDV-Landschaft in den Krankenhäusern ist durch die unterschiedlichsten EDV-Systeme , -Plattformen und Datenformate geprägt. Hinzu kommt, daß viele der DV-Systeme nicht oder nur unzureichend auf einen elektronischen Datenaustausch mit anderen Systemen eingerichtet sind. Mit dem

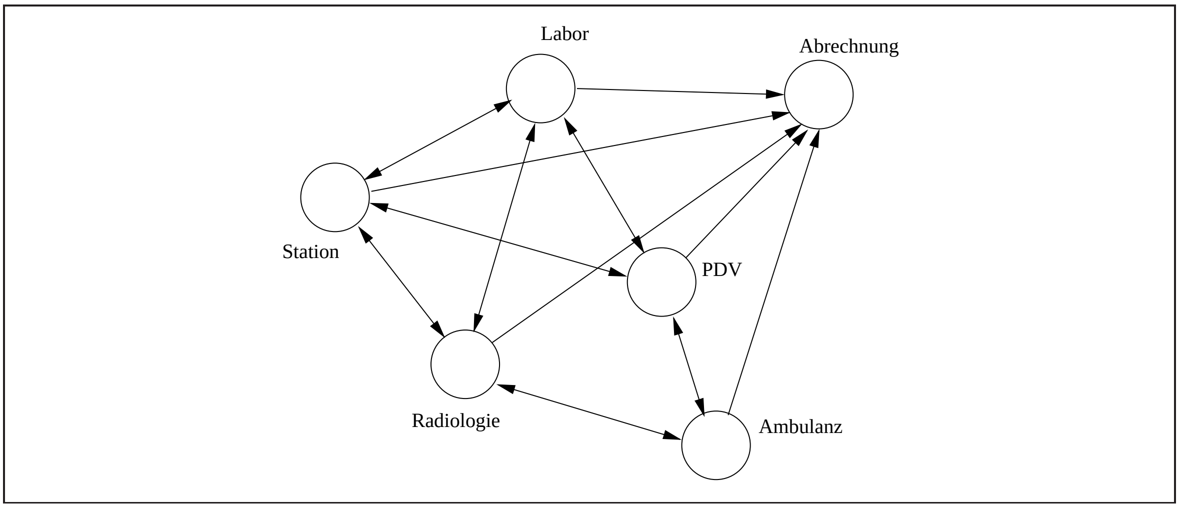


Abbildung 1.1: Vernetzung der EDV-Systeme eines Krankenhauses über Punkt-zu-Punkt-Verbindungen.

Einsatz eines Kommunikationsservers lassen sich hier viele Hürden bei der Einrichtung und der Unterhaltung leistungsfähiger Kommunikationsverbindungen überwinden.

Für die Realisierung eines Kommunikationsservers sind verschiedene Software-Architekturen denkbar. Neben einer zentralen Lösung, die die oben angesprochene sternförmige Vernetzung der DV-Systeme berücksichtigt, ist auch ein verteilter Ansatz denkbar. Die zur Zeit auf dem Markt befindlichen Produkte benutzen unterschiedliche Architekturen und unterstützen verschiedene Bandbreiten an Funktionen.

Viele Krankenhäuser stehen zur Zeit vor der Entscheidung, welche Bereiche der elektronischen Kommunikation zwischen den EDV-Systemen sie über einen Kommunikationsserver abwickeln wollen. Hierbei spielt eine Rolle, welche Architektur – zentral oder verteilt – sich im jeweiligen DV-Umfeld am besten einpassen läßt und die Ansprüche an die Performanz und die Sicherheit am ehesten erfüllt. Es geht also darum zu entscheiden, welches der auf dem Markt verfügbaren Produkte die eigenen Anforderungen am besten abdeckt.

Ziel dieser Arbeit ist es, einen Anforderungskatalog zu erarbeiten, der den Krankenhäusern als Grundlage für die Zusammenstellung eines eigenen spezifischen Anforderungskataloges dienen kann. Der Anforderungskatalog wird neben allgemeinen Anforderungen auch solche enthalten, die sich aus dem speziellen Einsatzgebiet „Krankenhaus“ ergeben. Auch theoretische Anforderungen und Möglichkeiten werden mitberücksichtigt, die zur Zeit zwar noch nicht erfüllt werden können, aber doch die ganze Bandbreite der Anforderungen aufzeigen sollen.

Neben dem Anforderungskatalog soll auch eine Taxonomie für Kommunikationsserver den Krankenhäusern eine Hilfestellung für die eigene Entscheidung bieten. Die Taxonomie dient der Differenzierung der verschiedenen Software-Architekturansätze und der Darstellung der Konsequenzen, die sich aus den verschiedenen Architekturen zum Beispiel für die Performanz ergeben.

Anforderungskatalog und Taxonomie bieten schließlich die Grundlage für einen Vergleich einiger auf dem Markt zur Zeit verfügbaren Produkte. Ausgehend von einer Analyse der Architektur und des Funktionsumfanges der betrachteten Kommunikationsserver, erfolgt eine detaillierte Gegenüberstellung der Produkte anhand des Anforderungskataloges sowie eine Einordnung in die Taxonomie. Auch dieser Vergleich kann den Entscheidungsträgern in den Krankenhäusern eine Grundlage für ihre Entscheidungen bieten.

Viele Probleme, die sich aus dem Aufbau von Kommunikationsverbindungen über einen Kommuni-

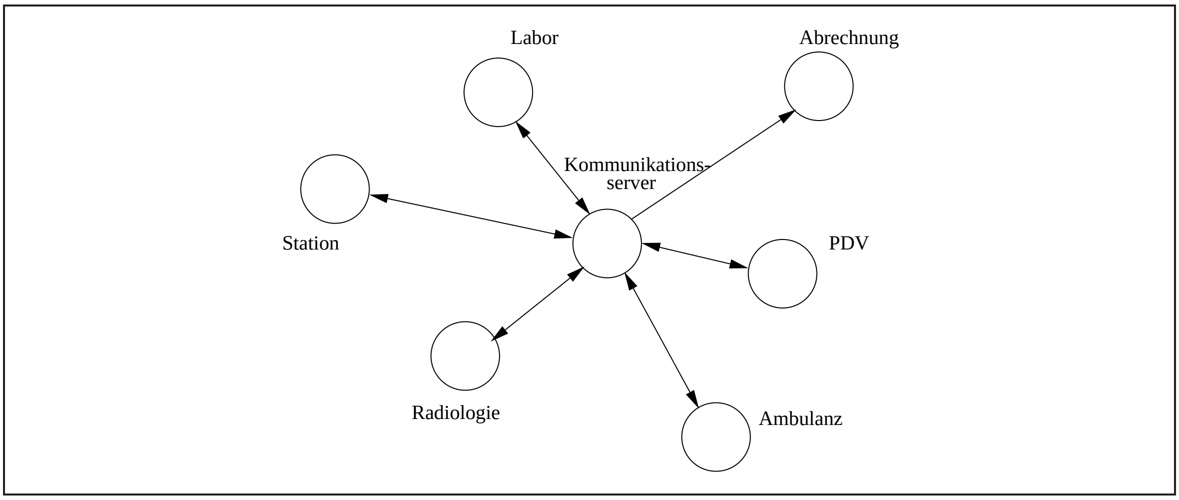


Abbildung 1.2: Vernetzung der EDV-Systeme eines Krankenhauses über einen Kommunikationsserver.

kationsserver ergeben, lassen sich nicht vollständig auf einer rein theoretischen Basis ermitteln. Zur Vertiefung der Problematiken und zur exakteren Analyse der Anforderungen an einen Kommunikationsserver wurden daher im Rahmen dieser Arbeit auch praktische Erfahrungen mit dem Umgang mit Kommunikationsservern durch die Implementierung einiger Kommunikationsverbindungen gewonnen. Diese Erfahrungen sind mit in den Anforderungskatalog und die Analyse weiterer Kommunikationsserver eingeflossen.

1.2 Überblick

Nach einigen allgemeinen Begriffsdefinitionen in Kapitel 2 werden in Kapitel 3 das Aufgabengebiet und der prinzipielle Aufbau von Kommunikationsservern vorgestellt. Die Analyse der Aufgaben und der Funktionsweise der Kommunikationsserver führt in Kapitel 4 zu der Ausarbeitung eines Katalogs an Anforderungen, denen sich ein Kommunikationsserver bei einem Einsatz im Krankenhaus stellen muß. Die Gliederung des Katalogs orientiert sich dabei stark an dem in Kapitel 3 entwickelten allgemeinen Modells eines Kommunikationsservers.

Dieses Modell ist auch Grundlage für die im Kapitel 5 vorgestellte Taxonomie für Kommunikationsserver. Taxonomie und Anforderungskatalog bieten die Basis für die im Kapitel 6 dargestellte Analyse der zur Zeit auf dem Markt angebotenen Kommunikationsserver. Nach einer kurzen Beschreibung der Funktionsweise folgt eine Einordnung der Kommunikationsserver in die Taxonomie. Ein umfassender Vergleich der Kommunikationsserver, der sich an dem Aufbau des Anforderungskatalogs orientiert, befindet sich im Anhang C.

Die Entwicklung des Anforderungskatalogs und die Analyse der Funktionsweise der Kommunikationsserver wurde duch eine praktische Implementierung von Kommunikatonsverbindungen über einen Kommunikationsserver unterstützt. Die Ergebnisse dieser Implementierung werden in Kapitel 7 vorgestellt.

In Kapitel 8 folgt eine Analyse und Diskussion der Ergebnisse dieser Arbeit. Neben einer Analyse des Anforderungskatalogs, des Vergleichs und der implementierten Kommunikationsverbindungen wird ein Ausblick auf weitere Einsatzmöglichkeiten eines Kommunikationsservers zur Unterstützung der elektronischen Kommunikation im Krankenhaus gegeben.

In einigen Abbildungen wird die Notation von Rumbaugh et al. zur objektorientierten Modellierung

(OMT) verwendet. Eine kurze Erläuterung dieser Notation befindet sich im Anhang A.1. Für eine ausführliche Darstellung von OMT sei auf [RBP+91] verwiesen.

Die erweiterte Notation der dargestellten Datenflußdiagramme befindet sich im Anhang A.2.

Die in dieser Arbeit erwähnten kommerziellen Software-Produkte sind durchgehend mit einem hochgestellten *trade-mark*-Zeichen (z.B.: DataGate^{TM}) versehen. Der Herstellernachweis zu diesen Produkten befindet sich im Anhang B.

Im Anhang D werden die Kommunikations- und Transportprotokolle beschrieben, die für die Implementierung der Kommunikationsverbindungen über den Kommunikationsserver (siehe Abschnitt 7.3) verwendet werden. Die Beschreibung der für diese Kommunikationsverbindungen implementierten Kommunikationsklienten folgt im Anhang E.

Kapitel 2

Begriffsdefinitionen

Viele Begriffe in der medizinsichen Informatik sind nicht eindeutig definiert und werden teilweise mit sehr unterschiedlichen Bedeutungen verbunden. Um für diese Arbeit eine einheitliche Nomenklatur zu schaffen, sollen in diesem Abschnitt die grundlegenden, verwendeten Begriffe definiert werden.

2.1 Krankenhauskommunikationssystem

In [Pro94] unterscheidet Prokosch verschiedene Formen des DV-Einsatzes im Krankenhaus, darunter „funktionsorientierte Subsysteme“ und „Krankenhauskommunikationssystem“.

Dabei sind unter „funktionsorientierten Subsystemen“ die DV-Systeme in einem Krankenhaus zu verstehen, die primär auf die Unterstützung der Arbeitsabläufe innerhalb einer Krankenhausabteilung ausgelegt sind. Eine Kommunikation mit anderen DV-Systemen wird von ihnen, wenn überhaupt, nur am Rande berücksichtigt [Pro94, Definition 2.8, Seite 14].

Ein „Krankenhauskommunikationssystem“ (KKS) verbindet diese funktionsorientierten Subsysteme des Krankenhauses sowohl auf der physikalischen, als auch auf der logischen Ebene. Die physikalische Integration bezieht sich hierbei auf den Zusammenschluß aller administrativen und klinischen Subsysteme zu einem klinikweiten Netzwerk.

Definition 2.1 (Krankenhauskommunikationssystem (KKS))

Ein *Krankenhauskommunikationssystem* entsteht durch die physikalische und logische Integration klinischer und administrativer Subsysteme zu einem klinikweiten Netzwerk. Die anfallenden Kommunikationsprozesse werden dabei durch ein DV-Modul zur Auftragskommunikation unterstützt. [Pro94, Definition 2.9, Seite 15]

Die logische Integration setzt eine umfassende Analyse aller Kommunikationsprozesse, die zwischen den einzelnen Krankenhausabteilungen bestehen, sowie der in den Kommunikationsprozessen ausgetauschten Daten voraus. Unter Auftragskommunikation im DV-technischen Sinn werden im Krankenhaus die Funktionen zur Auftragserfassung, Übertragung von Aufträgen an die Leistungsstelle, die Darstellung des aktuellen Bearbeitungszustandes sowie die Übermittlung und Präsentation der Leistungsergebnisse verstanden.

Der Begriff „Krankenhausinformationssystem“ (KIS) soll in diesem Zusammenhang bewußt vermieden werden, da nach Prokosch ein KKS noch nicht automatisch ein KIS darstellt. Erst wenn die Funktionalität des KKS um informations- und wissensverarbeitende Funktionen erweitert wird, kann von einem KIS gesprochen werden (vgl. [Pro94]).

In Anlehnung an [Dud89] definiert Prokosch „Kommunikation“ und „Kommunikationsprozeß“ wie folgt:

Definition 2.2 (Kommunikation und Kommunikationsprozeß)

Kommunikation wird als Austausch von Meldungen (Kommunikationsinhalten) zwischen mindestens zwei als Sender und Empfänger bezeichneten Partnern verstanden. Alle während einer Kommunikation ablaufenden Vorgänge von der Formulierung der Nachricht im Sender, der Transformierung der Kommunikationsinhalte in eine den Übertragungsmedien entsprechende Darstellungsform bis zu ihrer Darstellung im Empfänger werden als *Kommunikationsprozeß* bezeichnet.

[Pro94, Definition 2.3, Seite 9]

2.2 Subsysteme

Des weiteren sollen die funktionsorientierten Subsysteme kurz als „Subsysteme" des Krankenhauskommunikationssystems bezeichnet werden. Hierbei sollen in bezug auf einen bestimmten Kommunikationsprozeß „kommunizierfähige" und „kommunizierunfähige" Subsysteme unterschieden werden. Entscheidend ist hierbei, daß die Kommunizierfähigkeit eines Subsystems immer in Abhängigkeit von einem bestimmten Kommunikationsprozeß gesehen wird. Ein für den Kommunikationsprozeß A kommunizierunfähiges Subsystem kann durchaus für den Kommunikationsprozeß B kommunizierfähig sein. Für die Aufnahme eines Subsystems in das KKS muß das Subsystem für alle Kommunikationsprozesse, in die es durch das KKS eingebunden wird, kommunizierfähig gemacht werden (siehe Abschnitt 3.3).

Definition 2.3 (Kommunizierfähiges Subsystem)

Ein Subsystem wird in bezug auf einen bestimmten Kommunikationsprozeß als *kommunizierfähig* bezeichnet, wenn es in der Lage ist, die für den Kommunikationsprozeß notwendigen Nachrichten zu erzeugen bzw. entgegenzunehmen und darzustellen.

Nimmt man diese Definition zum Maßstab, so müssen fast alle Subsysteme in einem KKS als ursprünglich kommunizierunfähig gelten. Es gibt im Krankenhaus kaum DV-Systeme, die auf eine elektronische Kommunikation mit anderen DV-Systemen in Form von Nachrichten ausgelegt sind. Die Systeme, die von Haus aus eine Import- bzw. Exportschnittstelle besitzen, über die die für einen Kommunikationsprozeß notwendigen Daten bereitgestellt bzw. aufgenommen werden können, stellen bereits die Ausnahme dar. Die exportierten Daten lassen sich in der Regel im komplexen Umfeld eines KKS noch nicht als Nachrichten verwenden. Sie müssen um weitere Daten, wie zum Beispiel den gewünschten Empfänger im KKS erweitert werden. In den meisten Fällen müssen also die Subsysteme des KKS um ein Modul erweitert werden, das die Kommunizierfähigkeit des Subsystems im Sinne der Definition 2.3 herstellt. Im Abschnitt 3.3 wird auf diese Problematik detaillierter eingegangen.

Neben der Erzeugung bzw. der Darstellung der Nachrichten im DV-System gehören zum Kommunikationsprozeß auch die Abbildung der Nachrichten in eine dem Übertragungsmedium angepaßte Darstellungsform sowie die Übertragung zu den Empfängern. An dieser Stelle setzt im KKS die Unterstützung der Kommunikation durch das in der Definition 2.1 erwähnte DV-Modul zur Auftragskommunikation ein. Ein Kommunikationsserver kann teilweise die Rolle dieses DV-Moduls übernehmen, indem er Basistechnologien für die Übertragung der anfallenden Nachrichten bereitstellt. Die Auftragserfassung, Darstellung der aktuellen Bearbeitungszustände sowie die Darstellung der Leistungsergebnisse, die auch zur Auftragskommunikation gehören, können von ihm allerdings nicht erbracht werden. Diese Aufgaben müssen bei Einsatz eines Kommunikationsservers durch Module der Subsysteme bzw. durch eigenständige Module erbracht werden.

2.3 Kommunikationsserver

Die Ursprünge der Entwicklung der Kommunikationsserver liegen in den USA. Dort sind Bezeichnungen wie *Integration Engine*, *Interface Engine* oder *Connection Server* gebräuchlich. Im deutschen Sprachgebrauch hat sich die Bezeichnung „*Kommunikationsserver*“ als Übersetzung der englischen Begriffe durchgesetzt.

Unklar ist bisher geblieben, was ein Kommunikationsserver zu leisten hat und welche Aufgaben er in der Kommunikation zwischen DV-Systemen zu erfüllen hat. Dies gilt insbesondere für das relativ neue Einsatzgebiet „Krankenhaus“. Unumstritten sind die Basisdienste, wie das Empfangen und Weiterleiten der Nachrichten. Auch die Konvertierung der Kommunikationsprotokolle (Schicht 7 im ISO/OSI-Modell) durch den Kommunikationsserver gehört im allgemeinen zu seinen Standardaufgaben. Doch wie steht es um über diese Basisdienste herausgehende Aufgaben? So zum Beispiel die Synchronisation des Nachrichtenaustausches oder die kontrollierte Steuerung von Kommunikationsprozessen im Sinne der Auftragskommunikation. Im Krankenhaus werden überwiegend kommunizierunfähige DV-Systeme eingesetzt. Wo sind hier die Grenzen des Aufgabengebietes der Kommunikationsserver zu ziehen?

Um für diese Arbeit eine einheitliche Grundlage zu schaffen, soll hier der Begriff „*Kommunikationsserver*“ und das Aufgabengebiet genauer definiert werden.

Definition 2.4 (Kommunikationsserver)

Ein *Kommunikationsserver* ist ein Software-System, das den Aufbau, den Betrieb und die Kontrolle von Kommunikationsverbindungen zwischen autonomen, heterogenen, in bezug auf Kommunikationsprozesse kommunizierfähigen DV-Systemen ermöglicht.

Nach dieser Definition gehört die Synchronisation der Nachrichtenübertragung zum Aufgabengebiet eines Kommunikationsservers. Die Zuständigkeitsgrenze zwischen Subsystem und Kommunikationsserver verläuft demnach zwischen der Bereitstellung und der Übertragung der Nachrichten.

Da Kommunikationsserver der Hauptgegenstand der Betrachtungen in dieser Arbeit darstellen, ist ihnen ein eigenes Kapitel (Kapitel 3) gewidmet, in dem die Definition 2.4 und die Aufgaben eines Kommunikationsserver detaillierter ausgeführt werden.

2.4 Nachrichten

Eine Nachricht dient dem Austausch von Informationen zwischen zwei Subsystemen. Diese Informationen bestehen im Krankenhaus in der Regel aus Daten zu einem Patienten, zum Beispiel Stammdaten oder Untersuchungsergebnisse. Nachrichten ohne patientenbezug, zum Beispiel Materialanforderungen, treten ebenfalls auf. Nachrichten können isoliert auftreten oder miteinander verbunden sein. Die Materialanforderung wäre ein Beispiel für eine isolierte Nachricht. Anfragen (*Queries*) an ein datenhaltendes Subsystem und die zugehörigen Antworten (*Replies*) stellen miteinander verbundene Nachrichten dar.

Für jede Nachricht, die ein Subsystem senden oder empfangen kann, muß die **Nachrichtenstruktur** definiert werden. Die Nachrichtenstruktur legt die genaue Anordnung der Datenfelder innerhalb der Nachricht fest. Logisch zusammengehörende Datenfelder können zu Segmenten zusammengefaßt werden, die wiederum Bestandteil anderer Segmente sein können. In den Datenfeldern stehen die eigentlichen Daten der Nachricht. Ein Datenfeld kann zusätzlich über einen Datentyp (z.B.: *Character* oder *Integer*) und eine Annotation verfügen, die die Bedeutung der Daten beschreibt. So könnte ein Datenfeld, das den Namen eines Patienten aufnehmen soll, mit dem Datentyp *String* und der Annotation „Name des Patienten“ versehen werden.

Datentyp und Annotation werden in der Regel nicht mit der Nachricht übergeben, sondern sind in der Dokumentation der Nachrichtenstruktur festgehalten. Sie sind aber für die spätere Transformation der Nachricht in andere Nachrichtenstrukturen wichtig.

In Anlehnung an die Begriffswelt der Datenbanktheorie werden synonym zu Nachrichtenstruktur oft auch die Begriffe *Datensatzbeschreibung*, *Datensatzformat* oder *Nachrichtenformat* verwendet.

Jeder Nachricht muß ein **Nachrichtentyp** zugeordnet werden. Dieser Nachrichtentyp stellt das Bindeglied zwischen der rohen Nachricht und ihrer Strukturdefiniton dar. Beispiele für Nachrichtentypen im Krankenhaus sind „Neuaufnahme eines Patienten“ oder „Übermittlung von Untersuchungswerten aus dem Labor“. Anders als Datentyp und Annotation der Datenfelder muß dieser Nachrichtentyp explizit in der Nachricht enthalten sein. In der Regel sind hierfür ein oder bei hierarchischen Nachrichtentypen mehrere Datenfelder der Nachricht vorgesehen.

In Abbildung 2.1 wird der Aufbau einer Nachricht aus den Klassen „Nachricht“, „Segment“ und „Datenfeld“ dargestellt. Der Nachrichtentyp ist ein Attribut der Klasse „Nachricht“. Die Klasse „Segment“ stellt dabei eine rekursive Aggregation dar. So gliedert sich zum Beispiel eine HL7-Nachricht in Segmente, Felder, Komponenten und Subkomponenten. Ein Modell der Nachrichtenstrukturen in HL7 wird in [Krö96] vorgestellt.

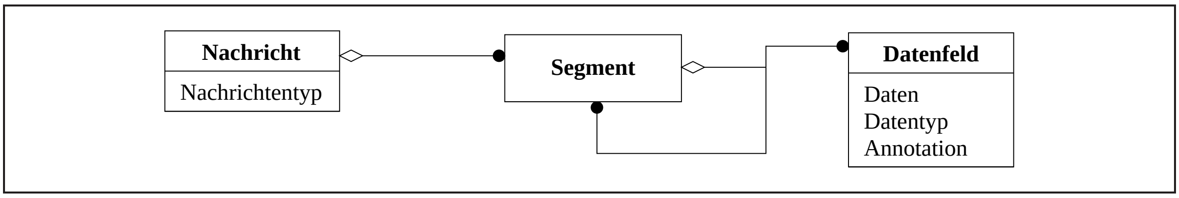


Abbildung 2.1: Aufbau einer Nachricht aus Segmenten und Datenfeldern in OMT-Notation.

Letztendlich besteht jede Nachricht aus einer Zeichenkette (*String*), in der die einzelnen Daten strukturiert angeordnet sind. Grundsätzlich gibt es zwei Möglichkeiten, die Nachrichtenstruktur auf einen String abzubilden:

1. **Fixe Feldlänge**
   Für jedes Datenfeld wird Offset und Länge innerhalb des Strings festgelegt. Da die Datenfelder nun eine fixe Länge besitzen, muß zusätzlich ein Füllzeichen (meistens das Leerzeichen) vereinbart werden. Ist ein Datum kürzer als die vorgesehene Feldlänge, so wird das Datenfeld mit diesem Zeichen links- oder rechtsbündig aufgefüllt.

2. **Variable Feldlänge**
   Es werden Sonderzeichen vereinbart, die jeweils das Ende eines Datenfeldes anzeigen. Durch unterschiedliche Zeichen kann eine Nachricht so in die Segmente und Datenfelder zerlegt werden. Auch weitere Unterteilungen sind möglich (Segmente, Felder, Subfelder etc.).

Abbildung 2.2 zeigt jeweils eine Beispielnachricht mit fixer und variabler Feldlänge.

2.5 Transportprotokolle

Die Transportprotokolle sind für das Versenden der Nachrichten zwischen zwei kommunizierenden DV-Systemen zuständig. Sie regeln die physikalische Übertragung der Nachrichten über das Netz. Im ISO/OSI-Referenzmodell [Tan96] sind sie in der vierten Schicht, dem *Transport Layer*, angesiedelt. Hier kommen in der Regel Standards wie TCP/IP oder LU6.2 [Tan96, Sch88] zum Einsatz. Zum *Transport Control Protocol* (TCP) ist anzumerken, daß es älter als der OSI-Standard ist und nicht vollständig mit den im Standard beschriebenen Funktionen von *TP Level 4* übereinstimmt (vgl. [Sch88]).

Grundsätzlich sollen in dieser Arbeit *dateibasierte* und *dateilose* Transportprotokolle unterschieden werden.

```
Position  0123456789012345678901234567890123456789012345...
String    PATDATQUERY 600MACDOC  MACDOC  83452738        ...
```

(a) Nachricht mit fixer Feldlänge

```
MSH|^~\&|SAPISH||SCHMEDHNO||19950701121110||ADT^A01|MDae19950...
EVN|A01|19950701121100||
PID|||20074|20081|Testmann^Bernd|Testmann|19670604|M|||^Muench...
```

(b) Nachricht mit variabler Feldlänge (HL7 Nachricht)

Abbildung 2.2: Beispielnachrichten mit fixer und variabler Feldlänge.

Bei einer **dateibasierten** Übertragung werden die Nachrichten nicht direkt zwischen den Applikationen ausgetauscht, sondern die sendende Applikation erzeugt eine Datei, in die die Nachrichten geschrieben werden. Diese Datei wird dann mit Hilfe eines dateibasierten Transportprotokolls (z.B. FTP) in den Verzeichnisraum der empfangenden Applikation übertragen. Diese kann nun die Nachrichten aus der Datei einlesen. Zur Unterstützung solcher dateibasierter Transportprotokolle muß der Kommunikationsserver in der Lage sein, Nachrichten aus Dateien zu lesen und Dateien mit Nachrichten zu erzeugen.

Bei der direkten oder **dateilosen** Übertragung (z.B. über TCP-Sockets [Tan96]) fällt die Datei als Transportmedium einer Nachricht weg. Die Nachrichten werden direkt zwischen den Applikationen ausgetauscht.

Die Transportprotokolle decken in der Regel lediglich die unteren vier Ebenen des ISO/OSI-Modells ab. Sie müssen daher so erweitert werden, daß auch die fünfte und sechste Ebene des Modells berücksichtigt werden. Dies geschieht durch die Vereinbarung, welche Darstellungsform für das Versenden der Nachrichten verwendet werden soll. So kann zum Beispiel eine Längenkodierung (die ersten Bytes einer Nachricht geben die Länge der Nachricht an) oder ein *Envelope* (Anfang und Ende der Nachrichten werden durch Steuerzeichen angezeigt) für das Versenden der Nachrichten über eine Socket [Tan96] verwendet werden. Auch die Vereinbarungen für Sicherungsmechanismen, wie die Bildung von Checksummen oder der Austausch von positiven oder negativen Quittungen (*Acknowledgements*), sind in diesen Ebenen anzusiedeln.

Definition 2.5 (Transportprotokoll)

Die Transportprotokolle regeln den physikalischen Austausch der Nachrichten zwischen zwei Applikationen (ISO/OSI-Modell Ebenen 1 bis 4). Sie werden durch Protokolle für die Darstellungsform und die Übertragungssicherung der Nachrichten erweitert (Ebenen 5 und 6 des ISO/OSI-Modells).

2.6 Kommunikationsprotokolle

Die Kommunikationsprotokolle legen die Darstellungsformen der Nachrichten auf der Applikationsebene fest (7. Ebene des ISO/OSI-Modells [Tan96, Sch88]). In ihnen werden die Nachrichtenstrukturen definiert. Dabei kann ein Kommunikationsprotokoll aus mehreren verschiedenen Nachrichten bestehen.

Definition 2.6 (Kommunikationsprotokoll)
Ein Kommunikationsprotokoll besteht aus den Definitionen einer oder mehrerer Nachrichtenstrukturen, die in einem abgegrenzten Umfeld zum Einsatz kommen.

Im Krankenhaus werden zur Zeit noch selten standardisierte Kommunikationsprotokolle wie HL7 [Ham93] oder Edifact [EDI95] eingesetzt. In der Regel werden für die Nachrichten Strukturen verwendet, die entweder von den Anwendern selbst definiert oder von den Herstellern der Subsysteme vorgegeben werden. Diese Kommunikationsprotokolle sollen in dieser Arbeit als **proprietäre Kommunikationsprotokolle** bezeichnet werden.

2.7 Synchronisation der Nachrichtenübertragung zwischen DV-Systemen

Eine Synchronisation der Nachrichtenübertragung findet auf vielen Ebenen im ISO/OSI-Modell statt ([Sch88, Tan96]). Auf der untersten Ebene muß eine Synchronisation der Abtastfrequenzen bei der Signalübertragung über das Übertragungsmedium erfolgen. Die höheren Ebenen synchronisieren den Austausch der Nachrichtenpakete oder der kompletten Nachrichten über zum Beispiel den Austausch von Quittungen zwischen Sender und Empfänger. Der Sender wartet nach dem Senden einer Nachricht auf eine positive Quittung des Empfängers. Erst dann versendet er eine weitere Nachricht. Eine negative Quittung zeigt einen Fehler in der Nachrichtenübertragung an und veranlaßt den Sender, die Nachricht erneut zu versenden. Mit diesem Verfahren wird eine große Sicherheit bei der Übertragung erreicht. Als Konsequenz können allerdings lange Wartezeiten beim Sender entstehen, wenn zum Zeitpunkt der Nachrichtenübertragung der Empfänger nicht empfangsbereit ist. Werden keine Quittungen zwischen den Systemen ausgetauscht, so spricht man von einer *asynchronen Übertragung*.

Findet die Synchronisation in den Transportprotokollen statt, so soll von einer **Synchronisation auf Transportprotokollebene** gesprochen werden. Die Struktur der ausgetauschten Quittungen wird hier im Transportprotokoll festgelegt. Ein Beispiel wäre das im Anhang D.2 beschriebene *HL7 Lower Level Protocol*. Auf dieser Ebene wird lediglich die Synchronisation des Nachrichtenaustausches zwischen den Übertragungseinrichtungen von DV-Systemen erreicht. Eine Synchronisation der DV-Systeme selber läßt sich durch die Definition von Acknowledgement-Nachrichten bewerkstelligen. Hier wartet das DV-System nach dem Versenden einer Nachricht auf den Empfang einer Acknowledgement-Nachricht. Beide Nachrichten stellen für die Übertragungseinrichtungen der DV-Systeme eigenständige Nachrichten dar, die wiederum auf Transportprotokollebene synchron oder asynchron übertragen werden können. Eine Synchronisation der Systeme über Nachrichten soll als **Synchronisation auf Nachrichtenebene** bezeichnet werden. Auch hier können lange Wartezeiten entstehen, wobei nun sogar das gesamte DV-System und nicht nur die Übertragungseinrichtung betroffen ist.

Im Abschnitt 3.4 wird noch genauer auf die Synchronisierung der Nachrichtenübertragung über einen Kommunikationsserver eingegangen.

2.8 Komponenten, Module und Programme von Software-Systemen

Software-Systeme wie ein Kommunikationsserver stellen sehr komplexe Systeme dar, die aus vielen Teilen zusammengesetzt sind. Alle Teile dienen einer übergeordneten Funktion. Bei einem Kommunikatonsserver ist dies zum Beispiel die Unterstützung der Kommunikation zwischen verschiedenen DV-Systemen. Diese übergeordnete Funktion läßt sich in Teilfunktionen zerlegen. Bei einem Kommunikatonsserver unter anderem in die Teilfunktionen „Konfigurierung“ und „Kommunikationsunterstützung“. Die Teilfunktionen lassen sich weiter unterteilen, so daß sich eine hierarchische Struktur eines Software-Systems bilden läßt.

Bezeichnung	Erklärung	Beispiel
Komponente	Zusammenfassung von logisch zusammengehörenden Funktionen des Software-Systems zu einer abstrakten Einheit.	**Konfigurationskomponente:** Alle Funktionen des Kommunikationsservers, die sich mit der Konfigurierung des Systems befassen.
Teilkomponente	Weitere Unterteilung der Komponenten in einzelne Funktionseinheiten	Teilkomponente für die Konfigurierung der Kommunikationsverbindungen
Modul	Zusammenfassung von Funktionen zu einer mehr oder weniger abgeschlossenen Programmeinheit.	Konfigurationsmodul für das Editieren der Konfigurationsdateien.
Programm/Prozeß	Prozeß auf Betriebssystemebene, der eine bestimmte Funktion eines Programm-Moduls ausführt.	Editoren für die einzelnen Konfigurationsdateien.

Tabelle 2.1: Tabelle der in dieser Arbeit verwendeten Hierarchie zur Beschreibung von Software-Systemen.

Die einzelnen Stufen der Hierarchie bieten verschiedene Sichten auf ein Software-System, die auf unterschiedlichen Abstraktionsebenen liegen. Die oberste Stufe bietet eine Sicht, die nur übergeordnete Funktionen eines Software-Systems bietet, die weitestgehend unabhängig von der Implementierung sind. Die nächsten Stufen verdeutlichen die Architektur des Software-Systems immer mehr, bis auf der letzten Stufe die unteilbaren Programmeinheiten (Prozeduren und Funktionen) betrachtet werden.

Für diese Arbeit soll eine vierstufige Hierarchie eingeführt werden, die Sichten auf unterschiedlichen Abstraktionsebenen ermöglicht:

1. Die oberste Stufe soll durch die Zusammenfassung von logisch zusammengehörenden Funktionen eines Software-Systems gebildet werden. Bei einem Kommunikationsserver wären dies zum Beispiel die schon oben erwähnten Funktionen „Konfigurierung" und „Kommunikationsunterstützung". Die Bestandteile des Software-Systems sollen auf dieser Stufe der Hierarchie in Anlehnung an [Goe93] als **Systemkomponenten** oder kurz als **Komponenten** bezeichnet werden. Zu einer Komponente gehören nicht nur die ausführbaren Programmteile eines Software-Systems. Bei der Konfigurationskomponente eines Kommunikationsservers würden auch die Konfigurationsdateien mit eingeschlossen sein.

2. Die zweite Stufe läßt eine Unterteilung der Komponenten in **Teilkomponenten** zu. Diese Ebene befindet sich auf der gleichen Abstraktionsebene wie die erste Stufe der Hierarchie. So könnte in der Konfigurationskomponente eines Kommunikationsservers eine Teilkomponente gebildet werden, die für die Konfigurierung der Anschlüsse der Subsysteme zuständig ist.

3. Während die ersten beiden Stufen von der eigentlichen Programmarchitektur weitestgehend abstrahieren, orientiert sich die dritte Stufe an der Architektur eines Software-Systems. Die Einheiten auf dieser Ebene sollen als **Programm-Module** oder kurz als **Module** bezeichnet werden. Sie stellen eine Zusammenfassung von Funktionen zu einer mehr oder weniger abgeschlossenen Programmeinheit dar. So könnte die Konfigurationskomponente des Kommunikationsservers ein Konfigurationsmodul enthalten, mit dessen Hilfe die Konfigurationsdateien editiert werden können.

4. Die letzte Ebene schließlich orientiert sich sehr stark an der eigentlichen Implementierung der Module. So werden in ihr die einzelnen Programmteile der Module betrachtet. Diese stellen

eigenständige ausführbare Programme dar, denen ein Betriebssystemprozeß zugeordnet werden kann. Sie werden als **Programme** oder **Prozesse** bezeichnet oder tragen eigenständige Namen, die ihre Funktion beschreiben. Das Konfigurationsmodul des Kommunikationsservers kann zum Beispiel aus verschiedenen Editoren für die einzelnen Dateien bestehen.

Die vier Stufen der Hierarchie werden in der Tabelle 2.1 nochmals zusammengefaßt.

Kapitel 3

Kommunikationsserver

In diesem Kapitel soll auf die Aufgaben eines Kommunikationsservers aus Definition 2.4 detaillierter eingegangen werden. Aus den Aufgaben lassen sich Funktionen ableiten, die von einem Kommunikationsserver für die Ausführung der Aufgaben erbracht werden müssen. Sie werden im Abschnitt 3.1 dargestellt. Abschnitt 3.2 unterteilt einen Kommunikationsserver in verschiedene Komponenten. Dabei ist unter einer Komponente eine Zusammenfassung von logisch zusammengehörenden Funktionen zu verstehen. Die weiteren Abschnitte des Kapitels vertiefen einige spezielle Aufgaben des Kommunikationsservers. So den Anschluß der Subsysteme an den Kommunikationsserver (Abschnitt 3.3), da hier die Grenzen des Zuständigkeitsbereichs des Kommunikationsservers bei der Generierung und Weiterleitung der Nachrichten festgelegt werden muß. Abschnitt 3.4 beschäftigt sich mit den Möglichkeiten der Synchronisation des Nachrichtenaustausches über einen Kommunikationsserver hinweg. Die Diskussionen zwischen den Herstellern der Kommunikationsserver und den Anwendern in den Krankenhäusern auf den aktuellen Tagungen und Kongressen (z.B. GMDS-Tagung September 1996 in Bonn [GMD], DataGate^{TM}-Anwendertreffen Januar 1997 in Essen) zeigen, daß dieses Problem zur Zeit noch nicht zufriedenstellend gelöst ist. Abschnitt 3.5 geht schließlich auf das Problem eines intern vom Kommunikationsserver verwendeten Kommunikationsprotokolls ein. Einige Kommunikationsserver bilden alle Nachrichten intern auf einheitliche Nachrichtenstrukturen ab. In der Regel wird hierfür ein vorhandener Standard wie HL7 verwendet. Vor- und Nachteile sollen in diesem Abschnitt diskutiert werden.

3.1 Aufgaben eines Kommunikationsservers

Nach Definition 2.4 auf Seite 7 ist es Aufgabe des Kommunikationsservers, Nachrichten von einem Subsystem entgegenzunehmen und Nachrichten an ein bzw. mehrere Subsysteme weiterzuleiten. Das Importieren oder Exportieren der Daten aus den Nachrichten in bzw. aus dem Subsystem gehört nach dieser Definition nicht zu seinen Aufgaben. Diese Funktionen müssen vom Subsystem selbst erbracht werden. Der Kommunikationsserver kann also von bereits vorhandenen Nachrichten ausgehen und muß sich nicht um deren Generierung kümmern. Abschnitt 3.3 zeigt allerdings, daß die Realität oft anders aussieht.

Die Aufgaben eines Kommunikationsservers lassen sich nun folgendermaßen umreißen:

1. Nachrichten von einem Subsystem entgegennehmen.

   Die Nachrichten werden von dem Subsystem erzeugt und entweder dateibasiert oder dateilos (vgl. Abschnitt 2.5) durch ein entsprechendes Transportprotokoll vom Kommunikationsserver eingelesen. Nachrichten, die vom Kommunikationsserver empfangen werden, werden auch als *Inbound*-Nachrichten bezeichnet.

2. Empfänger der Nachrichten ermitteln.

   Im nächsten Schritt müssen die Empfänger der Nachricht ermittelt werden. Die Empfänger sind entweder in der Nachricht codiert oder werden implizit durch den Typ der Nachricht festgelegt (z.B.: Nachrichten vom Typ „Stammdaten eines neuaufgenommenen Patienten“ werden immer an das Laborsystem gesandt). Dazu ist es notwendig, daß die Typen der Nachrichten vom Kommunikationsserver ermittelt werden können.

3. Übermitteln der Nachrichten an die Empfänger.

   Nachdem die Subsysteme ermittelt wurden, an die die Nachrichten gesandt werden sollen, können die Nachrichten übermittelt werden. Dabei müssen unter Umständen die Nachrichtenstrukturen in die Kommunikationsprotokolle der Empfänger transformiert werden. Nachrichten, die den Kommunikationsserver verlassen, werden auch als *Outbound*-Nachrichten bezeichnet.

Um diesen Aufgaben gerecht zu werden, muß ein Kommunikationsserver einige grundlegende Funktionen bereitstellen. Diese sollen im folgenden kurz dargestellt werden. Eine ausführlichere Beschreibung der Funktionen befindet sich im Kapitel 4, wo auch Anforderungen an die einzelnen Funktionen aufgestellt werden. Die Reihenfolge stellt keine Bewertung der Funktionen in bezug auf ihre Relevanz für einen Kommunikationsserver dar, sondern spiegelt in etwa die Schritte bei der Konfigurierung bzw. den Verarbeitungsprozeß einer Nachricht im Kommunikationsserver wider. Die Funktionen F1 bis F8 sowie F10 wurden auch von Heitmann identifiziert und sind in [Hei96] beschrieben. Die Funktionen F4 bis F11 stellen zusätzliche Funktionen dar.

F1: Definition der logischen Kommunikationsbeziehungen
Bevor Kommunikationsverbindungen über einen Kommunikationsserver eingerichtet werden können, müssen die logischen Kommunikationsbeziehungen zwischen den Subsystemen des KKS definiert worden sein. Alle Kommunikationsprozesse mit den beteiligten Subsystemen und den ausgetauschten Nachrichten müssen identifiziert und definiert werden. Die Aufgabe des Kommunikationsservers ist es nun, geeignete Funktionen zur Abbildung der logischen Kommunikationsbeziehungen im Kommunikationsserver bereitzustellen. Dies geschieht in der Regel über die Definition von Routing-Tabellen, in denen festgelegt wird, welche Nachrichten zwischen welchen Subsystemen ausgetauscht werden.

F2: Definition der Transportprotokolle
Um die Nachrichten zwischen den Subsystemen und dem Kommunikationsserver austauschen zu können, müssen Kommunikationsverbindungen zwischen den Subsystemenen und dem Kommunikationsserver eingerichtet werden. Eine *Kommunikationsverbindung* besteht dabei aus der physikalischen Leitung über das Netz und einem Transportprotokoll (siehe Abschnitt 2.5), das den Nachrichtenaustausch über die Verbindung regelt. Während die physikalische Verbindung von dem KKS zugrundeliegenden Netzwerk erbracht wird, ist die Definition und Ausführung der Transportprotokolle Aufgabe des Kommunikationsservers.

Für jedes Subsystem können mehrere Transportprotokolle und auch mehrere physikalische Verbindungen eingerichtet werden. So können zum Beispiel Laborbefunde dateibasiert über FTP alle 3 Stunden an den Kommunikationsserver gesandt werden, während zeitkritische Notfalluntersuchungen über TCP-Sockets sofort nach der Erstellung übertragen werden. Mehrere physikalische Verbindungen bieten sich bei sehr wichtigen Kommunikationsverbindungen als alternative Kommunikationswege bei Ausfall einer Verbindung an. Bei dem Beispiel der Laborbefunde können die zeitkritischen Notfallbefunde bei Ausfall der Socket-Verbindung auch per Fax vom Kommunikationsserver auf die entsprechende Station gesandt werden.

F3: Definition der Kommunikationsprotokolle
Neben der Definition und dem Aufbau der Kommunikationsverbindungen müssen die Strukturen der ausgetauschten Nachrichten definiert werden. Dies geschieht über die Definition eines Kommunikationsprotokolls für jede Kommunikationsverbindung. Jeder Verbindung wird dabei genau

ein Kommunikationsprotokoll zugeordnet. Theoretisch können über eine Kommunikationsverbindung auch unterschiedliche Kommunikationsprotokolle verwendet werden. Das Transportprotokoll ist von den Nachrichtenstrukturen unabhängig, so daß von diesen keine Beschränkungen ausgehen. Der Verarbeitungsprozeß der Nachrichten im Kommunikationsserver ist aber sehr stark von den Nachrichtenstrukturen abhängig. Würden über eine Verbindung Nachrichten aus unterschiedlichen Kommunikationsprotokollen versandt (z.B. HL7-Nachrichten und Nachrichten aus einem proprietären Protokoll), müßte der Kommunikationsserver nach dem Empfang der Nachricht jeweils zuerst das Kommunikationsprotokoll ermitteln, um dann die richtigen Verarbeitungsmethoden (siehe Funktionen F5 bis F7) anwenden zu können. Um diesen Aufwand zu vermeiden, wird für jede Kommunikationsverbindung nur ein Kommunikationsprotokoll vorgesehen.

Die Kommunikationsprotokolle definieren nicht nur die Strukturen der Nachrichten, sondern durch sie wird auch die Semantik der Daten in den Nachrichten festgelegt. Dies ist für die spätere Definition der Transformationen der Nachrichten in andere Kommunikationsprotokolle erforderlich (siehe Funktion F7). Die Semantik der Datenfelder wird allerdings nicht in der Nachricht selbst kodiert, sondern sie wird lediglich in der Dokumentation der Kommunikationsprotokolle festgehalten.

Während sich diese ersten drei Funktionen mit der Konfigurierung der Kommunikationsverbindungen befassen, ergeben sich die folgenden Funktionen aus dem Betrieb und der Kontrolle der Kommunikationsprozesse durch den Kommunikationsserver:

F4: Übertragung der Nachrichten zwischen den Subsystemen und dem Kommunikationsserver
Aus der Definition 2.4 geht hervor, daß der Kommunikationsserver von bereits vorhandenen Nachrichten ausgehen kann. Seine Aufgabe beschränkt sich hier darauf, die Übertragung der Nachrichten über die Ausführung der Transportprotokolle und den darin definierten Maßnahmen zur Übertragungssicherung und zur Synchronisation zu vollziehen.

F5: Nachrichtenidentifikation
Der erste Schritt der Nachrichtenverarbeitung im Kommunikationsserver besteht aus der Identifikation des Nachrichtentyps. In standardisierten Kommunikationsprotokollen sind die Nachrichtentypen bereits festgelegt (in HL7 zum Beispiel über das `MSH`-Segment). Aber auch in proprietären Kommunikationsprotokollen müssen Nachrichtentypen und Identifikationsregeln für diese Typen vorgesehen sein. Der Kommunikationsserver muß also die Definition und die Anwendung von Identifikationsregeln für die einzelnen Nachrichten aus den Kommunikationsprotokollen ermöglichen. Eine Identifikationsregel für eine HL7-Nachricht könnte folgendermaßen lauten:

> „Wenn im MSH-Segment im Feld 9 'ADT' und im EVN-Segment im Feld 1 'A01' steht, dann handelt es sich um eine Nachricht vom Typ 'Neuaufnahme'."

Der Nachrichtentyp wird für die weitere Nachrichtenverarbeitung benötigt.

F6: Ermittlung der Empfänger der Nachrichten (*Routing*)
Bei der Definition der logischen Kommunikationsbeziehungen wird festgelegt, welche Nachrichten zwischen welchen Subsystemen ausgetauscht werden. Somit ergeben sich meistens die Empfänger einer Nachricht implizit aus dem identifizierten Nachrichtentyp. Die Aufgabe des Kommunikationsservers beschränkt sich somit hier auf eine Auswertung der Definition der logischen Kommunikationsbeziehungen, die im allgemeinen in Form von Routing-Tabellen erfolgt ist. Der Nachrichtentyp wurde in der Nachrichtenidentifikation bestimmt. Der Sender steht normalerweise durch die Eindeutigkeit einer Kommunikationsverbindung fest, da jedes Subsystem über eine eigene Verbindung angeschlossen ist. Werden über eine Kommunikationsverbindung verschiedene Subsysteme angeschlossen, so muß der Sender der Nachricht entsprechend dem Nachrichtentyp aus dem Inhalt der Nachricht ermittelt werden. Der oder die Empfänger werden dann mit Hilfe des Nachrichtentyps und des Senders über die Routing-Tabelle ermittelt.

Für ein flexibles Multi- und Broadcasting ist aber auch eine dynamische Ermittlung der Empfänger aus dem Inhalt einer Nachricht, zum Beispiel aus speziellen Datenfeldern, denkbar.

F7: Nachrichtentransformation
Durch die Einführung eines Kommunikationsservers werden nur noch Kommunikationsprotokolle zwischen den Subsystemen und dem Kommunikationsserver vereinbart und nicht mehr zwischen zwei Subsystemen. Da die einzelnen Subsysteme aber möglicherweise unterschiedliche Protokolle verwenden, wird eine Transformierung der Nachrichten in die verschiedenen Kommunikationsprotokolle innerhalb des Kommunikationsserver notwendig. Neben einer Transformierung der Nachrichtenstrukturen kann auch eine Werttransformation einzelner Datenfelder notwendig werden. So zum Beispiel die Transformierung der Darstellung eines Datums aus dem Format „`JJMMTT`“ in das Format „`TT.MM.JJJJ`“.

Die Aufgabe des Kommunikationsservers ist es, die Definition von Transformationsregeln und deren Anwendung auf die Nachrichten zu unterstützen. Die Transformationsregeln sind dabei stets von dem Nachrichtentyp und der Nachrichtenstruktur der gesendeten und der zu sendenden Nachricht abhängig. Eine solche Transformationsregel könnte, in einen natürlichsprachlichen Text umgewandelt, folgendermaßen lauten:

> „Wenn der Kommunikationsserver eine Nachricht vom Typ 'Neuaufnahme' vom PDV-System an das Laborsystem erhält, dann wird das Feld 4 in das Feld 5 der Empfängernachricht kopiert, das Datum in Feld 7 gelesen und in der Form `TTMMJJ` in Feld 2 geschrieben, ...“

F8: Fehlermanagement
Fehler können in der Struktur oder im Inhalt einer Nachricht auftreten. So können bei HL7-Nachrichten obligate Segmente fehlen oder der Aufbau eines Segments fehlerhaft sein. Fehler im Inhalt der Datenfelder machen sich meistens bei der Werttransformation bemerkbar. So zum Beispiel, wenn in einem Datenfeld mit dem Datentyp *Integer* Buchstaben eingetragen sind.

Beim Fehlermanagement geht es um eine effektive Behandlung dieser Fehler. Neben dem Erkennen der Fehler müssen auch Regeln definiert werden können, die festlegen, was mit den falschen Nachrichten geschehen soll. So können sie in einer Datenbank abgelegt oder mit einem entsprechenden Hinweis an den Sender zurückgesandt werden.

F9: Persistenzsicherung
Nachrichten dürfen innerhalb des Kommunikationsservers nicht verloren gehen. Durch Zwischenspeichern von Nachrichten in persistenten Queues oder ähnlichen Speicherstrukturen ist die Persistenz der Nachrichten auch nach einem Systemabsturz vom Kommunikationsserver zu gewährleisten.

F10: Überwachen der Kommunikationsverbindungen (*Monitoring*)
Eine wichtige Aufgabe des Kommunikationsservers ist neben der Verarbeitung der Nachrichten die Überwachung der Kommunikationsverbindungen. Darunter ist sowohl die Protokollierung der Vorgänge in Log-Dateien (*Logging*) als auch eine Kontrolle der Verbindungen zu verstehen. So sollte der Kommunikationsserver über Alarmierungsfunktionen verfügen, die zum Beispiel den Ausfall einer Verbindung zu einem Subsystem erkennen und einen entsprechenden Alarm auslösen können.

F11: Testen der Konfigurationen
Diese Aufgabe steht eigentlich am Anfang des Betriebs einer neuen Kommunikationsverbindung über den Kommunikationsserver. Da aber auch die übrigen Funktionen, die sich aus dem Betrieb ergeben, konfiguriert und getestet werden müssen, steht diese Funktion in der Liste an letzter Stelle.

Aufgabe des Kommunikationsservers ist es, geeignete Funktionen zum Testen der konfigurierten Kommunikationsverbindungen bereitzustellen. Eine Möglichkeit wäre die Simulation einer Kommunikationsverbindung, so daß mit Hilfe von Testnachrichten der Verarbeitungsprozeß überprüft werden kann.

3.2 Komponenten eines Kommunikationsservers

Nachdem im letzten Abschnitt die grundlegenden Funktionen eines Kommunikationsservers beschrieben wurden, sollen nun Komponenten eines Kommunikationsservers identifiziert werden. Wie in Abschnitt 2.8 bereits dargestellt, soll hier unter einer *Komponente* der Teil der Kommunikationsserver-Software verstanden werden, der für die Ausführung einer bestimmten Funktionalität zuständig ist. In einer Komponente sollen somit logisch zusammengehörende Funktionen zusammengefaßt werden. Für eine weitere Strukturierung können auch Teilkomponenten gebildet werden. Die Komponenten entsprechen dabei nicht den Programm-Modulen, die bei einer Realisierung der Funktionen bei einer Implementierung gebildet werden (vgl. Abschnitt 2.8).

Die im folgenden aufgeführten Komponenten beinhalten die im Abschnitt 3.1 vorgestellten grundlegenden Funktionen eines Kommunikationsservers. Komponenten für weitergehende Funktionalitäten können hinzukommen, sollen hier aber nicht betrachtet werden. Hinter den Bezeichnungen der Komponenten werden jeweils die Nummern der Funktionen aus Abschnitt 3.1 angegeben, die in dieser Komponente zusammengefaßt werden.

K1: Serverkomponente (F4 - F7):
Die Serverkomponente umfaßt alle Funktionen, die für den Austausch der Nachrichten zwischen den Subsystemen und die Nachrichtenverarbeitung im Kommunikationsserver notwendig sind. Sie läßt sich weiter in eine *Verbindungskomponente* und eine *Verarbeitungskomponente* unterteilen. Die Verbindungskomponente leitet eine empfangene Nachricht zur Verarbeitung an die Verarbeitungskomponente weiter. Diese übermittelt die möglicherweise transformierte Nachricht an die Verbindungskomponente zurück. Die Steuerung und Sicherung des Nachrichtenaustausches zwischen diesen Komponenten sowie die Steuerung des Verarbeitungsprozesses in der Verarbeitungskomponente wird von der *Steuerungskomponente* (K2) übernommen.

K1.1: Verbindungskomponente:
Zur Aufgabe der Verbindungskomponente gehört der Aufbau und der Unterhalt der Verbindungen zwischen dem Kommunikationsserver und den Subsystemen, sowie der Austausch der Nachrichten über diese Verbindungen (F4). Bei einer Implementierung wird diese Komponente in der Regel auf viele einzelne Module verteilt, so daß für jedes Subsystem für die Nachrichtenübertragung ein eigenes Modul zuständig ist. Diese werden als **Subsystem-Agenten** bezeichnet. Auf dieser abstrakten Ebene, sollen aber alle Funktionen zu einer Komponente zusammengefaßt werden.

K1.2: Verarbeitungskomponente:
In der Verarbeitungskomponente werden alle Funktionen für den Verarbeitungsprozeß der Nachrichten im Kommunikationsserver zusammengefaßt. Dazu gehören

- die **Nachrichtenidentifikation** zur Bestimmung der Struktur einer Nachricht (F5),
- das **Routing** mit der Ermittlung der Empfänger der Nachrichten (F6) und gegebenenfalls
- die **Nachrichtentransformation**, die die Nachricht in die Nachrichtenstruktur des Kommunikationsprotokolls des Empfängers überführt (F7).

K2: Steuerungskomponente (F8, F9):
Die Steuerungskomponente umfaßt alle Funktionen, die für die Steuerung des Nachrichtenflusses im Kommunikationsserver notwendig sind. Hierzu gehören die folgenden Funktionen:

- **Fehlermanagement (F8):** Fehler, die bei der Übertragung der Nachrichten auftreten können, werden in der Regel durch die Transportprotokolle in der Verbindungskomponente abgefangen. Logisch gehören die entsprechenden Funktionen aber in diese Komponente. Das Fehlermanagement regelt außerdem die Behandlung von Fehlern, die während der Nachrichtenverarbeitung in der Verarbeitungskomponente auftreten können. So können Fehler in der Nachrichtenstruktur, im Inhalt oder im Typ einer Nachricht vorliegen (siehe auch Abschnitt 4.2.3).

- **Persistenzsicherung (F9):** Nachrichten dürfen innerhalb des Kommunikationsservers nicht verloren gehen. Funktionen, die die Persistenz der Nachrichten im Kommunikationsserver gewährleisten (z.B. Verwaltung von Queues) sind ebenfalls in dieser Komponente angesiedelt.
- Hinzu kommen die Funktionen, die den Nachrichtenfluß zwischen der Verbindungs- und Verarbeitungskomponente sowie in der Verarbeitungskomponente selber regeln.

K3: Überwachungskomponente (F10):
Alle Funktionen, die die Überwachung der Kommunikationsverbindungen (*Monitoring*) ermöglichen, werden in dieser Komponente zusammengefaßt. Sie läßt sich in die folgenden Teilkomponenten zerlegen:

K3.1 Monitorkomponente:
Neben der graphischen Darstellung gehört auch die Überwachung der Kommunikationsverbindungen zur Aufgabe der Funktionen der Monitorkomponente. Allgemein sollten die in dieser Komponente zusammengefaßten Funktionen für den Administrator die Schnittstelle zur Kontrolle des Kommunikationsservers im täglichen Betrieb darstellen.

K3.2 Logging-Komponente:
Nicht nur in der Testphase einer Kommunikationsverbindung ist ein ausführliches Logging aller Vorgänge im Kommunikationsserver wichtig. Anhand einer Log-Datei lassen sich Fehler zum Beispiel im Nachrichten-Routing aufgrund einer falschen Nachrichtenidentifikation nachvollziehen. Auch läßt es sich über die Log-Dateien nachweisen, falls eine Nachricht von einem Subsystem den Kommunikationsserver nicht erreicht hat oder aber erfolgreich bei einem Subsystem abgeliefert wurde. Diese Komponente stellt entsprechende Funktionen zur Verfügung, die das Protokollieren aller Vorgänge in der Serverkomponente ermöglichen.

K3.3 Archivierungskomponente:
Nicht nur der Nachrichtenfluß, sondern auch die Inhalte der Nachrichten müssen zur effektiven Fehleranalyse protokolliert werden können. Dies sollte getrennt von den Log-Dateien in eigenen Nachrichtenarchiven erfolgen. Die hierfür notwendigen Funktionen werden in der Archivierungskomponente zusammengefaßt.

K4: Konfigurationskomponente (F1, F2 und F3, Konfiguration von F4 bis F10):
Alle bisher vorgestellten Komponenten müssen vom Administrator eines Kommunikationsservers konfiguriert werden. Dies gilt vor allem für die Serverkomponente. Aber auch das Logging und das Fehlermanagement bedürfen in der Regel einer Anpassung durch den Administrator. Alle Funktionen, die für die Konfigurierung notwendig sind, werden in dieser Komponente zusammengefaßt.

K5: Testkomponente (F11):
Die Testkomponente stellt allgemein Funktionen zum Testen der über den Kommunikationsserver eingerichteten Kommunikationsverbindungen zur Verfügung. Diese Testfunktionen sollten unabhängig von dem eigentlichen Betrieb des Kommunikationsservers anwendbar sein.

Alle hier beschriebenen Komponenten mit ihren Teilkomponenten und ihren Funktionen werden in der Tabelle 3.1 nochmals zusammengefaßt.

Zu Beginn dieses Abschnitts wurde bereits darauf hingewiesen, daß die vorgestellten Komponenten lediglich eine logische Zusammenfassung der Funktionen eines Kommunikationsservers darstellen. Bei tatsächlich implementierten Kommunikationsservern werden die Funktionen in unterschiedlicher Weise auf Programm-Module verteilt. Eine mögliche Unterteilung eines Kommunikationsservers in einzelne Module soll im folgenden dargestellt werden. Für die Modellierung wurde die OMT-Notation von Rumbaugh et al. [RBP+91] verwendet. Mit dieser Methode können auch die Beziehungen zwischen den einzelnen Modulen und somit zwischen den Komponenten dargestellt werden. OMT wurde gewählt, weil hiermit die Modellierung der statischen Struktur im Objektmodell besonders gut unterstützt wird.

Komponente		Funktionen
K1	Serverkomponente	
K1.1	Verbindungskomponente	Verbindungsaufbau und Nachrichtenaustausch zu den Subsystemen (F4)
K1.2	Verarbeitungskomponente	Nachrichtenidentifikation, Routing, Nachrichtentransformation (F5 - F7)
K2	Steuerungskomponente	Behandlung von fehlerhaften Nachrichten und Fehlern bei der Nachrichtenverarbeitung (F8); Persistenz der Nachrichten bei einem Systemabsturz (F9); Steuerung des Nachrichtenflusses zwischen Verbindungs- und Verarbeitungskomponente und innerhalb der Verarbeitungskomponente (F8)
K3	Überwachungskomponente	
K3.1	Monitorkomponente	Darstellung und Überwachung der Kommunikationsverbindungen (F10)
K3.2	Logging-Komponente	Protokollierung der Vorgänge im Kommunikationsserver (F10)
K3.3	Archivierungskomponente	Archivierung der Nachrichten (F10)
K4	Konfigurationskomponente	Bereitstellung von Funktionen zur Konfigurierung der Server-, Monitor- und Steuerungskomponenten (F1, F2 und F3, Konfigurierung von F4 bis F10)
K5	Testkomponente	Funktionen zum Testen der Kommunikationsverbindungen (F11)

Tabelle 3.1: übersicht über die Funktionen der wichtigsten Komponenten eines Kommunikationsservers.

Als Grundlage für die Modellierung wurde die Analyse verschiedener kommerzieller Kommunikationsserver verwendet (siehe Kapitel 6). Das daraus entstandene Modell (Abbildung 3.1 auf der nächsten Seite) stellt dabei den größten gemeinsamen Nenner der verschiedenen Implementierungen dar. Auf die einzelnen Unterschiede in der Modellierung wird jeweils bei der Beschreibung der Kommunikationsserver im Kapitel 6 eingegangen. Die Zuordnung der Komponenten K1 bis K5 zu den Modulen wird in der Abbildung durch die Umrahmung einzelner Module durch gestrichelte Kästen verdeutlicht. In [Krö96] wird ein weiteres Modell eines Kommunikationsservers in OMT-Notation vorgestellt.

Beschreibung zur Abbildung 3.1

Zentraler Bestandteil des Modells ist das Servermodul, das die Funktionen der Serverkomponente (K1) umfaßt. Bei allen betrachteten Kommunikationsservern werden für die angeschlossenen Subsysteme eigene Module bereitgestellt, die die Funktionen der Verbindungskomponente für diese Subsysteme bereitstellen. Diese Module werden als **Subsystem-Agenten** bezeichnet. Synonym zu Subsystem-Agent wird häufig auch die Bezeichnung *Interface-Agent* verwendet (vgl. [Krö96]). Ein Subsystem-Agent ist in der Regel genau einem Verarbeitungsmodul zugeordnet. Ein Kommunikationsserver kann aus mehr als einem Server-Modul bestehen.

Im Servermodul sind auch die Funktionen aus der Steuerungskomponente (K2) enthalten, die den Nachrichtenfluß innerhalb des Servermoduls steuern. Die weiteren Funktionen der Steuerungskompo-

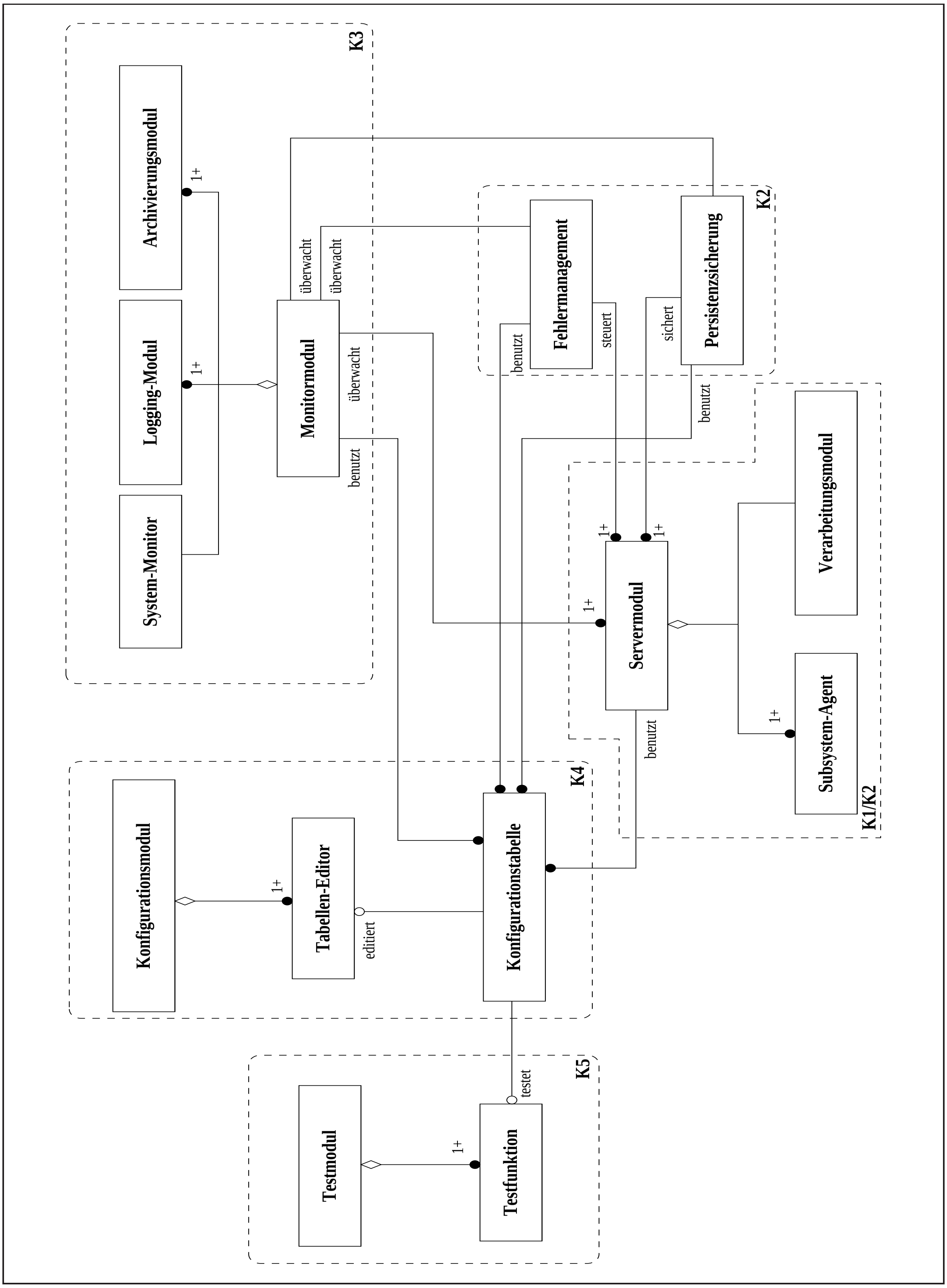


Abbildung 3.1: Beispielhafte Modellierung eines Kommunikationsservers in OMT-Notation. Eine Beschreibung des Modells befindet sich auf der Seite 19.

nente werden durch eigenständige Module implementiert (Module *Fehlermanagement* und *Persistenzsicherung*).

In dem Modell wird davon ausgegangen, daß sämtliche Module des Kommunikationsservers über Tabellen konfiguriert werden. Ihre Funktion ist die Parametrisierung der Kommunikationsverbindungen im Kommunikationsserver. Sie gehören somit logisch zur Konfigurationskomponente (K4). Das Konfigurationsmodul stellt die entsprechenden Werkzeuge (Tabellen-Editoren) bereit.

Die Abbildung der Funktionen der Überwachungskomponente (K3) auf das Monitormodul bedarf kaum einer weiteren Erläuterung. Die Funktionen der Monitorkomponente werden hier durch ein Modul *System-Monitor* realisiert. Diese Bezeichnung soll im weiteren für die Beschreibung dieser Funktionen beibehalten werden.

Die Funktionen der Testkomponente (K5) werden schließlich durch das Testmodul mit den in ihm implementierten Testfunktionen repräsentiert. Da alle Kommunikationsverbindungen über die Konfigurationstabellen parametrisiert werden, greifen die Testfunktionen auch auf diese Tabellen zu. Auf der logischen Ebene testen sie aber nicht die Tabellen, sondern die darin spezifizierten Kommunikationsverbindungen.

3.3 Verfahren der Subsystemanbindung an den Kommunikationsserver

Bei der Anbindung der Subsysteme an den Kommunikationsserver werden von Heitmann [Hei96] *invasive* und *non-invasive* Verfahren unterschieden. Bei dem invasiven Verfahren muß zusätzliche Software in das Subsystem eingebracht werden, um eine Anbindung an den Kommunikationsserver zu ermöglichen. Diese Software kann auch nur aus einem Skript bestehen, das eine Dateiübertragung vornimmt. Bei dem non-invasiven Verfahren ist die Einbringung von fremder Software in das Subsystem nicht notwendig.

Die Begriffe *invasive* und *non-invasive Anbindung* sollen in dieser Arbeit in Anlehnung an [Hei96] wie folgt definiert werden:

Definition 3.1 (Invasive und non-invasive Anbindung)

Die Anbindung eines Subsystems gilt dann als *invasiv*, wenn der für dieses Subsystem zuständige Subsystem-Agent vollständig oder teilweise auf die Plattform des Subsystems eingebracht wird. Die Anbindung eines Subsystems gilt dann als *non-invasiv*, wenn keine Teile des zuständigen Subsystem-Agenten auf die Plattform des Subsystems eingebracht werden.

Auf der logischen Ebene gehören somit alle Programm-Module, die für die Nachrichtenübertragung zwischen einem Subsystem und dem Kommunikationsserver zuständig sind, zum Subsystem-Agenten und fallen somit unter die Administration durch den Kommunikationsserver. In der Praxis stellen aber in der Regel die Rechnergrenzen auch die Grenzen der Zuständigkeitsbereiche dar. Zum Beispiel können die Nachrichten eines Subsystems dateibasiert per FTP zum Rechner des Kommunikationsservers übertragen werden. Dort werden sie dann von dem Subsystem-Agenten aus der Datei ausgelesen und an die Verarbeitungskomponente weitergeleitet. Auf der logischen Ebene ist das Modul, das die Übertragung per FTP vornimmt, ein Bestandteil des Subsystem-Agenten und somit ein Bestandteil des Kommunikationsservers. In der Praxis wird ein solches Modul aber in der Regel durch das Subsystem verwaltet und unterhalten.

Eine andere Situation ergibt sich, wenn ein Subsystem in der Lage ist, Nachrichten über ein dateiloses Transportprotokoll an den Subsystem-Agenten des Kommunikationsservers zu senden. Hier ist die Grenze zwischen Subsystem und Kommunikationsserver an der Schnittstelle, die durch das Transportprotokoll gebildet wird, zu ziehen.

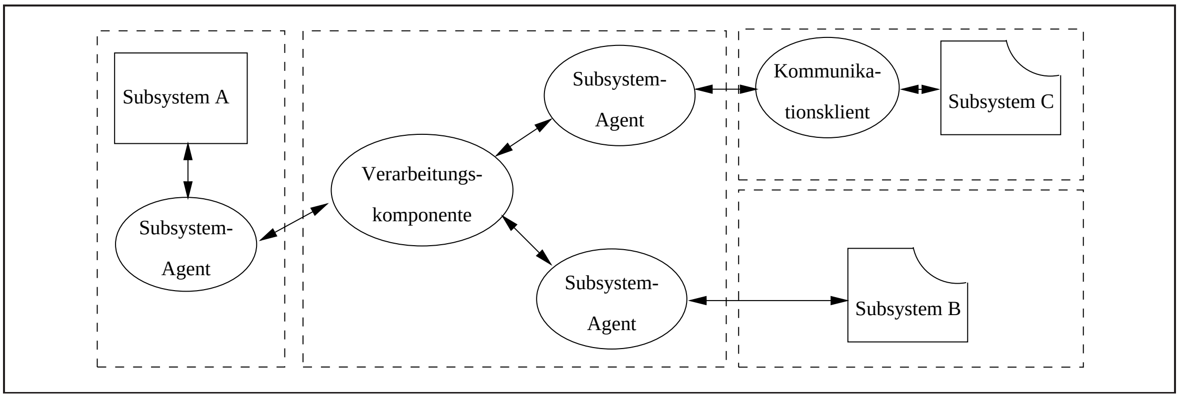


Abbildung 3.2: Invasive versus non-invasive Anbindung der Subsysteme an einen Kommunikationsserver. In der Abbildung sind das Subsystem A invasiv und die Subsysteme B und C non-invasiv mit dem Kommunikationsserver verbunden. Die Subsysteme A und B benötigen keine zusätzliche Software für die Nachrichtenübertragung zum bzw. vom Subsystem-Agenten. Subsystem C wurde um einen Kommunikationsklienten nachträglich erweitert.

Abbildung 3.2 verdeutlicht die beiden unterschiedlichen Anbindungsmethoden. In der Abbildung ist ein Subsystem A invasiv und Subsysteme B und C non-invasiv an einen Kommunikationsserver angebunden. In Anlehnung an [Hei96] soll das Modul des Subsystems, das die Bereitstellung bzw. die Aufnahme der Nachrichten vornimmt, als **Kommunikationsklient** bezeichnet werden.

Im Sinne der Definition von Kommunikationsservern (Definition 2.4 auf Seite 7) ist dieses Modul notwendig, ein Subsystem für den mit den Nachrichten verbundenen Kommunikationsprozeß kommunizierbar zu machen. Die Praxis zeigt, daß die wenigsten Subsysteme in einem Krankenhaus von vornherein kommunizierfähig sind, so daß der Kommunikationsklient in der Regel nachträglich hinzugefügt werden muß. Dies hat zur Konsequenz, daß die meisten Kommunikationsklienten auf einen Kommunikationsprozeß spezialisiert sind. Somit wird für ein Subsystem in der Regel für jeden Kommunikationsprozeß ein eigener Kommunikationsklient benötigt.

Die Trennung zwischen Subsystem-Agent und Kommunikationsklient ist nicht immer vollständig möglich. Mischformen können durchaus vorkommen. So ist es bei einigen Kommunikationsservern (z.B. DataGate^{TM}, Cloverleaf^{TM} und dem pro7-System^{TM}) möglich, Agenten und Klienten vollständig miteinander zu verschmelzen. Dies bedeutet eine Verletzung der in Definition 2.4 eingeführten Zuständigkeitsgrenzen des Kommunikationsservers. Der Subsystem-Agent als Teil des Kommunikationsservers ist nun auch an der Generierung der Nachrichten aus den Daten des Subsystems bzw. an der Importierung der Daten aus der Nachricht in das Subsystem beteiligt.

3.4 Synchronisation der Nachrichtenübertragung über den Kommunikationsserver

Eine Synchronisierung der Nachrichtenübertragung zwischen zwei DV-Systemen kann auf der Ebene der Transportprotokolle sowie auf Nachrichtenebene erfolgen (vgl. Abschnitt 2.7). Durch den Einsatz eines Kommunikationsservers kommunizieren die Subsysteme nicht mehr direkt miteinander. Alle Kommunikationsprozesse laufen über den Kommunikationsserver ab.

Inwieweit in einem Krankenhaus eine Synchronisierung zweier Subsysteme durch den Kommunikationsserver erforderlich ist, ist noch Gegenstand der aktuellen Diskussion. Die meisten Kommunikationsserver sind für eine asynchrone Nachrichtenübertragung (vgl. Abbildung 3.3(a) auf Seite 24) ausgelegt. Dies macht auch durchaus Sinn, da es eine Aufgabe des Kommunikationsservers ist, den

Nachrichtenaustausch zwischen den Subsystemen von der Verfügbarkeit der Empfänger zu entkoppeln. Eine synchrone Nachrichtenübertragung setzt immer voraus, daß alle Kommunikationspartner verfügbar und kommunikationsbereit sind. Ansonsten können sich lange Wartezeiten für die Sender der Nachrichten ergeben, bzw. bestimmte Kommunikationsprozesse erst gar nicht ausführbar sein, da sie ein Subsystem blockieren würden.

Bei allen in dieser Diplomarbeit analysierten Kommunikationsprozessen, die über Kommunikationsserver realisiert wurden, handelt es sich um zeitunkritische Datenübertragungen. So zum Beispiel die Übertragung der Stammdaten der neuaufgenommenen Patienten vom PDV-System an die Stationssysteme, die in den Uni-Kliniken in Münster alle sechs Stunden erfolgt. Hier ist eine asynchrone Übertragung zwischen dem PDV-System und den Stationssystemen angemessen.

Betrachtet man aber zeitkritische Übertragungen, wie zum Beispiel die interaktive Abfrage von Daten aus einer Datenbank über den Kommunikationsserver hinweg, so reicht eine asynchrone Übertragung nicht aus. Wäre das Subsystem, das die Datenbank unterhält, zur Zeit der Anfrage nicht erreichbar, so würde bei einer asynchronen Übertragung der Anwender unter Umständen sehr lange auf eine Antwort seiner Anfrage warten müssen. Bei einer synchronen Übertragung könnte er aber vom Kommunikationsserver benachrichtigt werden, wenn das entsprechende Subsystem zur Zeit nicht erreichbar ist.

Einige Kommunikationsserver (z.B. CLOVERLEAF^{TM}) sehen Maßnahmen zur Synchronisierung zweier Subsysteme vor. Im folgenden sollen die prinzipiellen Möglichkeiten der Synchronisation des Nachrichtenaustausches zwischen zwei Subsystemen durch einen Kommunikationsserver beschrieben werden.

Der Nachrichtenaustausch zwischen einem Subsystem und dem Kommunikationsserver kann weiterhin sowohl auf Transportprotokollebene (Abbildung 3.3(b)) als auch auf Nachrichtenebene (Abbildung 3.3(c)) erfolgen. Bei beiden Methoden muß der Kommunikationsserver in der Lage sein, die entsprechenden Synchronisationsmaßnahmen zu erbringen. Bei der Synchronisation auf Transportprotokollebene wären dies die Generierung und Auswertung von Acknowledgements durch die Subsystem-Agenten. Eine Synchronisation auf der Ebene der Nachrichten macht eine Generierung und Auswertung von Acknowledgement-Nachrichten durch die Verarbeitungskomponente notwendig.

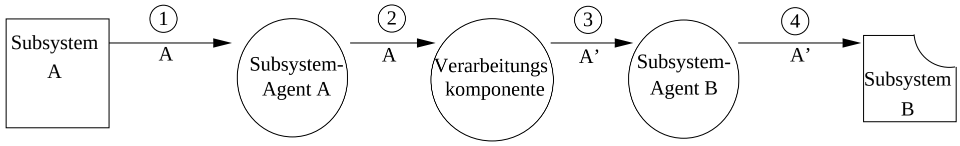
Soll jedoch der Nachrichtenaustausch zwischen zwei Subsystemen synchronisiert werden, so muß diese Synchronisation über den Kommunikationsserver hinweg erfolgen. Der Kommunikationsserver kann dabei eine aktive und eine passive Rolle übernehmen.

Bei der **passiven Synchronisation** erfolgt die Synchronisation der Subsysteme über den Austausch von Achnowledgement-Nachrichten, die von den Subsystemen generiert werden (Abbildung 3.3(d)). Der Kommunikationsserver unterscheidet hierbei nicht zwischen den Nachrichten, die die Daten enthalten, und den Acknowledgement-Nachrichten. Problematisch wird diese Synchronisation, wenn sich Nachrichten von anderen Subsystemen „dazwischen drängeln“, die ebenfalls an die am synchronisierten Kommunikationsprozeß beteiligten Subsysteme bestimmt sind. Da der Kommunikationsserver nicht an den Maßnahmen zur Synchronisation beteiligt ist, wird er diese Nachrichten möglicherweise vor den Acknowledgement-Nachrichten an ein Subsystem übermitteln. Die Subsysteme müssen dann in der Lage sein, zwischen Nachrichten des synchronisierten Kommunikationsprozesses und anderen Nachrichten zu unterscheiden.

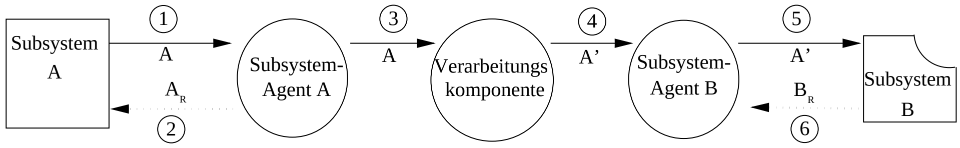
Bei der **aktiven Synchronisation** ist der Kommunikationsserver aktiv an der Synchronisation der Subsysteme beteiligt. Dies kann auf zwei unterschiedliche Weisen erfolgen:

1. Die Acknowledgement-Nachrichten werden weiterhin in den Subsystemen erzeugt. Der Kommunikationsserver behandelt aber die Acknowledgement-Nachrichten eines Kommunikationsprozesses bevorzugt vor allen anderen Nachrichten, die nicht zum Kommunikationsprozeß gehören, aber auch für die beteiligten Subsysteme bestimmt sind. Der Kommunikationsserver muß dann in der Lage sein, zwischen Nachrichten mit Daten und Acknowledgement-Nachrichten zu unterscheiden. Auch muß er eine Kommunikationsverbindung für die Dauer des Synchronisationsprozesses für andere Nachrichten sperren können.

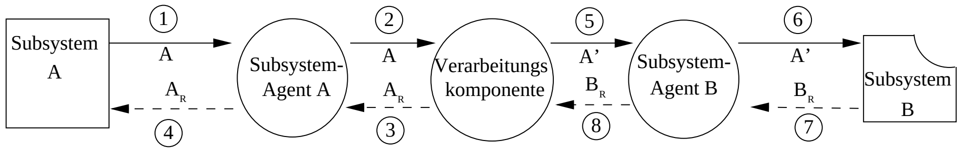
2. Läßt sich in einem Kommunikationsserver die Nachrichtenübertragung zwischen zwei Subsystem-Agenten synchronisieren, so können die beiden entsprechenden Subsysteme auch auf der Ebene



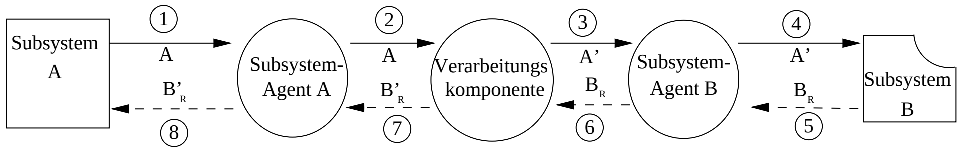
(a) Asynchrone Übertragung



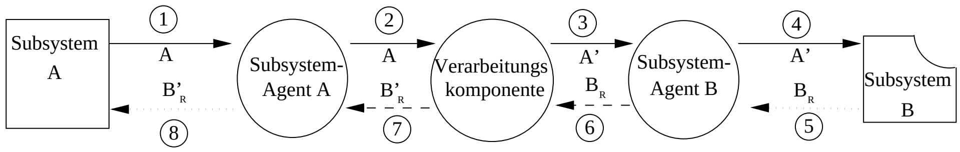
(b) Synchronisierung auf Transportprotokollebene zwischen Subsystem und Kommunikationsserver



(c) Synchronisierung auf Nachrichtenebene zwischen Subsystem und Kommunikationsserver



(d) Synchronisierung auf Nachrichtenebene zwischen zwei Subsystemen



(e) Synchronisierung auf Transportprotokollebene zwischen zwei Subsystemen

Abbildung 3.3: Methoden der Synchronisierung des Nachrichtenaustausches durch einen Kommunikationsserver. Die Zahlen an den Datenflußkanten (① bis ⑧) geben die Reihenfolge des Austausches der Nachrichten zwischen den Prozessen an. Acknowledgement-Nachrichten werden mit einem tiefgestellten „R“ (z.B.: B_R) gekennzeichnet.

der Transportprotokolle synchronisiert werden (siehe Abbildung 3.3(e)). Auf die Möglichkeiten der Synchronisation der beiden Subsystem-Agenten soll hier nicht weiter eingegangen werden. Sie sind sehr stark von der Implementierung des Kommunikationsservers abhängig. In Abbildung 3.3(e) wird eine Synchronisation der Subsystem-Agenten über ausgetauschte Nachrichten gezeigt. Dies macht eine Generierung bzw. Interpretation von entsprechenden Nachrichten durch die Subsystem-Agenten notwendig.

Abbildung 3.3 zeigt schematisch die verschiedenen Möglichkeiten der Synchronisation der Nachrichtenübertragung über einen Kommunikationsserver. Die verwendete Notation entspricht weitestgehend der im Anhang A.2 dargestellten Notation für Datenflußdiagramme. Die Darstellung der Pfeile zwischen den Prozessen besitzt allerdings eine geänderte Bedeutung, die in der Legende der Abbildung erklärt ist.

3.5 Darstellungsformat der Nachrichten im Kommunikationsserver

Die im Abschnitt 1.1 dargestellte Motivation für die Einführung eines Kommunikationsservers in ein KKS leitete sich aus der Reduzierung der Schnittstellen zwischen den Subsystemen ab. Diese Reduzierung ist aber nur auf den ersten Blick richtig. Die Reduzierung der Schnittstellenanzahl gilt nur für die Ebenen 4 bis 6 des ISO/OSI-Modells. Bei Einsatz eines Kommunikationsservers muß für jedes Subsystem nur noch eine physikalische Verbindung und ein Transportprotokoll eingerichtet werden, um es in das KKS einzubinden. Dies trifft aber nicht für die 7. Ebene des ISO/OSI-Modells, der Ebene der Kommunikationsprotokolle, zu.

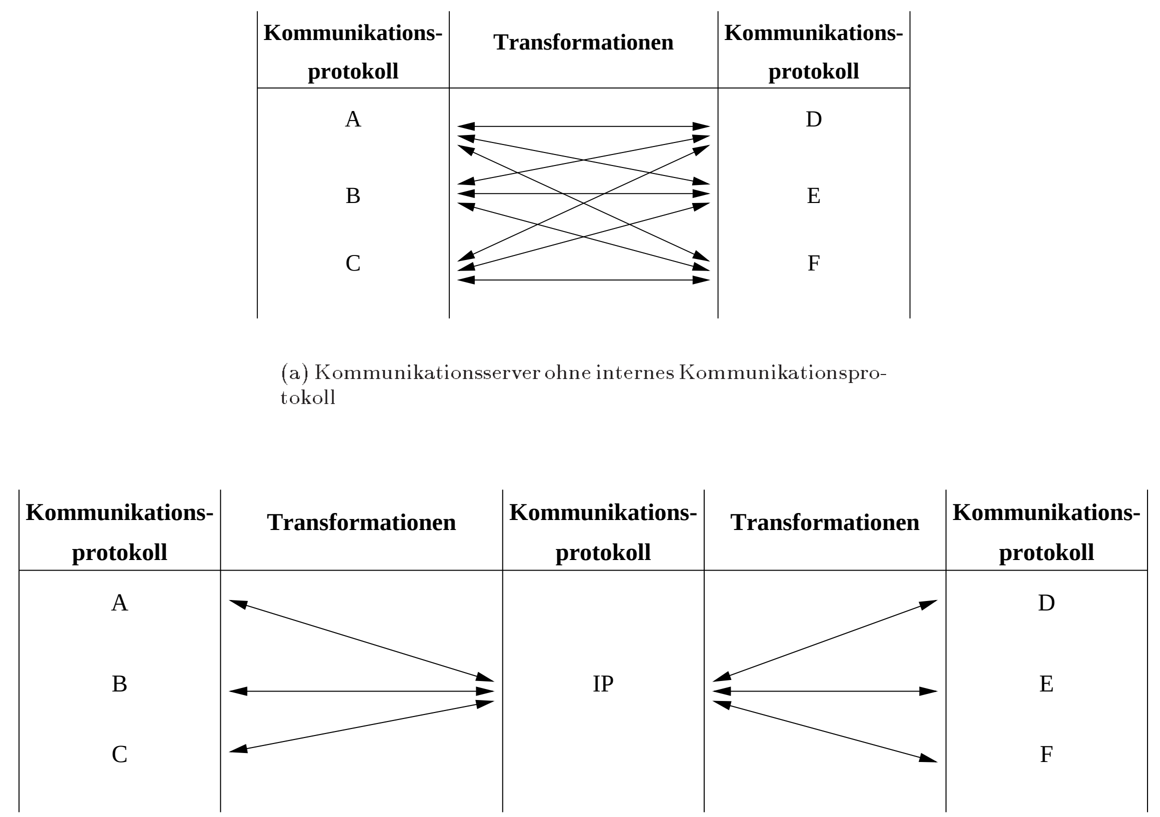
Faßt man die Transformation zwischen zwei Kommunikationsprotokollen als eine Schnittstelle auf, so müssen auch bei Einsatz eines Kommunikationsservers zwischen allen Kommunikationspartnern Schnittstellen auf dieser Ebene implementiert bzw. definiert werden. Zwar bietet der Kommunikationsserver hierfür eine einheitliche Implementierungsumgebung, so daß die Wartung sehr vereinfacht wird, doch ändert dies nichts an der Anzahl der Schnittstellen und dem damit verbundenen administrativen Aufwand. Auf der Ebene der Kommunikationsprotokolle bleiben somit die alten Punkt-zu-Punkt-Beziehungen zwischen den Subsystemen erhalten.

Werden allerdings alle Nachrichten intern im Kommunikationsserver in einem einheitlichen Kommunikationsprotokoll dargestellt, so kann auch auf der Ebene der Kommunikationsprotokolle die angestrebte Reduzierung der Anzahl der Schnittstellen erreicht werden. Dazu werden alle Nachrichten, die der Kommunikationsserver empfängt, nach der Identifikation in das intern verwendete Kommunikationsprotokoll transformiert. Nach dem Routing folgt dann eine Transformation der Nachrichten aus dem internen Kommunikationsprotokoll in die entsprechenden Protokolle der Empfänger. Ein solches intern verwendetes Kommunikationsprotokoll soll als **internes Kommunikationsprotokoll** bezeichnet werden.

Definition 3.2 (Internes Kommunikationsprotokoll)

Ein *internes Kommunikationsprotokoll* ist ein Kommunikationsprotokoll, das vom Kommunikationsserver intern für die Darstellung der Nachrichten von allen Subsystemen verwendet wird.

Die Verwendung eines internen Kommunikationsprotokolls bedeutet zwar, daß jede Nachricht zweimal transformiert werden muß, insgesamt ergibt sich aber für den Kommunikationsserver und somit für das KKS eine erhebliche Reduzierung der Anzahl der Schnittstellen auf dieser Ebene ($O(n)$ statt bisher $O(n^2)$). Abbildung 3.4 verdeutlicht diese Reduzierung der Schnittstellenanzahl. Die Pfeile in der Abbildung stellen jeweils einen (bidirektionalen) Transformationsprozeß zwischen zwei Kommunikationsprotokollen dar. Dabei zeigt sich deutlich der Punkt-zu-Punkt Charakter der Transformationen, wenn kein internes Protokoll verwandt wird (Abbildung 3.4(a)). Die Auflösung dieser Punkt-zu-Punkt-Beziehungen durch die Verwendung eines internen Protokolls zeigt Abbildung 3.4(b).



(a) Kommunikationsserver ohne internes Kommunikationsprotokoll

(b) Kommunikationsserver mit internem Kommunikationsprotokoll

Abbildung 3.4: Auflösung der Punkt-zu-Punkt Beziehungen zwischen den Subsystemen auf der Ebene der Kommunikationsprotokolle durch die Verwendung eines internen Kommunikationsprotokolls (IP) für die Darstellung der Nachrichten im Kommunikationsserver. Die Pfeile zwischen den Nachrichtenstrukturen stellen jeweils eine (bidirektionale) Transformation zwischen den Kommunikationsprotokollen dar.

Die Verwendung eines internen Kommunikationsprotokolls hat aber auch Nachteile. Zum einen bedeutet der Mehraufwand der doppelten Transformation jeder Nachricht eine Verzögerung des Nachrichtenflusses über den Kommunikationsserver. Wird ein standardisiertes Kommunikationsprotokoll als internes Protokoll verwendet (z.B. HL7 bei PRO7-SYSTEM^{TM} und HL7-CONNECTION-SERVER^{TM}) und wird dieser Standard auch von vielen Subsystemen eingesetzt, so können viele Transformationen entfallen. Dies trifft allerdings auch zu, wenn vom Kommunikationsserver kein internes Kommunikationsprotokoll verwendet wird.

Für die Subsysteme, die nicht diesen Standard verwenden, kann das interne Kommunikationsprotokoll aber eine Einschränkung bedeuten. Sie können nur Nachrichten verwenden, die sich auch im internen Protokoll darstellen lassen. Die Abbildungen zwischen dem internen Protokoll und den Kommunikationsprotokollen der Subsysteme müssen immer bijektiv und vollständig sein.

Auch Kommunikationsserver, die standardmäßig kein internes Kommunikationsprotokoll verwenden, lassen sich so konfigurieren, daß ein internes Protokoll simuliert wird. Dies sei am Beispiel des Kommunikationsservers DataGate^{TM} gezeigt. DataGate^{TM} erlaubt die Einrichtung sogenannter „interner Puffer" (*internal Buffers*). Diese Puffer werden vom Kommunikationsserver wie Subsystem-Agenten behandelt (in DataGate^{TM} als *Communication Clients* bezeichnet). Sie stellen allerdings alle Nachrichten, die sie vom Verarbeitungsmodul erhalten, direkt als neue Nachrichten wieder in den Verarbeitungsprozeß zurück. Werden die Nachrichten aller Subsysteme über diesen internen Puffer geleitet, so läßt sich ein internes Kommunikationsprotokoll simulieren in dem alle Nachrichten der Subsysteme in das interne Kommunikationsprotokoll transformiert und zum internen Puffer weitergeleitet werden. Dieser schickt die Nachrichten direkt wieder an das Verarbeitungsmodul zurück, wo erst jetzt die eigentlichen Empfänger ermittelt und die Transformationen in die Kommunikationsprotokolle der Empfänger durchgeführt werden. Abbildung 3.5 verdeutlicht den Nachrichtenfluß über den internen Puffer. In der Abbildung werden die Nachrichten nach ihren Sendern bezeichnet (Subsystem A sendet Nachricht A). Wird eine Nachricht in ein anderes Kommunikationsprotokoll transformiert, so erhält der Bezeichner der Nachricht ein Index, der das neue Kommunikationsprotokoll anzeigt (z.B. steht A_B für eine Nachricht vom Subsystem A, die im Kommunikationsprotokoll des Subsystems B vorliegt). Bei Nachrichten, die im Original-Kommunikationsprotokoll vorliegen, wird auf dieses Index verzichtet.

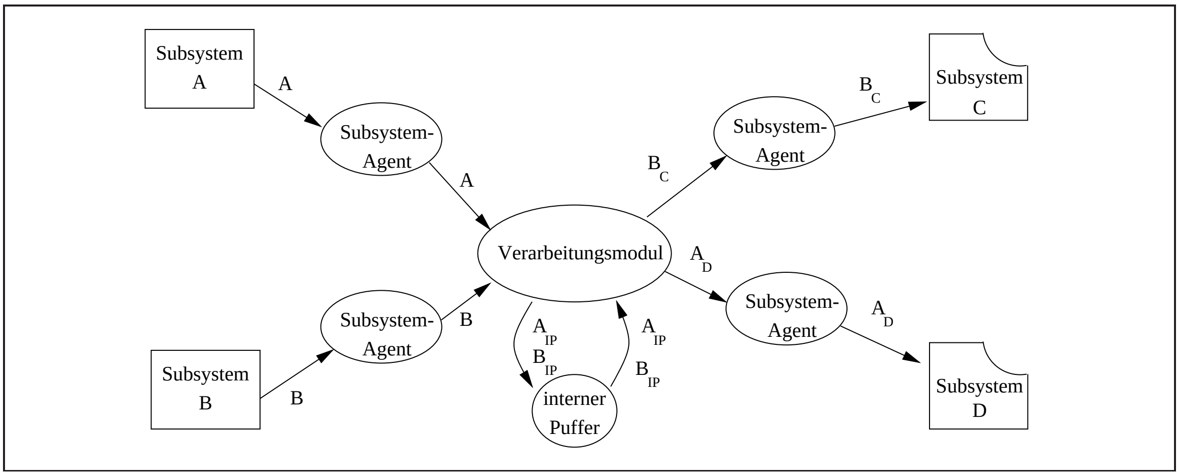


Abbildung 3.5: Simulation eines internen Kommunikationsprotokolls mit dem Kommunikationsserver DataGate^{TM}. Die Indizes der Bezeichnern der Nachrichten geben das Kommunikationsprotokoll an, in der die Nachricht vorliegt (z.B. A_B bedeutet: Nachricht A liegt im Kommunikationsprotokoll des Subsystems B vor).

Kapitel 4

Anforderungskatalog

Nachdem im Kapitel 3 die grundlegenden Funktionen und deren Zuordnung zu Komponenten eines Kommunikationsservers beschrieben wurden, sollen nun die Funktionen der Komponenten detaillierter dargestellt werden. Zusätzlich werden aus den Funktionen Anforderungen abgeleitet, die ein Kommunikationsserver für eine Unterstützung der Funktionen erfüllen sollte. Neben allgemeinen Anforderungen werden auch solche berücksichtigt, die sich aus dem speziellen Umfeld „Krankenhaus“ ergeben. Diese werden im folgenden durch den Zusatz „(S)“ vor der Nummer der Anforderung gekennzeichnet.

Der Gliederung des Katalogs ergibt sich aus den Komponenten eines Kommunikationsservers, die im Abschnitt 3.2 vorgestellt wurden. Hinzu kommen noch zwei Abschnitte, die den Schutz der Nachrichten im Kommunikationsserver vor einem unbefugten Zugriff (Abschnitt 4.6 ab Seite 42) und den direkten Anschluß eines Datenbankmanagementsystems (DBMS) an einen Kommunikationsserver betreffen (Abschnitt 4.7 ab Seite 43). Die im letzten Abschnitt beschriebenen Anforderungen gehören nicht zu den grundsätzlichen Aufgaben eines Kommunikationsservers, sie können sich aber aus der zentralen Stellung des Servers im KKS ergeben.

Die Anforderungen sind möglichst allgemein gehalten. Sie sollen eine spezielle software-technische Realisierung nicht vorschreiben. Jede Anforderung wird jeweils in den vorangehenden Ausführungen aus den Funktionen der Komponente abgeleitet und erläutert.

Der Anforderungskatalog wurde ursprünglich auch als Fragekatalog angelegt. Zu jeder Anforderung gehören eine oder mehrere Fragen, die die konkrete Umsetzung der Anforderung in einem speziellen Kommunikationsserver betreffen. Mit Hilfe dieses Fragebogens wurden im Rahmen dieser Arbeit sechs verschiedene Kommunikationsserver analysiert. Die Ergebnisse stellen die Grundlage zu einem umfassenden Vergleich der Kommunikationsserver dar (siehe Kapitel 6). In den folgenden Abschnitten werden diese Fragen aus Gründen der Lesbarkeit allerdings weggelassen. Sie werden im Abschnitt C mit jeweils den Antworten zu den einzelnen Kommunikationsservern dargestellt.

4.1 Anforderungen an die Serverkomponente

4.1.1 Unterstützung der Transportprotokolle

Im Abschnitt 2.5 wurde schon einiges über Transportprotokolle gesagt. Sie sind für die Übertragung der Nachrichten zwischen den Subsystemen und der Verbindungskomponente des Kommunikationsservers zuständig. In dieser Arbeit werden grundsätzlich dateibasierte und dateilose Transportprotokolle unterschieden. In diesem Abschnitt werden nun Anforderungen an die Unterstützung der Transportprotokolle durch einen Kommunikationsserver aufgestellt.

Welche Transportprotokolle für den Anschluß eines bestimmten Subsystems verwendet werden können, wird in der Regel durch die Möglichkeiten des Nachrichtenaustausches im Subsystem bestimmt. Wie in Abschnitt 3.3 bereits dargestellt wurde, sind die meisten Subsysteme in einem Krankenhaus vorerst nicht kommunizierfähig. Sie müssen durch Kommunikationsklienten erweitert werden, die die Nachrichten generieren bzw. aufnehmen. Die Nachrichten müssen nun zwischen dem Kommunikationsklienten und der Verbindungskomponente, also dem zuständigen Subsystem-Agenten, ausgetauscht werden. Dies kann entweder direkt über ein dateiloses oder indirekt über ein dateibasiertes Transportprotokoll geschehen. Für beide Transportprotokollarten gibt es definierte Standardprotokolle. So zum Beispiel FTP (*file transmission protocol*) für die dateibasierte oder TCP-Sockets für die dateilose Übertragung. Diese Protokolle können, wie in Abschnitt 2.5 beschrieben, durch Maßnahmen zur Sicherung und Synchronisation der Übertragung (5. Ebene des ISO/OSI-Modells) erweitert werden. Auch hier gibt es Standards für die einzelnen Transportprotokolle. Ein Beispiel ist das im Anhang D.2 beschriebene *HL7 Lower Level Protocol*, das für dateilose Transportprotokolle angewendet werden kann. Desweiteren muß in den Protokollen eine Darstellungsform der Nachrichten (6. Ebene des ISO/OSI-Modells) definiert werden. Im *HL7 Lower Level Protocol* werden zum Beispiel Steuerzeichen definiert, die den Beginn und das Ende einer Nachricht anzeigen (*Envelope*). Für TCP-Sockets wird häufig eine Längencodierung der Nachrichten verwendet, in der die ersten zwei oder vier Bytes des Byte-Streams die Länge der gesamten Nachricht anzeigen. Eine solche Längencodierung läßt sich auch für die dateibasierten Transportprotokolle anwenden, wenn in einer Datei mehr als eine Nachricht übertragen wird. Hier wird häufig auch das Zeilenendezeichen als Nachrichtentrennzeichen verwendet.

Für eine flexible Übertragung der Nachrichten zwischen dem Kommunikationsklienten und dem Subsystem-Agenten sollte der Kommunikationsserver möglichst viele der standardisierten Transportprotokolle unterstützen. Sie sollten dabei vom Administrator für die Subsystem-Agenten eingerichtet und konfiguriert werden können.

Die Erfahrungen, die bei dieser Diplomarbeit gewonnen wurden, zeigen, daß auch ein umfangreicher Satz an Standardprotokollen nicht ausreicht, alle Subsysteme adäquat mit dem Kommunikationsserver zu verbinden. Dies trifft vor allem auf die heterogene Umgebung in einem Krankenhaus zu. Viele, besonders ältere Subsysteme, machen spezielle Transportprotokolle notwendig. Ein Kommunikationsserver kann nicht alle Variationen der Protokolle anbieten. Daher ist es wichtig, daß neben den Standards auch andere Protokolle vom Kommunikationsserver unterstützt werden. Es muß also die Möglichkeit bestehen, zum Beispiel auf standardisierten Transportprotokollen andere Sicherungsmechanismen und Nachrichtendarstellungsformen aufzusetzen und in den Kommunikationsserver einzuhängen. Auch die Einbindung weiterer nicht standardisierter Transportprotokolle sollte möglich sein. Die Implementierung und Einbindung dieser Protokolle sollte auch vom Anwender vorgenommen werden können.

Anforderungen:

1.1: *Unterstützung standardisierter Transportprotokolle.*

1.2: *Unterstützung verschiedener Nachrichtendarstellungsformen und Sicherungsmaßnahmen.*

1.3: *Konfigurierung der Transportprotokolle durch den Anwender.*

1.4: *Unterstützung weiterer Transportprotokolle.*

4.1.2 Nachrichtenidentifikation

Der erste Schritt der Nachrichtenverarbeitung im Kommunikationsserver besteht aus der Nachrichtenidentifizierung, also der Ermittlung des Nachrichtentyps. Der Nachrichtentyp bestimmt die Struktur einer Nachricht, auf die die Verarbeitungskomponente in den folgenden Verarbeitungsschritten zugreifen muß.

Der Typ einer Nachricht wird im allgemeinen in bestimmten Datenfeldern der Nachrichten codiert (siehe auch Abschnitt 2.4). So wird zum Beispiel in HL7 eine Nachricht vom Typ „*Neuaufnahme*" über ein Feld im Segment „`MSH`" (Typ: `ADT`) und ein Feld im Segment „`EVN`" (Ereignis: `A01`) identifiziert.

Auch in proprietären Kommunikationsprotokollen werden in der Regel Datenfelder für die Codierung der Nachrichtentypen vorgesehen.

Für die Nachrichtenidentifizierung muß also der Kommunikationsserver auf den Inhalt einiger Datenfelder der Nachricht zugreifen können. Die Reglen für die Nachrichtenidentifizierung werden zusammen mit den Nachrichtenstrukturen in den Kommunikationsprotokollen festgelegt. Die Implementierung dieser Regeln müssen bei der Einrichtung einer Kommunikationsverbindung vom Administrator im Kommunikationsserver vorgenommen werden.

Viele Kommunikationsserver benutzen den Nachrichtentyp auch für die Bestimmung der Empfänger der Nachrichten (siehe Abschnitt 4.1.3). Hier wird häufig eine Schlüsselwortmethode angewendet: Die Identifizierungsfunktion gibt nicht den Nachrichtentyp zurück, sondern ein definiertes Schlüsselwort, das über die Auswertung bestimmter Datenfelder einer Nachricht ermittelt wird. Hierbei können sehr komplexe Identifizierungsregeln entstehen, für deren Implementierung der Kommunikationsserver neben Funktionen für den Zugriff auf Datenfelder auch Vergleichsoperationen und Kontrollstrukturen bereitstellen sollte.

In standardisierten Kommunikationsprotokollen sind die Identifizierungsregeln für die Nachrichten im Standard festgeschrieben. Bei diesen Protokollen ist eine automatische Identifzierung der Nachrichten durch den Kommunikationsserver denkbar.

Da Nachrichten mit gleichem semantischen Inhalt von unterschiedlichen Subsystemen in verschiedenen Kommunikationsprotokollen an den Kommunikationsserver gesandt werden können, sollte der Kommunikationsserver die Verwaltung von einheitlichen Bezeichnungen der Nachrichtentypen für alle Subsysteme unterstützen. So sollte zum Beispiel eine Nachricht, die die Neuaufnahme eines Patienten anzeigt, einheitlich unter dem Nachrichtentyp „*Neuaufnahme*“ geführt werden können, unabhängig davon, in welchem Kommunikationsprotokoll die Nachricht vorliegt.

Bei sehr umfangreichen Kommunikationsprotokollen mit vielen verschiedenen Nachrichten lassen sich oft Hierarchien der Nachrichtentypen bilden. So werden in HL7 37 verschiedene ADT (*Admission, Discharge and Transfer*) Nachrichten unterschieden (`A01` bis `A37`), die sich alle auf die Aufnahme und Entlassung eines Patienten beziehen. Eine Nachbildung dieser Hierarchie bei den Nachrichtentypen würde eine Handhabung der Nachrichten bei der Verarbeitung im Kommunikationsserver erleichtern. So könnten über den Nachrichtentyp „ADT“ alle Nachrichten dieser Klasse angesprochen werden, während der Subtyp „A01“ eine spezielle Nachricht der Klasse meint. Eine Unterstützung solcher Hierarchien setzt voraus, daß Nachrichten mit mehr als einem Nachrichtentyp versehen werden können (Obertyp, Subtyp, etc.).

Auch bei eigenen Kommunikationsprotokollen sind hierarchische Nachrichtentypen denkbar und sollten vom Kommunikationsserver unterstützt werden.

Anforderungen:

1.5: *Konfigurierbare Nachrichtenidentifikation zur Ermittlung der Struktur einer Nachricht.*

1.6: *Unterstützung einer automatisierten Nachrichtenidentifikation bei standardisierten Kommunikationsprotokollen.*

1.7: *Unterstützung einheitlicher Bezeichner für gleiche Nachrichtentypen in unterschiedlichen Kommunikationsprotokollen.*

1.8: *Unterstützung von Hierarchien bei Nachrichtentypen.*

4.1.3 Bestimmung der Empfänger der Nachrichten

Der zweite Schritt in der Verarbeitung der Nachrichten durch den Kommunikationsserver besteht aus der Bestimmung der Empfänger einer Nachricht (*Routing*). Dabei lassen sich drei verschiedene

Möglichkeiten des Routings unterscheiden:

1. Alle Nachrichten von einem Sender werden grundsätzlich zu bestimmten Empfängern gesandt,
2. die Empfänger einer Nachricht sind implizit durch den Nachrichtentyp festgelegt und
3. die Empfänger lassen sich aus dem Inhalt bestimmter Datenfelder der Nachricht entnehmen.

Während die erste Möglichkeit als **statisches Routing** bezeichnet wird, stellen die zweite und dritte Möglichkeit ein **dynamisches Routing** dar.

So kann zum Beispiel in einem Krankenhaus feststehen, daß alle Nachrichten vom Typ „*Neuaufnahme*“ an das Laborsystem gesandt werden. Zusätzlich könnte die obige Nachricht auch zum Stationssystem der Krankenstation gesandt werden, auf die der Patient aufgenommen werden soll. Während der erste Empfänger, das Laborsystem, durch den Nachrichtentyp festgelegt ist, muß die Ermittlung des zweiten Empfängers aus dem Nachrichteninhalt erfolgen.

Für die Routing-Methode über die Ermittlung der Empfänger durch den Nachrichtentyp werden im Kommunikationsserver Routing-Tabellen verwendet. Diese Tabellen sind so aufgebaut, daß für jeden Sender einer Nachricht über den Nachrichtentyp eindeutig die Empfänger bestimmt werden können. Diese Tabellen werden zum Zeitpunkt der Konfigurierung der Kommunikationsverbindungen definiert und ändern sich während der Laufzeit des Kommunikationsservers nicht.

Bei der dritten Routing-Methode müssen im Kommunikationsserver Regeln definiert werden, die aus dem Inhalt spezieller Datenfelder der Nachrichten die Empfänger ermitteln. Bei dieser Methode werden die Routing-Informationen nicht aus statischen Tabellen, sondern erst zur Laufzeit aus dem Nachrichteninhalt ermittelt.

Ein Kommunikationsserver sollte alle drei Verfahren des Routings unterstützen.

Einige Kommunikationsserver (z.B. DataGate^{TM} und Cloverleaf^{TM}) bieten ein Routing an, das das zweite und dritte Verfahren miteinander verbindet. Die Nachrichtenidentifikation liefert hierbei ein Schlüsselwort, das sich aus dem Nachrichtentyp und den aus Datenfeldern der Nachrichten bestimmten Empfängern herleitet. Die Regeln der Nachrichtenidentifizierung werten dabei neben den Datenfeldern, die den Nachrichtentyp bestimmen, auch die Datenfelder aus, die die Empfänger der Nachricht festlegen. Die Subsysteme der Empfänger lassen sich dann über Routing-Tabellen mit Hilfe des Schlüsselwortes ermitteln. So kann zum Beispiel bei diesem Verfahren das Ergebnis der Nachrichtenidentifzierung für eine Nachricht vom Typ „Neuaufnahme“ aus den Schlüsselwörtern „Neuaufnahme_Station_A“ und „Neuaufnahme_Station_B“ bestehen. Über Routing-Tabellen kann dann aus dem ersten Schlüsselwort das Subsystem der Station A bzw. aus dem zweiten das der Station B ermittelt werden.

Mit diesem Verfahren lassen sich die drei oben aufgeführten Routing-Methoden simulieren. Für alle möglichen Kombinationen aus Sender, Nachrichtentyp und Empfänger der Nachrichten werden Schlüsselwörter und die zugehörenden Regeln definiert. Ein statisches Routing läßt sich über ein spezielles Schlüsselwort (z.B.:`static`) erreichen.

Mit der Schlüsselwort-Methode läßt sich leicht ein *Broadcasting* einer Nachricht realisieren. Für das Schlüsselwort „*Broadcast*“ werden in der Routing-Tabelle alle Subsysteme als Emfänger vorgesehen. Auch ein *Multicasting* einer Nachricht läßt sich auf diese Weise erreichen. Hier wären aber für alle unterschiedlichen Empfängergruppen eigene Schlüsselwörter zu definieren.

Sind viele Subsysteme über den Kommunikationsserver miteinander verbunden, können die Routing-Tabellen schnell umfangreich werden. Eine graphische Präsentation und Definition dieser Tabellen bietet sich an. Dabei sollte der Umfang der Darstellung konfigurierbar sein, zum Beispiel Darstellung nur der Verbindungen, die von einem Subsystem ausgehen, oder nur der Verbindungen, über die ein bestimmter Nachrichtentyp versandt wird. Besonders anschaulich ist eine Darstellung der Routing-Tabellen in Form eines Netzwerkes. Die Subsysteme stellen dabei die Knoten dar und die kanten repräsentieren einen Nachrichtenfluß zwischen den Subsystemen. In der Darstellung sollte der

Kommunikationsserver weggelassen werden, da in dieser abstrakten Sicht die Kommunikationsverbindungen als *Punkt-zu-Punkt*-Verbindungen aufgefaßt werden.

Eine Kommunikationsverbindung, bei der die Empfänger der Nachrichten ausschließlich über den Nachrichteninhalt ermittelt werden (Methode 3), ohne daß vordefinierte Schlüsselwörter und Routing-Tabellen benutzt werden, läßt sich in der Netzwerk-Darstellung nicht abbilden. Zum Zeitpunkt der Konfigurierung werden lediglich die Regeln für die Ermittlung der Empfänger definiert. Empfänger können aber alle Subsysteme des KKS sein.

Anforderungen:

1.9: *Unterstützung einer statischen und dynamischen Ermittlung der Empfänger einer Nachricht.*

1.10: *Leichte Definition der Routing-Tabellen.*

1.11: *Konfigurierbare, graphische Darstellung der Routing-Informationen*

4.1.4 Transformation der Nachrichten

Stehen die Empfänger einer Nachricht fest, so erfolgt nun gegebenfalls eine Transformation der Nachricht in die Kommunikationsprotokolle der Empfänger. Dazu erstellt der Kommunikationsserver für jeden Empfänger eine Kopie der Nachricht und führt an dieser die Transformation durch. Für die Transformationen müssen vom Administrator für jede Nachricht aus dem Kommunikationsprotokoll des Senders Reglen definiert werden, wie die Nachrichten in die Kommunikationsprotokolle der Empfänger überführt werden müssen. Die Regeln lassen sich als eine Abbildung zwischen dem Kommunikationsprotokoll des Senders und dem Kommunikationsprotokoll jeweils eines Empfängers auffassen. Diese Abbildung ist in der Regel pariell, da meistens nicht alle Nachrichten des Kommunikatonsprotokolls des Senders an diesen Empfänger gesandt werden. Sie muß lediglich für die tatsächlich ausgetauschte Teilmenge von Nachrichten vollständig sein. Da die Abbildung häufig nicht bijektiv ist, müssen für beide Richtungen der Transformation eigene Abbildungen definiert werden.

Bei der Transformation der Nachrichten soll hier eine *Transformation der Nachrichtenstruktur* und eine *Wertkonvertierung* unterschieden werden.

Transformation der Nachrichtenstruktur

Bei der Transformation der Nachrichtenstruktur wird der Aufbau der Nachricht aus Datenfeldern und Segmenten verändert. Die Reihenfolge der Datenfelder in den Segmenten oder eine komplette Umstrukturierung der Nachricht in andere Segmente kann erfolgen. Die Feldlänge der Datenfelder kann verändert oder von einer fixen in eine variable Form trnasfromiert werden. Es ist auch möglich, daß mehrere Datenfelder zu einem zusammengefaßt oder einzelne Datenfelder gesplittet werden.

Für die Definition dieser Regeln muß also auf die Struktur der Nachrichten zugegriffen werden. Die Definition kann durch spezielle Editoren, die die Nachrichtenstrukturen des Senders und des Empfängers darstellen, wesentlich erleichtert werden. Es gilt aber, daß die Definition der Transformationen nicht von einem speziellen Editor abhängig sein sollte. Dies wird durch die Bereitstellung einer speziellen Beschreibungssprache für die Definition der Regeln ermöglicht.

Bei der Verwendung standardisierter Kommunikationsprotokolle ist eine automatisierte Transformation der Nachrichtenstrukturen denkbar. Dem Administrator bleibt bei der Konfigurierung einer Kommunikationsverbindung viel Arbeit erspart, wenn der Kommunikationsserver zum Beispiel eine Transformation von HL7 zu Edifact anbietet. Weicht dabei das Format der Nachrichten vom Standard ab, wäre eine Vortransformation notwendig. Zum Beispiel kann bei HL7 in der lokal verwendeten Version das Geschlecht einer Patientin über „`W`“ für „weiblich“ kodiert sein, während der Standard hier ein „`F`“ vorsieht. Auch die Erweiterung des Standards um Z-Segmente müßte getrennt behandelt werden.

Anforderungen:

1.12: *Leichte Implementierung der Funktionen zur Transformation der Nachrichten in andere Nachrichtenstrukturen.*

1.13: *Unterstützung einer automatisierten Transformation bei standardisierten Kommunikationsprotokollen.*

Wertkonvertierung

Neben der Transformation der Nachrichten in andere Nachrichtenstrukturen kann auch eine Transformation der Inhalte einzelner Felder notwendig werden. Diese Transformationen sollen hier als **Wertkonvertierungen** bezeichnet werden. Ein Beispiel ist die Konvertierung der Darstellungsform einer Datumsangabe oder die Umwandlung eines Zahlwertes in einen Text.

In der Regel erfolgt eine Wertkonvertierung gleichzeitig mit der Strukturtransformation. Sie kann aber auch isoliert durchgeführt werden, wenn zum Beispiel lediglich Transformationen der Datenfeldbelegungen bei sonst gleicher Nachrichtenstruktur durchgeführt werden sollen. Verwenden beide Subsysteme zum Beispiel HL7, aber mit unterschiedlichen Ausprägungen einzelner Datenfelder, so muß bei der Transformation lediglich eine Anpassung dieser Datenfelder erfolgen.

Da in Zukunft im Krankenhaus immer mehr multimediale Daten übertragen werden, sollte der Kommunikationsserver neben den allgemeinen Konvertierungsfunktionen (z.B.: Datumsformatkonversion) auch spezielle Funktionen zur Konvertierung von Graphik- und Audiodaten in andere Darstellungsformate (GIF, JPEG, MPEG Audio etc.) anbieten. Reichen für spezielle Konvertierungen die vom Kommunikationsserver angebotenen Funktionen nicht aus, sollten auch extern implementierte Konvertierungsfunktionen aufrufbar sein.

Lookup-Tabellen sind ein weiteres Mittel zur Konvertierung einzelner Felderinhalte. Mit ihrer Hilfe können zum Beispiel Zahlen-Codes (z.B. ICD-Code) in eine textuelle Beschreibung umgewandelt werden. Lookup-Tabellen sind universeller einsetzbar, wenn sie auch bidirektional angelegt werden können. Ein spezieller Editor erleichtert ihre Definition. Sie sollten aber auch ohne den Editor vom Anwender aufgestellt werden können.

Bei nicht standardisierten Kommunikationsprotokollen kann die Konvertierung einzelner Datenfelder abhängig von dem Inhalt anderer Nachrichtensegmente sein. Eine Transformation kann sich über mehrere Datenelder oder Segmente erstrecken. Kommunikationsprotokolle können Wiederholungen von Segmenten oder optionale Segmente vorsehen (z.B. bei HL7). Für die Implementierung der Transformationsregeln sollten also Kontrollstrukturen wie Schleifen und `if-then`-Konstrukte angeboten werden.

Anforderungen:

1.14: *Bereitstellung von allgemeinen und speziellen Konvertierungsfunktionen.*

1.15: *Unterstützung von Lookup-Tabellen.*

1.16: *Bereitstellung von Kontrollstrukturen für die Implementierung der Transformationen.*

4.2 Anforderungen an die Steuerungskomponente

4.2.1 Übertragung einzelner Nachrichten

Das Übertragen von Nachrichten über eine Kommunikationsverbindung setzt sich aus drei Schritten zusammen:

1. Dem Empfangen der Nachricht vom sendenden Subsystem,
2. dem Verarbeitungsprozeß der Nachricht im Kommunikationsserver (Nachrichtenidentifikation, Routen und Transformation) und
3. dem Versenden der Nachricht an die Empfänger.

Durch den Kommunikationsserver wird eine Kommunikationsverbindung zwischen zwei Subsystemen in zwei Teilverbindungen zerlegt: Die erste reicht vom sendenden Subsystem zum Kommunikationsserver und die zweite vom Kommunikationsserver zum empfangenden Subsystem. Da für beide Teilverbindungen eigene, möglicherweise verschiedene Transportprotokolle verwendet werden, ist eine synchrone Übertragung zwischen den Subsystemen auf der Ebene der Transportprotokolle so einfach nicht möglich. Unter einer *synchronen Übertragung* zwischen zwei Subsystemen soll dabei verstanden werden, daß der Empfänger den Sender von dem korrekten oder fehlgelaufenen Empfang einer Nachricht informiert und der Sender auf diese Acknowledgements wartet (vgl. auch Abschnitt 2.7). Auch der Empfänger kann bei der synchronen Übertragung auf bestimmte Nachrichten warten. Diese Bestätigung kann zwischen zwei Subsystemen nur in Form von Nachrichten erfolgen. Damit eine Bestätigung einer Nachricht eindeutig zugeordnet werden kann, ist es notwendig, daß der Kommunikationsserver nach dem Weiterleiten einer Nachricht den Empfang und das Zurücksenden der Bestätigung abwartet, also die Verbindung für andere Nachrichten blockiert, bis die synchrone Übertragung abgeschlossen ist.

Einige Nachrichten müssen besonders schnell übermittelt werden. So sollte zum Beispiel eine Blutwertanfrage aus dem OP vorrangig vor anderen Nachrichten behandelt werden. Nachrichten sollten also mit **Prioritäten** versehen werden können, die zum Beispiel die Reihenfolge der Behandlung der Nachrichten in einer Warteschlange (*Queue*) beeinflußt. Dabei ist sicherzustellen, daß Nachrichten mit einer sehr geringen Priorität nicht in den Queues „steckenbleiben", wenn sie stets von Nachrichten mit höherer Priorität „überholt" werden.

Anforderungen:

2.1: *Unterstützung verschiedener Nachrichtenlängen.*

2.2: *Unterstützung einer synchronisierten Übertragung von Nachrichten zwischen zwei Subsystemen.*

2.3: *Vergabe von Prioritäten an die Nachrichten.*

4.2.2 Verwaltung komplexer Aktionen

Neben der Übertragung einzelner Nachrichten können im Krankenhaus Kommunikationsprozesse auch aus mehreren, voneinander abhängigen Nachrichten bestehen. Diese Kommunikationsprozesse sollen hier als **komplexe Aktionen** bezeichnet werden. Dabei setzt sich eine komplexe Aktion aus mehreren, voneinander abhängigen Einzelnachrichten und mit diesen verbundenen Aktionen zusammen. Zusätzlich gehören zu einer komplexen Aktion Verfahrensregeln für den Kommunikationsserver, falls nicht alle Aktionen durchgeführt werden können. Für jede komplexe Aktion existiert ein Trigger, der aus einer Nachricht oder einem Ereignis (z.B. einem Alarm) bestehen kann.

Definition 4.1 (Komplexe Aktion)

Eine komplexe Aktion setzt sich aus einem Trigger, mehreren einzelnen Nachrichten, mit den Nachrichten verbundenen Aktionen und Verfahrensregeln für den Fall, daß nicht alle Teilaktionen durchgeführt werden können, zusammen.

Eine komplexe Aktion „*Neuaufnahme eines Patienten*“ könnte folgendermaßen aussehen:

> Das Subsystem für die zentrale Patientenaufnahme (ZPA-System) sendet eine Nachricht vom Typ „*Neuaufnahme*“ an den Kommunikationsserver. Dieser leitet die Nachricht an das PDV-System und das Stationssystem der Station, auf die der Patient aufgenommen werden soll, weiter. Zusätzlich erfolgt vom Kommunikationsserver eine Anforderung der (elektronischen) Krankenakte des Patienten im Zentralarchiv und vom Archiv eine Meldung an die jeweilige Krankenstation, ob eine Akte vorhanden ist bzw. die Bestätigung einer Aktenzustellung.

Die Nachricht vom Typ „*Neuaufnahme*“ dient hier als Trigger für die komplexe Aktion. Die komplexe Aktion selber setzt sich aus den Nachrichten an das PDV-System, die Krankenstation und das Zentralarchiv zusammen. Verbundene Aktionen sind das Weiterleiten der Aufnahme-Nachricht an das PDV-System und die Krankenstation, das Generieren und Übermitteln der Nachricht zum Anfordern der Patientenakte an das Zentralarchiv und das Erwarten und Weiterleiten der Meldung aus dem Zentralarchiv an die Krankenstation. Als Verfahrensregeln könnten hier gelten, daß der Kommunikationsserver nur eine bestimmte Zeit auf die Antwort aus dem Zentralarchiv wartet, dann die Antwort abmahnt oder eine entsprechende Meldung an die Krankenstation sendet.

Komplexe Aktionen können mit den unterschiedlichsten Teilaktionen verbunden sein. Eine einfache Aktion ist zum Beispiel das Kopieren und Versenden einer Nachricht an mehrere Empfänger (*Broadcasting* oder *Multicasting*). Aber auch der umgekehrte Weg ist bei komplexen Aktionen denkbar. So könnte eine komplexe Aktion daraus bestehen, daß von verschiedenen Subsystemen Nachrichten gesammelt, zusammengefügt und als eine Nachricht an ein weiteres Subsystem versandt werden. Für die Ausführung solcher komplexen Aktionen muß der Kommunikationsserver in der Lage sein, selbständig Nachrichten zu generieren, Nachrichten zu splitten oder mehrere Nachrichten zu einer zusammenzufügen.

Anforderungen:

- **2.4:** *Eigenständige Generierung von Nachrichten.*
- **2.5:** *Zusammenfügen einzelner Nachrichten von unterschiedlichen Sendern zu einer Nachricht.*
- **2.6:** *Unterstützung von komplexen Aktionen durch den Kommunikationsserver.*

4.2.3 Anforderungen an das Fehlermanagement und die Persistenzsicherung

Beim *Fehlermanagement* geht es um eine effektive Behandlung von Fehlern, die während des laufenden Betriebs des Kommunikationsservers auftreten können. Diese Fehler reichen von falschen Nachrichten bis hin zu einem Systemabsturz des Kommunikationsservers. Auf alle möglichen Fehler muß der Kommunikationsserver vorbereitet sein und adäquat reagieren können.

Nachrichten von einem Subsystem können fehlerhaft sein. Die Übertragungsprotokolle sichern eine nahezu fehlerfreie Übertragung der Nachrichten, wenn sie mit entsprechenden Sicherungsmechanismen ausgestattet sind (vgl. Abschnitt 4.1.1). Die Fehler können aber auch im Inhalt der Nachricht selber liegen, vom Subsystem also fehlerhaft erzeugt sein. Dabei lassen sich drei Fehlertypen unterscheiden, für die verschiedene Behandlungsstrategien denkbar sind.

1. Ein **Identifikationsfehler** tritt dann auf, wenn die Nachricht vom Kommunikationsserver keinem der im Kommunikationsprotokoll vereinbarten Nachrichtentypen bzw. keinem definierten Schlüsselwort (vgl. Abschnitt 4.1.2) zugeordnet werden kann. Eine solche Nachricht muß vom Kommunikationsserver zurückgewiesen und der Sender über den Sachverhalt informiert werden.

2. Eine Nachricht kann **nicht vollständig** sein. So können einige Datenfelder nicht belegt sein oder am Ende der Nachricht ganz fehlen. So zum Beispiel bei einer Stammdaten-Änderungsanzeige, in der nur die Datenfelder belegt sind, die sich im Stammdatensatz geändert haben. Andere Datenfelder oder Segmente können aber für Nachrichten zwingend erforderlich sein. So zum Beispiel die Angaben zur Identifizierung eines Patienten bei einem Behandlungsauftrag. Um hier unnötige Nachrichtenlaufzeiten und Belastung anderer Subsysteme zu vermeiden, sollte der Kommunikationsserver in der Lage sein, Nachrichten auf zwingende Datenfelder hin zu überprüfen und die Nachricht gegebenenfalls mit einem entsprechenden Hinweis zurückweisen zu können.

   Einige standardisierte Kommunikationsprotokolle wie zum Beispiel HL7 sehen zwingende Segmente in Nachrichten vor. Die Überprüfung der Nachricht sollte dann automatisch vom Kommunikationsserver vorgenommen werden können. Bei proprietären Kommunikationsprotokollen sollten zwingende Datenfelder bei der Definition der Nachrichtenstrukturen unterstützt werden. Auch hier kann dann eine automatische Kontrolle durch den Kommunikationsserver erfolgen.

3. Ein **Datentyp-Fehler** tritt dann auf, wenn ein Datenfeld in der Nachricht nicht dem in der Datenfelddefinition vereinbartem Datentyp entspricht. Zum Beispiel ein Text in einem numerischen Datenfeld. Wird dieses Datenfeld während der Strukturtransformation konvertiert, kann es zu einem Fehler kommen. Auch wenn der Typfehler den Transformationsprozeß nicht beeinträchtigen würde, sollte der Kommunikationsserver solche Fehler erkennen und die Nachricht zurückweisen können.

Grundsätzlich sollte der Kommunikationsserver das sendende Subsystem über den Fehler in der Nachricht informieren können. Fehlerhafte Nachrichten sollten dabei komplett oder aber zumindest eine entsprechende Mitteilung an das sendende Subsystem zurückgeschickt werden. Werden die Fehler lediglich in den Log-Dateien vermerkt und die Nachricht nur in einer Fehler-Datenbank abgelegt, lassen sich solche Fehler nur schwer aufdecken. Außerdem geht das sendende Subsystem von einer korrekten Übertragung der Nachricht aus, wenn es keine entsprechende Mitteilung vom Kommunikationsserver erhält. Hier ist allerdings anzumerken, daß keines der im Rahmen dieser Arbeit untersuchten Subsysteme mit einer solchen Fehlermeldung etwas anfangen könnte. Eine entsprechende Funktionalität müßte erst in den Kommunikationsklienten implementiert werden (siehe Abschnitt 3.3).

Fehlerhafte Nachrichten sollten nicht nur an den Sender zurückgesandt, sondern zusätzlich mit der genauen Angabe des Fehlers bzw. der Ursache in einer Fehler-Datenbank des Kommuniaktionsservers gespeichert werden. Zusammen mit den Log-Dateien (siehe Abschnitt 4.3.1) kann so ein Nachweis über den Fehler vom Administrator des Kommunikationsservers erbracht werden. Die Nachrichten sollten in der Fehler-Datenbank nur in verschlüsselter Form vorliegen. Wie bei der Speicherung von korrekten Nachrichten sollte auch hier der Administrator durch ein Modul für die Verwaltung der Fehler-Datenbank, wie das Einsehen und erneute Versenden der Nachrichten, unterstützt werden.

Kann eine Nachricht nicht an ein Subsystem übermittelt werden, da die Verbindung unterbrochen oder das Subsystem heruntergefahren ist, darf die Nachricht dennoch nicht verloren gehen. Grundsätzlich sollte ein Kommunikationsserver beim Versenden von Nachrichten nach dem Prinzip *„store-and-forward“* vorgehen. Dieses Prinzip beinhaltet, daß die Nachrichten für die Subsysteme in einer Warteschlange (*Queue*) gehalten werden, die die Reihenfolge der Nachrichten erhält (*FIFO-Prinzip: First in, first out*). Eine Nachricht wird erst dann aus der Warteschlange gelöscht, wenn sie erfolgreich an den Empfänger übermittelt werden konnte. Dabei sollte für jedes Subsystem getrennt eine eigene Queue verwaltet werden.

Queues bieten sich auch für den Empfang von Nachrichten in den Subsystem-Agenten an. Bildet sich ein Stau bei der Nachrichtenverarbeitung, können so dennoch Nachrichten angenommen und zwischengespeichert werden.

Die Queues sollten vom Kommunikationsserver auf externen Speichermedien (z.B.: Festplatte) gehalten werden. Nur so kann ihr Inhalt einen Stromausfall überleben. Da die Queues die kompletten Nachrichten enthalten, sind auch hier Verschlüsselungsmethoden notwendig.

Ist ein Subsystem für längere Zeit ausgefallen, können Nachrichten in der Queue „gefangen" sein. Der Kommunikationsserver sollte bei der Verwaltung der Queues eine maximale Verweildauer von Nachrichten berücksichtigen können. Konnte eine Nachricht innerhalb dieser Zeit nicht übermittelt werden, so wird sie aus der Queue entfernt und zum Beispiel an den Sender mit einem entsprechenden Kommentar zurückgeschickt. Die Verweildauer sollte dabei für jede Nachricht getrennt parametrisierbar sein. Ein solches Verfahren ist vor allem bei interaktiven Anfragen erforderlich. Hier wartet der Anwender eines Subsystems auf die Antwort seiner Anfrage. Kann die mit der Anfrage verbundene Nachricht nicht weitergeleitet werden, so muß der Anwender durch eine entsprechende Nachricht von diesem Sachverhalt informiert werden können.

Alternativ können für Subsysteme mehr als eine Kommunikationsverbindung vorgesehen werden. Ist zum Beispiel ein Stationssystem über das Netz nicht erreichbar, so kann ein wichtiger Laborbefund auch per Fax an die Station gesandt werden. Die Auswahlmöglichkeit alternativer Verbindungen zu Subsystemen sollten von einem Kommunikationsserver unterstützt werden.

Grundsätzlich gilt, daß Nachrichten während des Bearbeitungsprozesses im Kommunikationsserver nicht verlorengehen dürfen. Dazu ist es notwendig, daß in einer Recover-Datenbank kontinuierlich der Bearbeitungsstand einer Nachricht festgehalten wird. Nach einem Systemabsturz kann dann der Kommunikationsserver bei jeder Nachricht wieder an dem Punkt beginnen, der zuletzt erfolgreich durchgeführt werden konnte. Auch andere Recover-Mechanismen sind denkbar.

Anforderungen:

3.1: *Effektive Behandlung von fehlerhaften Nachrichten.*

3.2: *Speicherung von fehlerhaften Nachrichten in einer Fehler-Datenbank.*

3.3: *Bereitstellung von Queues zur persistenten Haltung von Nachrichten.*

3.4: *Unterstützung von alternativen Kommunikationsverbindungen zu Subsystemen.*

3.5: *Bereitstellung von Recover-Mechanismen für den Fall eines Systemabsturzes.*

3.6: *Unterstützung von Maßnahmen zur Ausfallsicherheit des Kommunikationsservers.*

4.3 Anforderungen an die Überwachungskomponente

4.3.1 Logging und Nachrichtenarchivierung

Nicht nur in der Testphase einer Kommunikationsverbindung ist ein ausführliches Logging aller Vorgänge im Kommunikationsserver wichtig. Anhand einer Log-Datei lassen sich Fehler zum Beispiel im Nachrichten-Routing aufgrund einer falschen Nachrichtenidentifikation nachvollziehen. Auch läßt sich nachweisen, daß eine Nachricht von einem Subsystem den Kommunikationsserver nicht erreicht hat oder aber erfolgreich bei einem Subsystem abgeliefert wurde.

Der Inhalt einer Log-Datei sollte leicht lesbar und verständlich sein. Ein Protokoll, das nur aus Abkürzungen besteht und sich nur mit Hilfe einer Betriebsanleitung lesen läßt, stellt keine geeignete Hilfe dar.

Der Umfang des Loggings ist sehr von der Phase, in der sich eine Kommunikationsverbindung befindet, abhängig. In der Testphase wird ein wesentlich ausführlicheres Protokoll benötigt als für Kommunikationsverbindungen, die sich schon längere Zeit ohne Beanstandungen im Betrieb befinden. Der Umfang der Log-Dateien sollte also konfigurierbar sein.

In einem KKS sind sehr viele Subsysteme an den Kommunikationsserver angeschlossen. Um hier in den Log-Dateien nicht die Übersicht zu verlieren, sollten für jedes Subsystem oder für jede Kommunikationsverbindung getrennte Log-Dateien verwendet werden können. Diese lassen sich bei einer Fehlersuche wesentlich leichter auswerten als ein Gesamtprotokoll für alle Verbindungen.

Vor allem in der Testphase einer Kommunikationsverbindung können die Log-Dateien sehr lang werden. Einen Fehler, der einige Tage zurückliegt, in einem kontinuierlichen Protokoll über mehrere Tage oder Wochen nachzuvollziehen, ist nur mit einem großen Zeitaufwand möglich. Der Kommunikationsserver sollte eine automatische Verwaltung der Log-Dateien anbieten. Zum Beispiel könnten zu einem definierten Zeitpunkt alle Log-Dateien geschlossen, in einem Archivsystem abgelegt und neue Log-Dateien eröffnet werden. So lassen sich zum Beispiel abgeschlossene Protokolle jeweils eines Tages erzeugen. Fehler, die an einem bestimmten Tag aufgetreten sind, lassen sich so leichter rekonstruieren.

Ein spezielles Auswertungsmodul für die Aufarbeitung der Log-Dateien kann die Fehlersuche erleichtern. Neben einer farblichen Hervorhebung der einzelnen Eintragstypen (z.B.: Warnungen, Fehler oder normale Informationen) wäre auch eine mehr strukturierte Darstellung des Nachrichtenbearbeitungsprozesses, getrennt für einzelne Nachrichten, hilfreich. So könnte zum Beispiel der Bearbeitungsprozeß einer Nachricht mit den einzelnen Bearbeitungsschritten (Empfangen, Identifizieren, Routen, Transformieren und Senden) graphisch dargestellt werden. Zu jedem Bearbeitungsschritt sind dann die Logging-Informationen abrufbar. Fehler im Bearbeitungsprozeß werden farblich hervorgehoben. Die Log-Dateien sollen aber auch ohne spezielle Editoren oder Viewer einsehbar bleiben.

Neben dem Logging des Nachrichtenverarbeitungsprozesses im Kommunikationsserver sollten auch Nachrichteninhalte gespeichert werden können (*Nachrichtenarchivierung*). Nur so lassen sich Fehler nachvollziehen, wenn die betroffene Nachricht bereits zugestellt und gelöscht wurde.

Anforderungen:

- **4.1:** *Konfigurierbares Logging für alle Schritte des Bearbeitungsprozesses einer Nachricht.*
- **4.2:** *Automatische Verwaltung der Log-Dateien durch den Kommunikationsserver.*
- **4.3:** *Bereitstellung eines Moduls für die Auswertung der Log-Dateien.*
- **4.4:** *Möglichkeit zur Speicherung von Nachrichteninhalten.*
- **4.5:** *Bereitstellung eines Moduls für die Verwaltung der gespeicherten Nachrichten.*

4.3.2 System-Monitor

In Abschnitt 3.2 wurde bereits erwähnt, daß das Modul, das die Funktionen der Monitorkomponente realisiert, als *System-Monitor* bezeichnet wird. Dieser System-Monitor stellt ein wichtiges Werkzeug für die Überwachung und Steuerung des Kommunikationsservers dar. Durch ihn sollten die einzelnen Kommunikationsverbindungen in einer übersichtlichen, graphischen Form präsentiert werden. Anders als bei der Darstellung des Nachrichten-Routing (siehe Abschnitt 4.1.3) sollte hier der Kommunikationsserver als Bestandteil der Kommunikationsverbindung mit berücksichtigt werden. Die Darstellung sollte konfigurierbar sein, so daß zum Beispiel nur bestimmte Kommunikationsverbindungen abgebildet werden können. Neben der Abbildung der Kommunikationsverbindungen ist auch eine Darstellung der Betriebszustände der Verbindungen, des Kommunikationsservers und der Subsysteme notwendig. So sollte auf einen Blick erkennbar sein, welche Subsysteme zur Zeit nicht erreichbar, welche Kommunikationsverbindungen aktiviert oder deaktiviert sind oder wo sich Nachrichten gerade in einer Warteschlange befinden.

Neben der bloßen Darstellung des Kommunikationsservers und der Subsysteme sollte ein System-Monitor auch Funktionen zur Steuerung der einzelnen Module anbieten. Verbindungen zu Subsystemen müssen aktiviert oder deaktiviert werden, neue Konfigurationen müssen aktiviert werden. Diese Funktionen gehören eigentlich in die Serverkomponente. Sie werden aber hier besprochen, da sie in der Regel vom System-Monitor aus aufrufbar sind.

Die Steuerung des Kommunikationsservers sollte auch ohne den System-Monitor möglich sein, sich dann aber nicht nur auf Betriebssystembefehle beschränken. Eine Version des System-Monitors, die ohne eine graphische Oberfläche auskommt, aber die gleiche Funktionalität aufweist, wäre hier wünschenswert. So könnte der Kommunikationsserver von jeder Stelle des klinikweiten Netzes aus kontrolliert werden. Der Inhalt von Nachrichten sollte aber aus Gründen des Datenschutzes auf diese Weise nicht abgerufen werden können!

Vom System-Monitor aus sollten auch die Log-Dateien und die gespeicherten Nachrichten einsehbar sein. Log-Dateien und Nachrichtenarchive sind für eine Fehleranalyse bei zum Beispiel dem Ausfall einer Kommunikationsverbindung notwendig.

Eine weitere wichtige Funktion des System-Monitors ist die Erzeugung von Statistiken. Zum Beispiel helfen Statistiken über die Auslastung einzelner Kommunikationsverbindungen diese optimaler zu konfigurieren. Die folgenden Parameter könnten für den Administrator von Interesse sein:

Nachrichtendurchsatz: Der Nachrichtendurchsatz ist eine wichtige Kenngröße für die Auslastung des Kommunikationsservers. Hierbei ist nicht nur der Durchsatz insgesamt, sondern auch der für einzelne Kommunikationsverbindungen oder gar für spezielle Nachrichtentypen von Interesse. Das Nachrichtenaufkommen unterliegt bei vielen Subsystemen einer tageszeitlichen Schwankung. Neuaufnahmen erfolgen in der Regel vormittags, während Untersuchungsergebnisse eher in der zweiten Tageshälfte anfallen. Mit Hilfe einer kontinuierlichen Auswertung des Nachrichtendurchsatzes lassen sich Spitzenzeiten und möglicherweise überlastete Verbindungen erkennen.

Länge der Queues: Die Länge der Queues sind ebenfalls ein Maß für die Belastung einzelner Verbindungen. Kommunikationsverbindungen mit ständig gefüllten Queues zeigen eine Überlastung der Verbindung oder des an diese Verbindung angeschlossenen Subsystems an. Hier bietet sich möglicherweise die Einrichtung einer weiteren Kommunikationsverbindung als Entlastung an. Überlange Queues können aber auch beim Ausfall einer Kommunikationsverbindung oder beim Ausfall eines Subsystems auftreten.

Speicherplatz: Der verfügbare Speicherplatz auf der Festplatte ist vor allem bei der Verwendung dateibasierter Transportprotokolle interessant. Ist auf der Platte kein Platz mehr, so können unter Umständen keine weiteren Nachrichten übertragen werden. Schreibt ein Subsystem über ein automatisiertes FTP Dateien auf die Festplatte des Kommunikationsservers und werden Speicherplatzprobleme nicht abgefangen, so kann es zu einem Nachrichtenverlust kommen. Auch hier kann der zeitliche Verlauf der Speicherplatzauslastung von Interesse sein. Werden zum Beispiuel zu bestimmten Zeitpunkten große Datenmengen dateibasiert übertragen, so sollten bei einer hohen Speicherplatzauslastung weitere dateibasierte Nachrichtenübertragungen möglichst auf andere Zeitpunkte verlegt werden. Speicherplatzprobleme können auch bei einer sehr detaillierten Überwachung einzelner Kommunikationsverbindungen durch die stark anwachsenden Log-Dateien entstehen.

CPU-Auslastung: Diese Kenngröße ist ein Maß für die Leistungsfähigkeit und Belastung der für den Kommunikationsserver eingesetzten Hardware. Für die Erkennung von Belastungsspitzen ist hier eine zeitkontinuierliche Darstellung der Auslastung besonders wichtig. Bei einer häufig hohen Auslastung sollte über den Einsatz eines leistungsstärkeren Rechners oder – wenn möglich – über die Verteilung einzelner Kommunikationsverbindungen auf mehrere Rechner nachgedacht werden.

Die Liste dieser Beispiele ist bestimmt nicht vollständig, soll aber an dieser Stelle genügen. Die Beispiele zeigen, daß es wichtig ist, daß die Statistiken möglichst nicht nur für den aktuellen Zeitpunkt, sondern auch für längere Zeiträume und für die Kommunikationsverbindungen getrennt erstellt werden können. Für retrospektive Auswertungen ist es erforderlich, daß die Statistiken kontinuierlich erstellt und gespeichert werden können. Ähnlich wie bei der Verwaltung der Log-Dateien (siehe Abschnitt 4.3.1) wäre eine automatische Verwaltung der Dateien, die die Statistiken enthalten, durch den Kommunikationsserver wünschenswert. Neben dem Aufstellen der Statistiken sind auch eine konfigurierbare Darstellung in unterschiedlichen Darstellungsformen (zum Beispiel Tabellen oder Graphiken) und ein Modul, das eine Auswertung ermöglicht, hilfreich.

Anforderungen:

4.6: *Darstellung der Kommunikationsverbindungen im System-Monitor.*

4.7: *Bereitstellung von Funktionen zur getrennten Steuerung der Module und Verbindungen des Kommunikationsservers.*

4.8: *Aufstellung und Verwaltung von Statistiken über verschiedene Parameter.*

4.3.3 Alarmfunktionen

Der Kommunikationsserver und seine Verbindungen können nicht ständig vom Administrator über den System-Monitor überwacht werden. Bestimmte Situationen, wie der Ausfall einer wichtigen Kommunikationsverbindung, sollen aber nicht erst von den Subsystemen durch das Ausbleiben von Nachrichten erkannt werden. Daher sollte ein Kommunikationsserver konfigurierbare *Alarmfunktionen* aufweisen. Dabei können bestimmte Situationen wie zum Beispiel der Überlauf einer Queue im Kommunikationsserver als Auslöser für einen Alarm definiert werden. Tritt diese Situation ein, wird der Alarm über die Alarmfunktion ausgelöst. Neben einer optischen oder akustischen Darstellung im System-Monitor ist auch die direkte Benachrichtigung des Administrators denkbar. Diese kann durch das Versenden von *eMail*, das Anfunken eines Piepers, das Faxen einer Meldung oder ähnlichem geschehen. Auch das Aufrufen von externen Prozessen oder Funktionen sollte hier unterstützt werden.

Der Umfang der Alarmfunktionen sollte sich über alle Bereiche des Kommunikationsservers, über die Hardware und das Netzwerk erstrecken. Dabei sollten sie für die einzelnen Kommunikationsverbindungen getrennt konfigurierbar sein.

Anforderungen:

4.9: *Bereitstellung von konfigurierbaren Alarmfunktionen.*

4.4 Weitere Anforderungen an die Konfigurationskomponente

In diesem Abschnitt werden Anforderungen an weitere Funktionen aus der Konfigurationskomponente aufgestellt. Die Anforderungen an die Konfigurierung der Funktionen aus den anderen Komponenten werden jeweils bei der Beschreibung der Funktionen in den entsprechenden Abschnitten dargestellt.

4.4.1 Definition der Nachrichtenstrukturen

Die Kommunikationsprotokolle legen die Darstellungsformen der Nachrichten auf Applikationsebene fest (7. Ebene des ISO/OSI-Modells). Der Kommunikationsserver muß an mehreren Stellen im Verarbeitungsprozeß einer empfangenen Nachricht auf den Inhalt einzelner Datenfelder zugreifen. So zum

Beispiel bei der Nachrichtenidentifikation und bei der Transformation der Nachricht in eine andere Nachrichtenstruktur. Hier ist es hilfreich, wenn die Strukturen der Nachrichten im Kommunikationsserver definiert werden können. Ein Zugriff auf die Datenfelder einer Nachricht kann dann über die in der Struktur definierten logischen Bezeichner erfolgen. Ändert sich einmal die Struktur einer Nachricht, so muß lediglich die Strukturdefinition im Kommunikationsserver angepaßt werden. Müssen dagegen in allen von der Strukturänderung betroffenen Identifikations- und Transformationsregeln die Änderungen vorgenommen werden, können bei der Vielzahl der Regeln leicht Fehler entstehen bzw. notwendige Anpassungen übersehen werden.

Die meisten Subsysteme in einem Krankenhaus verwenden proprietäre Kommunikationsprotokolle. Standardisierte Kommunikationsprotokolle wie zum Beispiel HL7 oder Edifact werden bisher nur wenig eingesetzt, werden aber von immer mehr Subsystemen unterstützt. Diese Standards müssen in der Regel an die Gegebenheiten im Krankenhaus angepaßt werden. So zum Beispiel das Hinzufügen von Z-Segmenten in HL7 für spezielle, nicht durch den Standard abgedeckten Nachrichten. Der Kommunikationsserver sollte alle im Krankenhausumfeld verwendeten standardisierten Kommunikationsprotokolle unterstützen. Wie die proprietären Kommunikationsprotokolle sollten auch die Standards im Kommunikationsserver selber definiert sein und dort vom Anwender angepaßt werden können.

Anforderungen:

5.1: *Definition der Nachrichtenstrukturen proprietärer Kommunikationsprotokolle im Kommunikationsserver.*

(S) **5.2:** *Hinterlegung der Nachrichtenstrukturen standardisierter Kommunikationsprotokolle im Kommunikationsserver.*

4.4.2 Änderung der Konfigurationen

An der Konfiguration eines Kommunikationsservers werden immer wieder Änderungen notwendig sein. Neue Subsysteme kommen hinzu, bereits verbundene ändern ihr Kommunikationsprotokoll oder neue Nachrichten werden versandt. Dafür ist es wichtig, daß Änderungen an der Konfiguration eines Kommunikationsservers auch im laufenden Betrieb möglich sind. Dabei sollte es möglich sein, die Änderungen auch ohne ein vollständiges „herunterfahren“ (*shutdown*) des Kommunikationsservers in den Betrieb zu übernehmen. Es sollten nur die Komponenten des Kommunikationsservers beeinträchtigt werden, die von der Änderung betroffen sind.

Anforderungen:

5.3: *Änderungen in der Konfiguration der Kommunikationsverbindungen müssen im laufenden Betrieb möglich sein.*

Dokumentation der Kommunikationsverbindungen

Neben der Darstellung der Konfigurationen der Kommunikationsverbindungen im Kommunikationsserver (z.B. Darstellung der Routing-Tabellen) sollte ein Kommunikationsserver auch eine externe Dokumentation der Kommunikationsverbindungen unterstützten. In einer solchen Dokumentation sollte jede Kommunikationsverbindung mit den eingesetzten Kommunikationsklienten und Subsystem-Agenten, den verwendeten Transport- und Kommunikationsprotokollen sowie allen Schritten der Nachrichtenverarbeitung dargestellt werden. Alle Informationen liegen im Kommunikationsserver in Form der Konfigurationsdateien oder -tabellen vor, so daß eine automatische Generierung der Dokumentation durch die Konfigurationskomponente erfolgen kann.

Anforderungen:

5.4: *Automatisierte, konfigurierbare Dokumentation der Kommunikationsverbindungen durch den Kommunikationsserver.*

4.5 Anforderungen an die Testkomponente

Neue Konfigurationen müssen, bevor sie in die Produktion gehen, ausführlich getestet werden. Ein Kommunikationsserver stellt dafür in der Regel ein Testmodul zur Verfügung. Mit dessen Hilfe sollte der Anwender alle Konfigurationen einzeln testen können. So läßt sich die Definition der Nachrichtenstrukturen, die Nachrichtenidentifikation, die Ermittlung der Empfänger und die Transformationen mit Hilfe von Testnachrichten überprüfen.

Sind die Konfigurationen mit Hilfe des Testmoduls evaluiert, soll eine neue Kommunikationsverbindung in der Regel nicht sofort in den endgültigen Betrieb gehen, sondern erst weiter mit Nachrichten aus den Subsystemen getestet werden. Da in dieser Phase einer Kommunikationsverbindung andere Anforderungen an den Kommunikationsserver gestellt werden als in der Produktionsphase (z.B.: ausführlicheres Protokollieren der Verbindung), sollte ein Kommunikationsserver spezielle Betriebsmodi zum Testen einzelner Kommunikationsverbindungen oder gar einzelner Nachrichtentypen zur Verfügung stellen. So lassen sich Kommunikationsverbindungen, die sich in der Testphase befinden, sauberer von den anderen Verbindungen trennen.

Anforderungen:

6.1: *Bereitstellung eines Moduls zum Testen der Konfigurationen des Kommunikationsservers.*

6.2: *Unterstützung verschiedener Betriebsmodi für einzelne Kommunikationsverbindungen.*

4.6 Anforderungen an den Datenschutz

4.6.1 Nachrichtenverschlüsselung

Da der Inhalt der Nachrichten im Krankenhaus meistens aus personenbezogenen Daten eines Patienten bestehen, sollten sie zwischen den Subsystemen nur in verschlüsselter Form ausgetauscht werden. Dazu ist es notwendig, die Nachrichten innerhalb des Kommunikationsservers zu entschlüsseln, damit eine Nachrichtentransformation vorgenommen werden kann. Vor dem Weiterleiten der Nachricht muß diese dann wieder verschlüsselt werden. Dabei kann nun auch eine andere Verschlüsselungsmethode verwendet werden. Unterschiedliche Verschlüsselungsmethoden für die einzelnen Subsysteme bieten einen besseren Schutz der Daten vor unbefugtem Zugriff als eine für alle Subsysteme einheitliche Methode. Der Kommunikationsserver dient dann auch auf dieser Ebene als Protokoll-Transformator.

Auch die Nachrichtenarchive und Log-Dateien sollten verschlüsselbar sein. Vor allem dann, wenn in den Log-Dateien auch Nachrichteninhalte gespeichert werden.

Anforderungen:

7.1: *Unterstützung von Verschlüsselungsmethoden für die Nachrichtenübertragung.*

7.2: *Möglichkeit zur Verschlüsselung der gespeicherten Nachrichten.*

7.3: *Möglichkeit zur Verschlüsselung der Log-Dateien.*

4.6.2 Zugriffsschutz

Der System-Monitor als „Steuerzentrale" des Kommunikationsservers sollte besonders gegen einen unbefugten Zugriff geschützt werden können. Eine Zugangskontrolle mit einer eigenen Benutzer- und Paßwortverwaltung wäre hier angebracht. Dies gilt nicht nur für den System-Monitor. Auch die anderen Module des Kommunikationsservers sollten einem Zugriffsschutz unterliegen.

Eine andere Möglichkeit wäre die Definition von **Benutzerrollen**. Jeder Benutzerrolle werden dabei unterschiedliche Befugnisse zugeteilt. So könnte einem Benutzer des Kommunikationsservers eine Benutzerrolle zugeteilt werden, die ihm zum Beispiel lediglich das Einsehen aber nicht das Ändern der Konfigurationstabellen erlaubt. Nur der Administrator besitzt alle Rechte.

Anforderungen:

7.4: *Vom Betriebssystem unabhängiger Zugriffsschutz zum System-Monitor und den weiteren Modulen des Kommunikationsservers.*

7.5: *Unterstützung von unterschiedlichen Benutzerrollen zur Steuerung der Rechte der Benutzer.*

4.7 Verwaltung einer Datenbank durch den Kommunikationsserver

Der Kommunikationsserver ist ein zentrales Element in einem KKS. Alle Subsysteme sind direkt mit ihm verbunden und alle Nachrichten, die von den Subsystemen erzeugt werden, laufen bei ihm zusammen. Der Kommunikationsserver bietet sich daher auch für administrative Aufgaben, wie zum Beispiel die Verwaltung einer zentralen Patientendatenbank oder eines zentralen Befundservers an. Soll diese Datenbank nicht ein weiteres eigenständiges Subsystem darstellen, ist es notwendig das DBMS der Datenbank mit dem Kommunikationsserver zu verbinden. Dies kann bei einer relationalen Datenbank zum Beispiel über eine SQL-Schnittstelle erfolgen. Der Kommunikationsserver kann dann über diese Schnittstelle die Datenbank direkt verwalten. Am Beispiel der Verwaltung einer Patientendatenbank sollen hier Anforderungen an einen Kommunikationsserver formuliert werden.

Alle Nachrichten, die die Daten in der Datenbank betreffen, müssen vom Kommunikationsserver in entsprechende Datenbankoperationen umformuliert werden. Dies können zum Beispiel Änderungen der Stammdaten eines Patienten, die Neuaufnahme eines Stammdatensatzes oder die Abfrage von Daten sein. Wird für die Datenbank eine standardisierte Schnittstelle wie zum Beispiel SQL verwendet, kann die Generierung der Datenbankoperationen auch automatisiert erfolgen.

Die Verwaltung der Datenbank soll dabei für die Subsysteme transparent bleiben. Das bedeutet, daß das Subsystem nicht „wissen" muß, ob die Datenbank durch den Kommunikationsserver verwaltet wird oder ein eigenständiges Subsystem darstellt.

Neben der Abfrage und dem Ändern des Datenbankinhaltes gehören zur Verwaltung einer Datenbank auch die Definition und Pflege der Datenbankschemata. So zum Beispiel das Neuanlegen oder

Ändern von Tabellen in einer relationalen Datenbank. Die Kenntnis der Datenbankstruktur ist auch für die Definition von komplexen Abfragen unbedingt erforderlich. Die Datenbankschemata müssen also im Kommunikatonsserver abgebildet und von dort aus definiert und geändert werden können. Unabhängig von Nachrichten muß auch der Betreiber des Kommunikationsservers die Möglichkeit haben, auf die Daten in der Datenbank zuzugreifen. Ein eigenes Modul des Kommunikationsservers für die Verwaltung der Datenbank ist somit erforderlich.

Anforderungen:

8.1: *Anschluß eines DBMS an den Kommunikationsserver.*

8.2: *Generierung von Datenbankoperationen aus Nachrichten von Subsystemen.*

8.3: *Bereitstellung eines Moduls zur Verwaltung der externen Datenbank.*

Kapitel 5

Eine Taxonomie für Kommunikationsserver

In diesem Kapitel soll nochmals auf die in Abschnitt 3.2 vorgestellten Komponenten eines Kommunikationsservers eingegangen werden. Untersucht werden soll, wie die Funktionen der Komponenten auf Programm-Module verteilt werden können und welche System-Architekturen dabei für die Kommunikationsserver entstehen. Dabei soll hier nur die Serverkomponente (Komponente K1 im Abschnitt 3.2) betrachtet werden. Sie stellt die grundlegenden Funktionen eines Kommunikationsservers bereit, die Funktionen für den Nachrichtenempfang, die Nachrichtenverarbeitung und -weiterleitung an die Subsysteme. Die Funktionen dieser Komponente und ihrer Teilkomponenten lassen sich auf sehr unterschiedliche Weise auf Programm-Module verteilen. Die dabei entstehenden Module beeinflussen als wichtigste Bestandteile entscheidend die Architektur eines Kommunikationsservers. Von der Architektur wiederum hängen die Performanz und die Verfügbarkeit des Kommunikationsservers ab. So dürfte ein Kommunikationsserver, der alle Funktionen der Serverkomponente in nur einem Modul vereinigt, kaum einen akzeptablen Nachrichtendurchsatz erreichen. Viele verschiedene Verarbeitungswege ermöglichen eine nebenläufige Verarbeitung der Nachrichten, die sich positiv auf die Performanz auswirken kann. Ein Kommunikationsserver, der für jedes Subsystem eigene, verteilbare Module mit den Funktionen der Verbindungs- und Verarbeitungskomponente bereitstellt, wird ein sehr hohes Maß an Verfügbarkeit bieten, Konsequenz ist aber ein hoher administrativer Aufwand, der sich negativ auf die Performanz auswirken kann.

Die Architektur, die sich aus der Aufteilung der Funktionen der Serverkomponente auf Programm-Module ergibt, liefert die Grundlage der Taxonomie für Kommunikationsserver, die im folgenden dargestellt werden soll.

5.1 Dimensionen der Taxonomie

In [ÖV91] wird eine Taxonomie entlang der Dimensionen *Verteiltheit*, *Heterogenität* und *Autonomie* für verteilte Datenbanksysteme vorgestellt. In Anlehnung daran unterteilt die Taxonomie die Kommunikationsserver entlang den folgenden Dimensionen:

1. Der *Verteiltheit* der Funktionen des Servermoduls,
2. der *Nebenläufigkeit* der Nachrichtenverarbeitung, sowie
3. der *Heterogenität* der Module des Kommunikationsservers.

Die drei Dimensionen der Taxonomie sollen im folgenden näher spezifiziert werden.

5.1.1 Verteiltheit des Servermoduls

Die Motivation für diese Dimension wurde bereits in der Einleitung zu diesem Kapitel dargestellt: Die Aufteilung der Funktionen des Servermoduls auf mehrere Programme ist eng mit der Performanz und der Verfügbarkeit des Kommunikationsservers verbunden.

Die Funktionen der Verbindungskomponente werden von allen in dieser Diplomarbeit betrachteten Kommunikationsserver auf verschiedene Module oder Programme verteilt. Bei der Besprechung der Funktionen der Verbindungskomponente im Abschnitt 3.2 wurde bereits erwähnt, daß dabei für jedes Subsystem eigene Programme bereitgestellt werden, die die Übertragung der Nachrichten zu diesen Subsystemen realisieren. Diese Programme werden auch als *Subsystem-Agenten* bezeichnet.

Auch die Funktionen der Verarbeitungskomponente lassen sich auf verschiedene Module aufteilen. Diesen Aspekt berücksichtigt die zweite Dimension der Taxonomie (siehe Abschnitt 5.1.2).

Die bei der Aufteilung der Module entstehenden Programme können nun auch im klassischen Sinne verteilt werden, also auf verschiedenen Rechnern ausgeführt werden. Somit läßt sich eine parallele Ausführung der Funktionen des Kommunikationsservers erreichen, die sich positiv auf die Performanz auswirken kann. Auch die Verfügbarkeit des Kommunikationsservers erhöht sich, wenn Kommunikationsverbindungen parallel betrieben werden können.

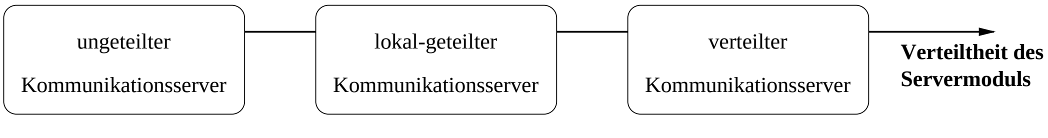
Bei der Verteiltheit der Funktionen der Serverkomponente sollen die folgenden Ausprägungen unterschieden werden:

1. **ungeteilt:**
   Bei dieser Ausprägung werden alle Funktionen des Servermoduls von einem Programm realisiert. Die Performanz eines solchen Kommunikationsservers dürfte keine akzeptablen Werte erreichen, vor allem dann, wenn, wie in einem KKS, viele Subsysteme miteinander verbunden werden müssen. Da die Kommunikationsserver in einem großen KKS für die Kommunikation eine entscheidende Rolle spielen, werden für sie normalerweise Rechner mit einer entsprechend leistungsfähigen Hardware und einem Betriebssystem eingesetzt, das Multitasking zuläßt. Nur in sehr kleinen Rechnernetzen mit einigen wenigen DV-Systemen und einem nur geringen Kommunikationsaufkommen, wäre eine solche ungeteilte Architektur eines Kommunikationsservers denkbar. Doch ist es fraglich, inwieweit sich hier der Einsatz eines Kommunikationsservers mit den damit verbundenen administrativen und finanziellen Aufwand noch rechtfertigen läßt.

2. **lokal-geteilt:**
   Hier werden die Funktionen des Servermoduls auf verschiedene Programme aufgeteilt, aber alle Programme müssen auf einem Rechner lokalisiert sein. Dies ist zum Beispiel notwendig, wenn für die Programme *Threads* anstelle von vollwertigen Prozessen verwendet werden.

3. **verteilt:**
   Bei dieser Ausprägung lassen sich die Programme, die die Funktionen des Servermoduls enthalten, im Netzwerk verteilen. So können zum Beispiel die Subsystem-Agenten auf die Rechner der Subsysteme ausgelagert werden. Eine Verteilung der Programme eines Kommunikationsservers setzt eine Interprozeßkommunikation voraus, die über Rechnergrenzen erfolgen kann.

Aus den drei dargestellten Ausprägungen lassen sich nun die folgenden drei Grade für diese erste Dimension der Taxonomie ableiten:



5.1.2 Nebenläufigkeit der Nachrichtenverarbeitung

Lassen sich die Funktionen des Servermoduls so auf Prozesse verteilen, daß mehrere getrennte Verarbeitungswege für Nachrichten entstehen, so kann eine *nebenläufige* Nachrichtenverarbeitung erreicht werden. Dabei soll erst dann von einer nebenläufigen Verarbeitung gesprochen werden, wenn mindestens zwei getrennte, vollständige Verarbeitungswege mit Nachrichtenidentifikation, Routing und Transformation für die Nachrichten existieren. Ist nur ein Verarbeitungsweg für alle Nachrichten vorgesehen, so liegt eine *serielle* Nachrichtenverarbeitung vor.

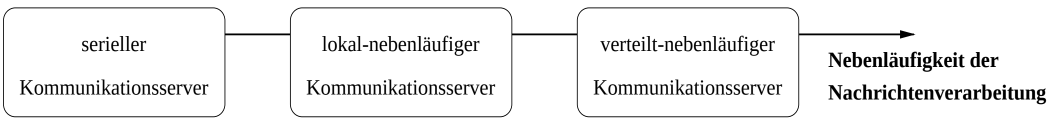
Analog zum Kriterium der Geteiltheit werden hier drei Ausprägungen der Nebenläufigkeit unterschieden:

1. **seriell:**
   Für alle Nachrichten exisitert nur ein Verarbeitungsweg. Somit können die Nachrichten nur nacheinander abgearbeitet werden. Der Verarbeitungsweg selbst kann sich dabei aus mehreren Prozessen zusammensetzen, die hintereinander geschaltet sind. Bei einer solchen Architektur müssen Maßnahmen getroffen werden, damit sehr große Nachrichten die anderen Kommunikationsprozesse nicht zu lange blockieren. Bei einem sehr großen Nachrichtenaufkommen, kann sich der Verarbeitungsprozeß als Flaschenhals auswirken und Nachrichtenstaus entstehen lassen.

2. **lokal-nebenläufig:**
   Existieren für die Nachrichten unterschiedliche, vollständige Verarbeitungswege, müssen aber alle an diesen Wegen beteiligten Prozesse auf einem Rechner lokalisiert sein, so soll von einer *lokal-nebenläufigen* Nachrichtenverarbeitung gesprochen werden. Der Flaschenhals der seriellen Nachrichtenverarbeitung fällt nun weg. Zwar können Kommunikationsverbindungen mit einem sehr hohen Nachrichtenaufkommen auf mehrere Verarbeitungswege verteilt werden. Da alle Verarbeitungswege aber auf einem Rechner lokalisiert sein müssen, kann nur eine quasi-parallele Verarbeitung der Nachrichten stattfinden. Fällt dieser Rechner aus, so werden, wie bei der seriellen Verarbeitung, alle Kommunikationsverbindungen unterbrochen.

3. **verteilt-nebenläufig:**
   Lassen sich nun die nebenläufigen Verarbeitungswege auch auf mehrere Rechner verteilen, so soll von einer *verteilt-nebenläufigen* Nachrichtenverarbeitung gesprochen werden. Mit einer solchen Architektur läßt sich eine hohe Verfügbarkeit einzelner Kommunikationsverbindungen erreichen, wenn diese über eigene, möglicherweise alternative Verarbeitungswege geführt werden können.

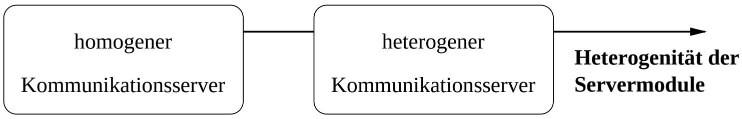
Auch für diese Dimension lassen sich drei Grade für die Taxonomie ableiten:



5.1.3 Heterogenität der Module

Liegt eine verteilte Architektur eines Kommunikationsservers vor, so läßt sich unterschieden, ob die Module oder Programme des Kommunikationsservers auch auf unterschiedlichen Plattformen installiert sein können oder ob sie auf eine Plattform beschränkt sind. So spricht man von einem *homogenen* Kommunikationsserver, wenn alle Module auf der gleichen Plattform (z.B. Sun-Rechner mit Solaris als Betriebssystem) installiert sein müssen. Ein *heterogener* Kommunikationsserver liegt dann von, wenn sich die Module auch auf verschiedenen Plattformen befinden können. So kann zum Beispiel ein Subsystem-Agent auf der Plattform Windows NT des Subsystems installiert sein, während sich die Prozesse der Nachrichtenverarbeitung auf einem zentralen Unix-Rechner befinden.

Entsprechend lassen sich zwei Grade für die Dimension *Heterogenität* der Taxonomie unterscheiden:



5.2 Klassen der Taxonomie

Die Klassen der Taxonomie werden nun durch die Zusammensetzung der drei Dimensionen *Verteiltheit*, *Nebenläufigkeit* und *Heterogenität* gewonnen. Sie werden in Abbildung 5.1 auf der nächsten Seite dargestellt.

Dabei lassen sich in Bezug auf die Verteiltheit drei Ebenen unterscheiden:

1. In der ersten Ebene besitzen alle Dimensionen den niedrigsten Grad (ungeteilt, seriell, homogen). Es liegt ein **monolithischer Kommunikationsserver** vor, bei dem die Funktionen der Serverkomponente von nur einem Prozeß ausgeführt werden.

2. In der zweiten Ebene ist eine Aufteilung des Servermoduls auf verschiedene Prozesse möglich, die Prozesse müssen aber auf einem Rechner lokalisiert sein. Diese Kommunikationsserver sind natürlicherweise homogen. Es lassen sich zwei verschiedene Klassen unterscheiden:

   (a) **zentraler, serieller Kommunikationsserver**
   Bei dieser Klasse besitzt die Dimension der Nebenläufigkeit den Grad 0 (seriell). Für alle Nachrichten wird nur ein Verarbeitungsweg bereitgestellt, der aber aus mehreren Prozessen bestehen kann.

   (b) **zentraler, nebenläufiger Kommunikationsserver**
   Hier kann die Nachrichtenverarbeitung lokal-nebenläufig durchgeführt werden. Da aber alle Prozesse des Servermoduls auf einem Rechner lokalisert sein müssen, handelt es sich um einen zentralen Kommunikationsserver.

   Das Prinzip dieser beiden Klassen wird mit Hilfe von Datenflußdiagrammen in Abbildung 5.2 und 5.3 auf Seite 50 verdeutlicht.

3. In der dritten Ebene schließlich sind alle sechs möglichen Kombinationen der Grade der drei Dimensionen für Kommunikationsserver denkbar:

   (a) homogener, verteilter, serieller Kommunikationsserver

   (b) heterogener, verteilter, serieller Kommunikationsserver

   (c) homogener, verteilter, lokal-nebenläufiger Kommunikationsserver

   (d) heterogener, verteilter, lokal-nebenläufiger Kommunikationsserver

   (e) homogener, verteilt-nebenläufiger Kommunikationsserver

   (f) heterogener, verteilt-nebenläufiger Kommunikationsserver

   Auch hier soll das Prinzip der Architekturen mit Hilfe von Datenflußdiagrammen beschrieben werden (siehe Abbildungen 5.4, 5.5 auf Seite 51 und 5.6 auf Seite 52).

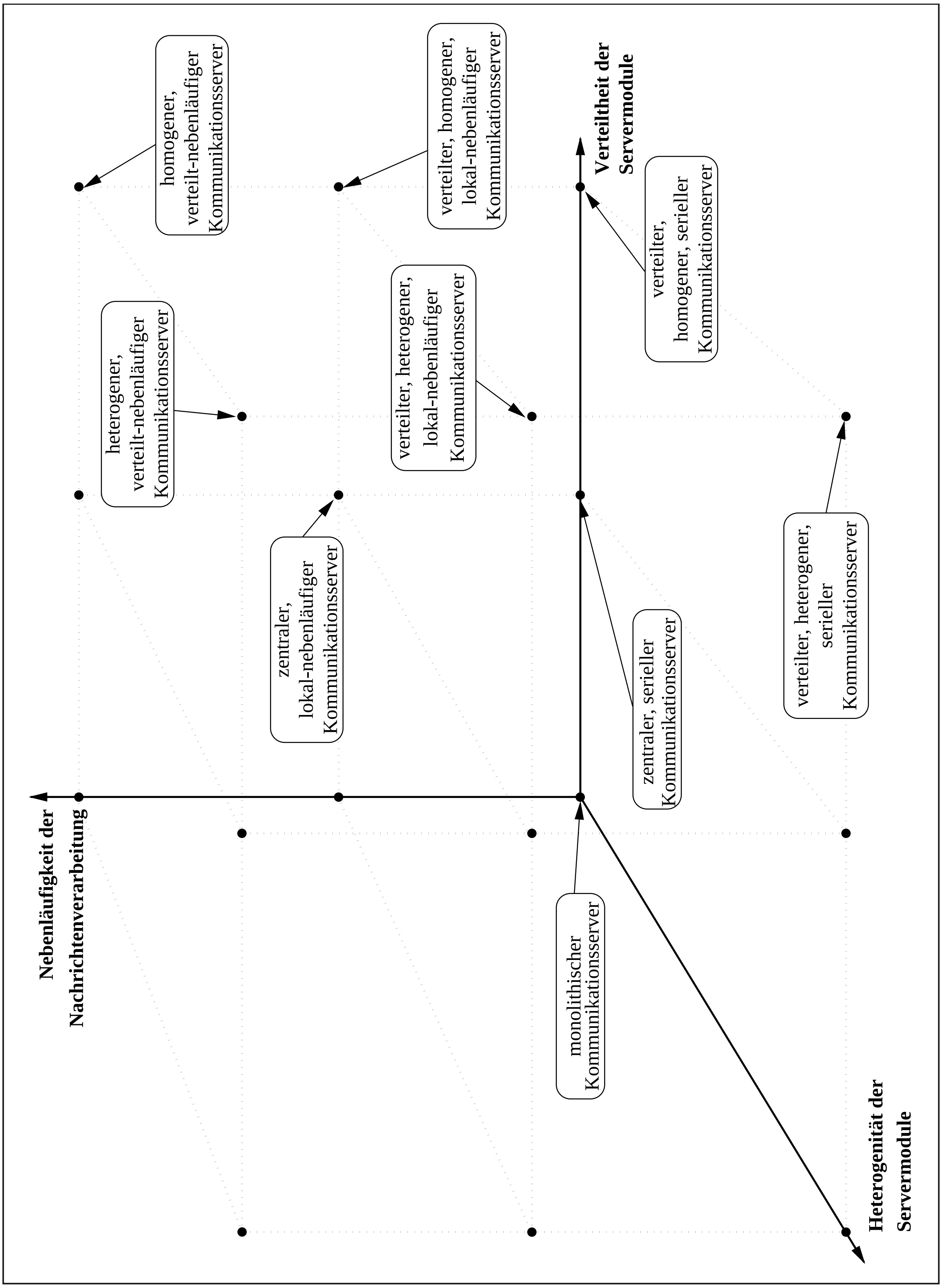


Abbildung 5.1: Dreidimensionales Diagramm der Klassen der Taxonomie.

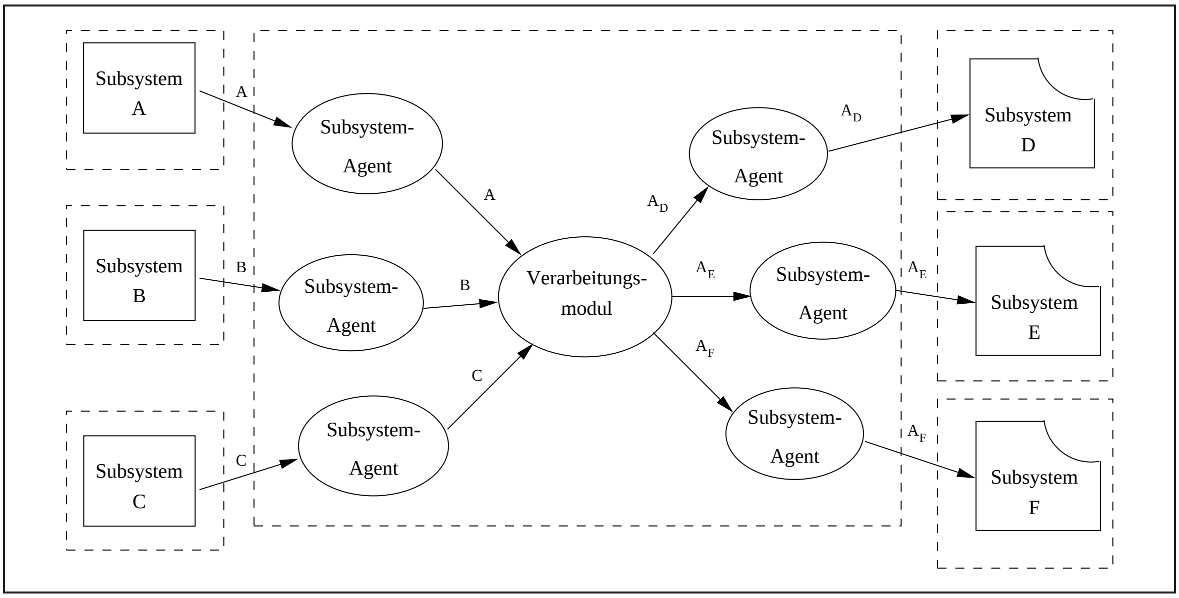


Abbildung 5.2: Datenflußdiagramm eines zentralen, seriellen Kommunikationsservers.

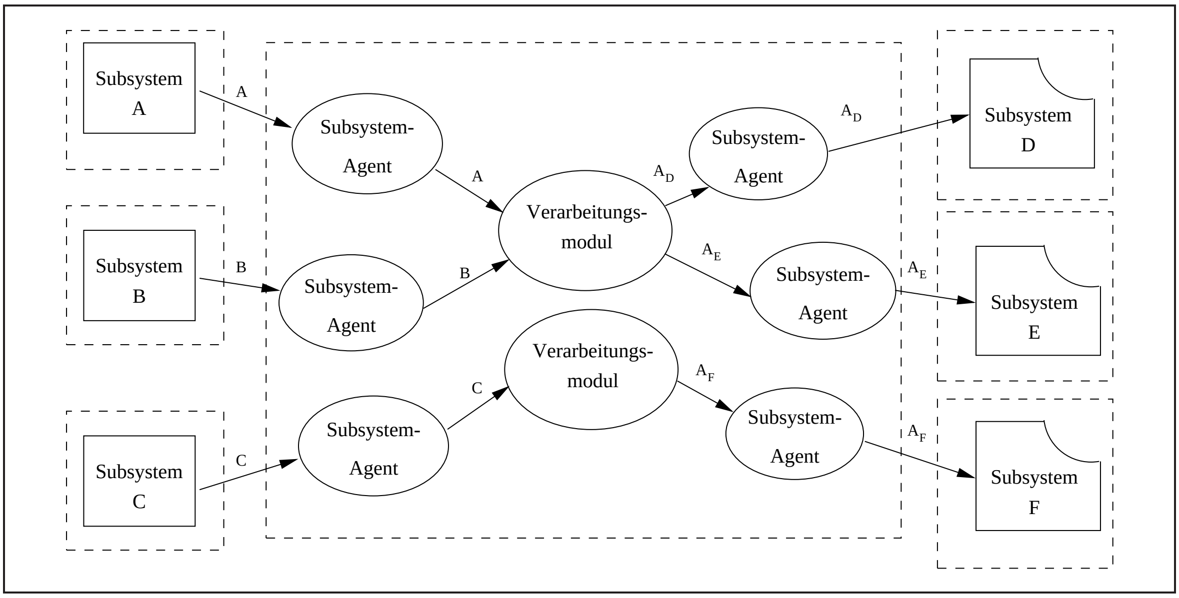


Abbildung 5.3: Datenflußdiagramm eines zentralen, lokal-nebenläufigen Kommunikationsservers.

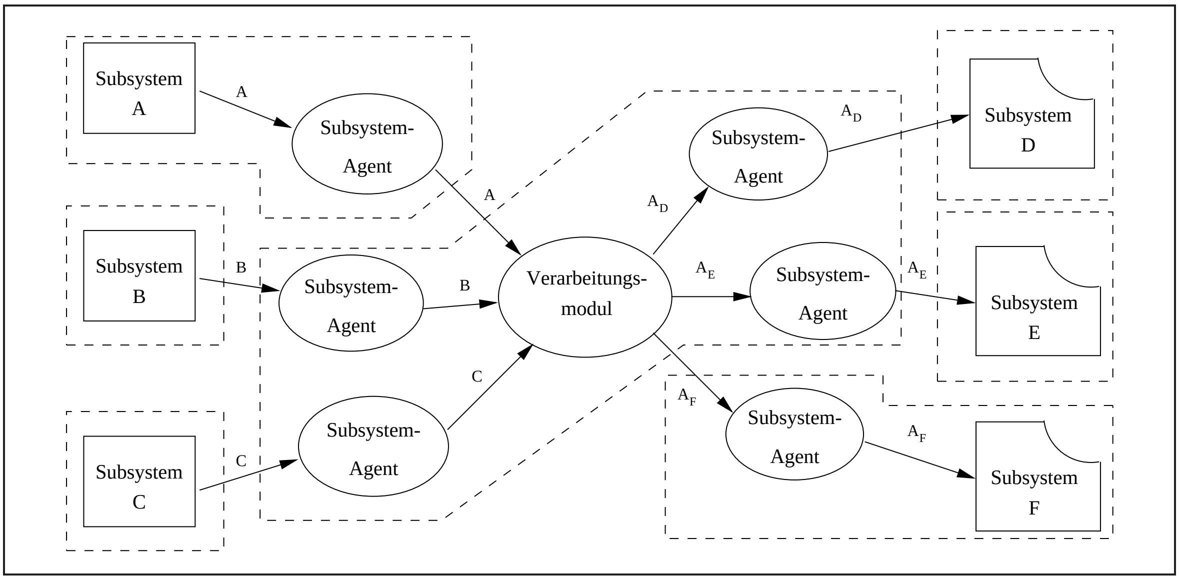


Abbildung 5.4: Datenflußdiagramm eines verteilten, seriellen Kommunikationsservers.

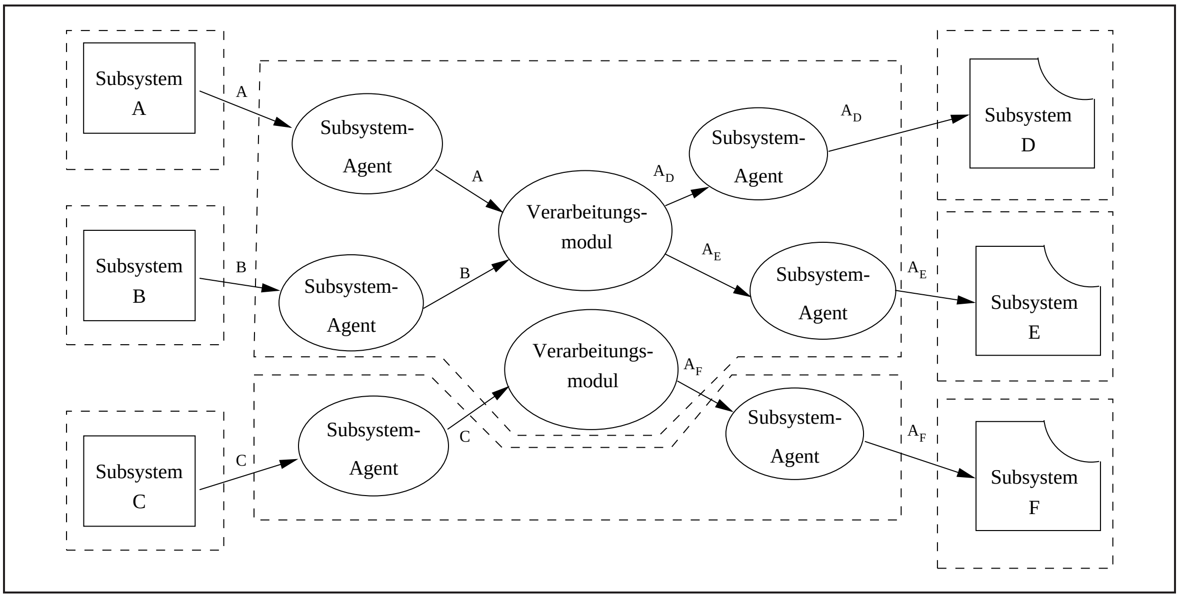


Abbildung 5.5: Datenflußdiagramm eines verteilten, lokal-nebenläufigen Kommunikationsservers.

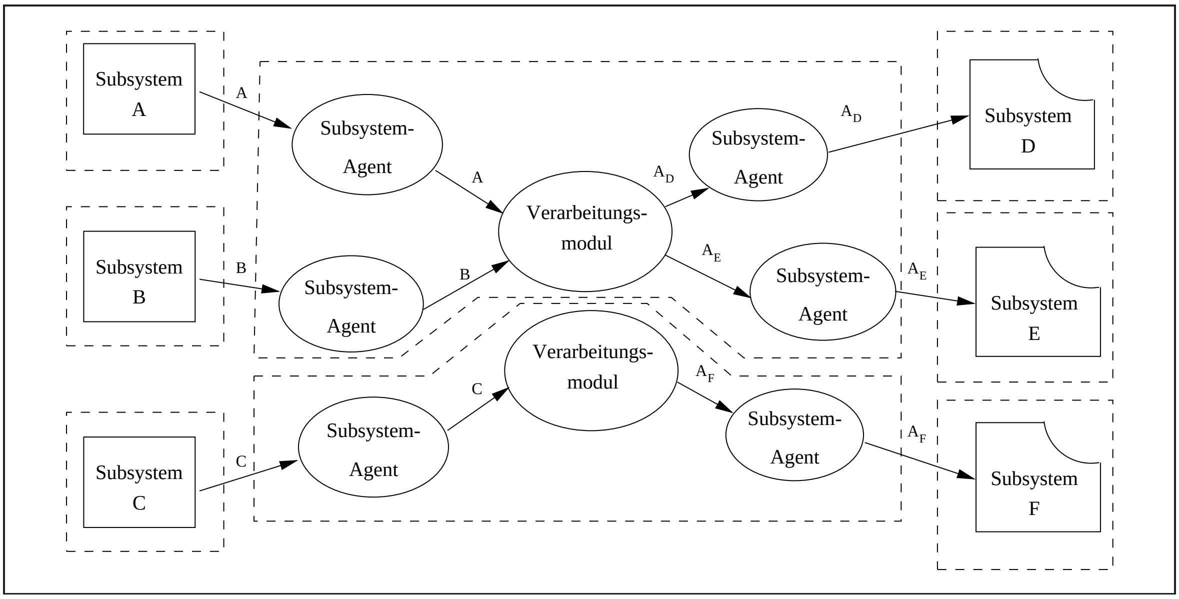


Abbildung 5.6: Datenflußdiagramm eines verteilt-nebenläufigen Kommunikationsservers.

Kapitel 6

Vergleich von Kommunikationsservern anhand des Anforderungskatalogs

Der Anforderungskatalog und die Taxonomie wurden im Rahmen dieser Arbeit als Grundlage für eine umfassende Analyse verschiedener Kommunikationsserver verwendet. Analysiert wurden die Architektur, der Nachrichtenfluß sowie der Funktionsumfang der Kommunikationsserver. Anschließend erfolgte eine Einordnung in die Taxonomie.

Bei der Analyse der Kommunikationsserver wurden drei verschiedene Wege beschritten:

1. Zwei der betrachteten Kommunikationsserver (Cloverleaf^{TM} von HCI und DataGate^{TM} von STC) wurden bzw. waren am Institut für Medizinische Informatik und Biomathematik der Uni-Kliniken Münster installiert und standen dort für eine ausführliche Analyse zur Verfügung. Bei dem Kommunikationsserver DataGate^{TM} erfolgte in dieser Zeit ein Release-Wechsel, so daß auch die Unterschiede zur Vorgängerversion mitberücksichtigt werden konnten.

2. Ein Kommunikationsserver (pro7-System^{TM} von ICD e.V., ProDV und ROKD) wurde Vorort beim Hersteller anhand eines Fragenkataloges analysiert.

3. An die Hersteller der übrigen Kommunikationsserver (HL7-Connection-Server^{TM} von `prompt!`, HealthView^{TM} von DEC, HCS^{TM} von SoftCon sowie TDM^{TM} von HP) wurde der Fragenkatalog mit der Bitte um eine möglichst ausführliche Beantwortung versandt.

Der in der zweiten und dritten Methode verwendete Fragenkatalog setzte sich aus den – leicht gekürzten – Ausführungen des Kapitels 4 mit den dort formulierten Anforderungen an Kommunikationsserver zusammen. Jede Anforderung wurde um eine Reihe von Fragen erweitert, die die genaue Umsetzung der Anforderung im Kommunikationsserver betrafen.

Während bei den ersten beiden Kommunikationsservern (Cloverleaf^{TM} und DataGate^{TM}) eine ausführliche Analyse möglich war, stützt sich die Analyse der übrigen Kommunikationsserver ausschließlich auf die Beantwortung der Fragebögen und eventueller Rückfragen. Leider war die Ausbeute der Fragebogenaktion äußerst gering. Nur vom Hersteller des Kommunikationsservers HCS^{TM} (SoftCon) wurde eine Antwort geschickt. Die Hersteller von HealthView^{TM} (Digital Equipment Corporation) und TDM^{TM} (Hewlett Packard) hatten mittlerweile den Support ihrer Produkte eingestellt, so daß diese Kommunikationsserver nicht mit in den Vergleich aufgenommen wurden. Der Hersteller des HL7-Connection-Server^{TM} (prompt!) konnte den Fragebogen aus Zeitgründen zwar nicht beantworten, stellte aber umfangreiches Informationsmaterial zur Verfügung, mit dessen Hilfe eine weitestgehende Beantwortung der Fragen vorgenommen werden konnte.

Die Ergebnisse der Analyse werden in den folgenden Abschnitten dargestellt. Dabei folgt zuerst eine Kurzdarstellung der einzelnen Kommunikationsserver (Abschnitt 6.1), sowie eine Einordnung der Produkte in die Taxonomie (Abschnitt 6.2 ab Seite 79). Anschließend werden in Abschnitt C ab Seite 114 alle Kommunikationsserver anhand der Auswertung der Fragebögen vergleichend gegenübergestellt.

6.1 Kurzbeschreibung der analysierten Kommunikationsserver

Im folgenden Abschnitt werden die im Rahmen dieser Arbeit analysierten Kommunikationsserver vorgestellt. Bei diesen Kurzbeschreibungen wird ein besonderer Wert auf die Darstellung der unterschiedlichen Architekturen der jeweiligen Serverkomponenten, sowie auf die Beschreibung des Nachrichtenflusses innerhalb dieser Komponenten gelegt. Wichtig für den Betrieb eines Kommunikationsservers ist auch die Konfigurierung und Überwachung der Kommunikationsverbindungen, das Fehlermanagement und die Persistenzsicherung der Nachrichten im Kommunikationsserver sowie der Schutz der Nachrichten vor einem unbefugten Zugriff. Auf diese Aspekte wird jeweils bei der Beschreibung der Kommunikationsserver eingegangen. Die Hardware- und Betriebsystemvoraussetzungen für den Kommunikationsserver werden jeweils am Ende einer Beschreibung dargestellt. Eine detailliertere Gegenüberstellung der Kommunikationsserver anhand des Anforderungskatalogs aus Kapitel 4 bietet der Anhang C.

Die Reihenfolge der Kommunikationsserver in dieser Kurzbeschreibung stellt keine Wertung dar. Die einzelnen Kommunikationsserver werden in alphabetischer Reihenfolge vorgestellt.

Die Darstellung der unterschiedlichen Nachrichtenflüsse wird im folgenden anhand eines durchgehenden Beispiels verdeutlicht. In diesem Beispiel wird von der folgenden Installation ausgegangen (siehe auch Abbildung 6.1):

- Es sind drei Subsysteme (A, B und C) mit dem Kommunikationsserver verbunden.
- Die Subsysteme A und B senden jeweils eine Nachricht an das Subsystem C.
- Subsystem C verschickt eine Nachricht, die an A und B übermittelt wird.

In den Abbildungen werden die Nachrichten nach ihren Sendern bezeichnet (Subsystem A sendet Nachricht A). Wird eine Nachricht in ein anderes Kommunikationsprotokoll transformiert, so erhält der Bezeichner der Nachricht ein Index, der das neue Kommunikationsprotokoll anzeigt (z.B. steht A_B für eine Nachricht vom Subsystem A, die im Kommunikationsprotokoll des Subsystems B vorliegt). Bei Nachrichten, die im Original-Kommunikationsprotokoll vorliegen, wird auf dieses Index verzichtet.

6.1.1 Cloverleaf™

Cloverleaf™ stellt das stark erweiterte Nachfolgeprodukt des Kommunikationsservers HCI-Link™ der Firma Healthcare Communication Inc. Dallas (HCI) dar. Dieser Kommunikationsserver wird in Deutschland von der deutschen HCI-Niederlassung *Health-Comm* mit Sitz in Essen vertrieben.

Für eine ausführliche Analyse wurde dem Institut für Medizinische Informatik und Biomathematik in Münster für einen Testzeitraum von HCI eine Lizenz zur Verfügung gestellt und Cloverleaf™ in der Version 3.1.3 installiert. Die folgenden Ausführungen beziehen sich daher auf diese Version. Die in dieser Beschreibung abgebildeten Bildschirmmasken von Cloverleaf™ sind Screenshots von dieser Installation. Diese Beschreibung orientiert sich an dem Handbuch zum Kommunikationsserver [Hea96].

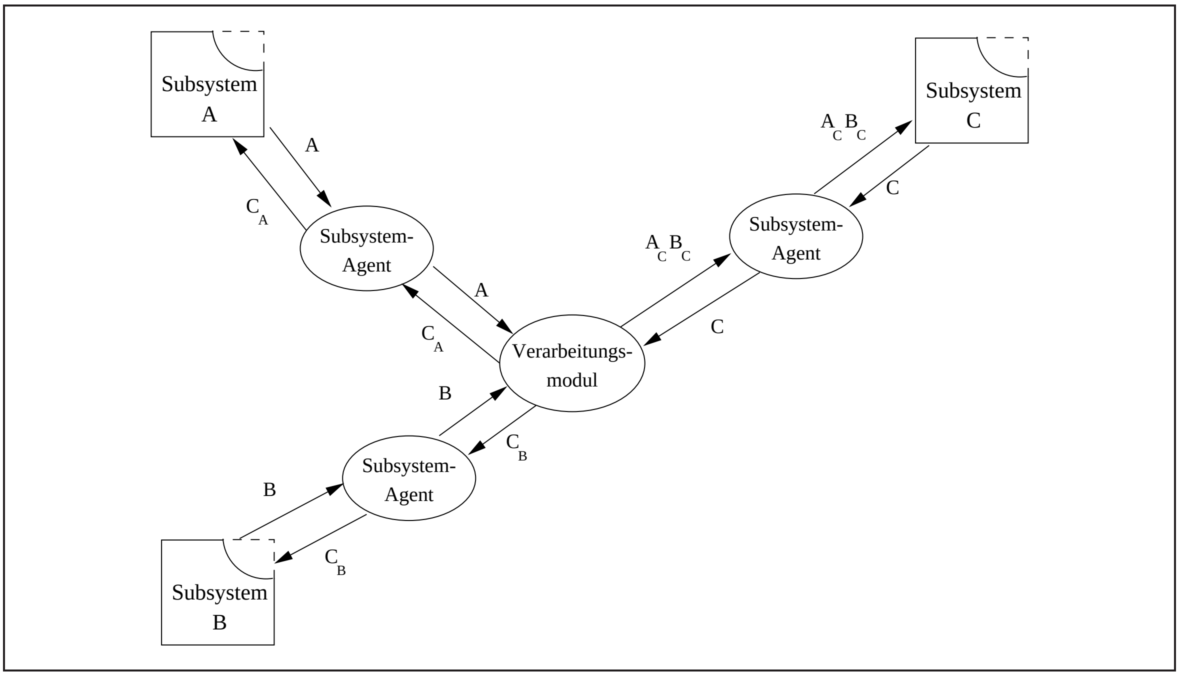


Abbildung 6.1: Darstellung eines beispielhaften Nachrichtenaustausches zwischen drei Subsystemen *A*, *B* und *C* über einen Kommunikationsserver. Dieses Beispiel wird durchgängig für die Darstellung der unterschiedlichen Nachrichtenflüsse der beschriebenen Kommunikationsserver verwendet.

Architektur der Serverkomponente

Das Modul, das bei CLOVERLEAF^{TM} die Funktionen der Serverkomponente ausführt, wird als *Cloverleaf Engine* bezeichnet. Dieses Servermodul kann die Kommunikationsverbindungen auf mehrere verschiedene Serverprozesse verteilen. Jeder Serverprozeß besteht dabei aus einem *Command Thread*, einem *Translation Thread* sowie mindestens einem *Protocol Thread*.

Command Thread: Der *Command Thread* ist die Steuerzentrale eines Serverprozesses. Hier werden alle Aktionen koordiniert.

Translation Thread: Der *Translation Thread* ist für die Verarbeitung (Identifikation, Routing, Transformation) der Nachrichten zuständig, die auf den über diesen Serverprozeß geführten Kommunikationsverbindungen anfallen. Er realisiert also die Funktionen der Verarbeitungskomponente für diese Verbindungen.

Protocol Thread: Für jedes an den Serverprozeß angeschlossene Subsystem wird schließlich ein *Protocol Thread* unterhalten. Diese sind für den Nachrichtenaustausch zwischen dem Serverprozeß und den Subsystemen zuständig, stellen also die Subsystem-Agenten von CLOVERLEAF^{TM} dar.

Der Aufbau des Servermoduls wird in OMT-Notation in Abbildung 6.2 gezeigt. Die Zuordnung der Module zu den Komponenten wird durch die gestrichelt gezeichneten Kästen angedeutet.

Die Aufteilung eines Serverprozesses in mehrere *Threads* erlaubt einen hohen Grad an Parallelität in der Nachrichtenverarbeitung. Allerdings lassen sich die *Threads* nicht auf mehrere Rechner verteilen, so daß keine echte Parallelität erreicht werden kann. Da das Servermodul aus mehreren Serverprozessen aufgebaut werden kann, können parallele Verarbeitungswege definiert werden. Bisher lassen sich aber auch die Serverprozesse nicht auf unterschiedliche Rechner verteilen.

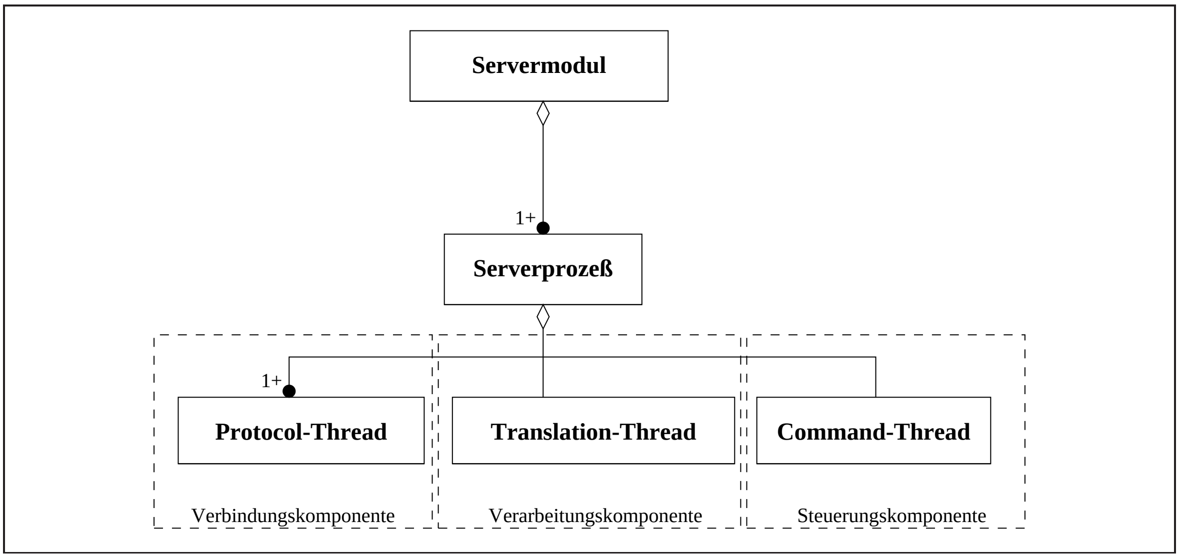


Abbildung 6.2: Darstellung der Module der Serverkomponente von CLOVERLEAF^{TM} in OMT-Notation. Die Zuordnung der Module zur Steuerungs-, Verbindungs- und Verarbeitungskomponente wird durch die gestrichelt gezeichneten Kästen angedeutet.

Nachrichtenfluß

Die Übertragung der Nachrichten zwischen den Subsystemen und CLOVERLEAF^{TM} erfolgt durch die *Protocol Threads* der Serverprozesse (entsprechen den Subsystem-Agenten). Jeder *Protocol Thread* ist genau einem Serverprozeß zugeordnet. An jeden *Protocol Thread* kann genau ein Subsystem angeschlossen werden. CLOVERLEAF^{TM} unterstützt eine Reihe standardisierter dateiloser und dateibasierter Transportprotokolle. Alle Protokolle lassen sich über einen speziellen Editor konfigurieren. Erweiterungen der Transportprotokolle lassen sich über eine sogenannte *Protocol Definition Language* (PDL, um spezielle Funktionen erweitertes TCL [Ous94]) vornehmen. Über diese PDL lassen sich auch weitere Transportprotokolle implementieren.

Hat ein *Protocol Thread* eine Nachricht empfangen, so wird als erstes eine Kopie in einer Recover-Datenbank gespeichert (siehe Abbildung 6.3). Anschließend wird die Nachricht an den *Translation Thread* des selben Serverprozesses weitergereicht. Hier erfolgt die Nachrichtenidentifikation und die Ermittlung der Empfänger.

Für die Identifikation der Nachrichten bietet CLOVERLEAF^{TM} verschiedene Methoden an. Nachrichten aus proprietären Kommunikationsprotokollen lassen sich über den Nachrichtentyp identifizieren, indem die genaue Position und Feldlänge des Datenfeldes in der Nachricht angegeben wird, in dem der Nachrichtentyp codiert ist. Eine zweite Methode verwendet Schlüsselwörter (siehe auch Abschnitt 4.1.3). Die Nachrichtenidentifizierung erfolgt über ein TCL-Skript, dessen Rückgabewert ein definiertes Schlüsselwort darstellt und für das Routing der Nachricht verwandt wird. Nachrichten aus standardisierten Kommunikationsprotokollen werden grundsätzlich über die im jeweiligen Standard definierten Nachrichtentypen und Identifizierungsregeln identifiziert.

Die Empfänger einer Nachricht werden über den Nachrichtentyp bzw. über das von der Identifizierungsfunktion zurückgegebene Schlüsselwort anhand von Routing-Tabellen ermittelt. Es besteht allerdings auch die Möglichkeit, das Routing vollständig durch ein TCL-Skript durchführen zu lassen.

Nach jedem Verarbeitungsschritt wird wieder eine Kopie der nun veränderten Nachricht in der Recover-Datenbank abgelegt. Sind mehrere Empfänger vorgesehen, wird für jeden Empfänger eine Kopie der Nachricht erstellt. Die Kopien stellen nun eigenständige Nachrichten dar. Nach einer gegebenenfalls

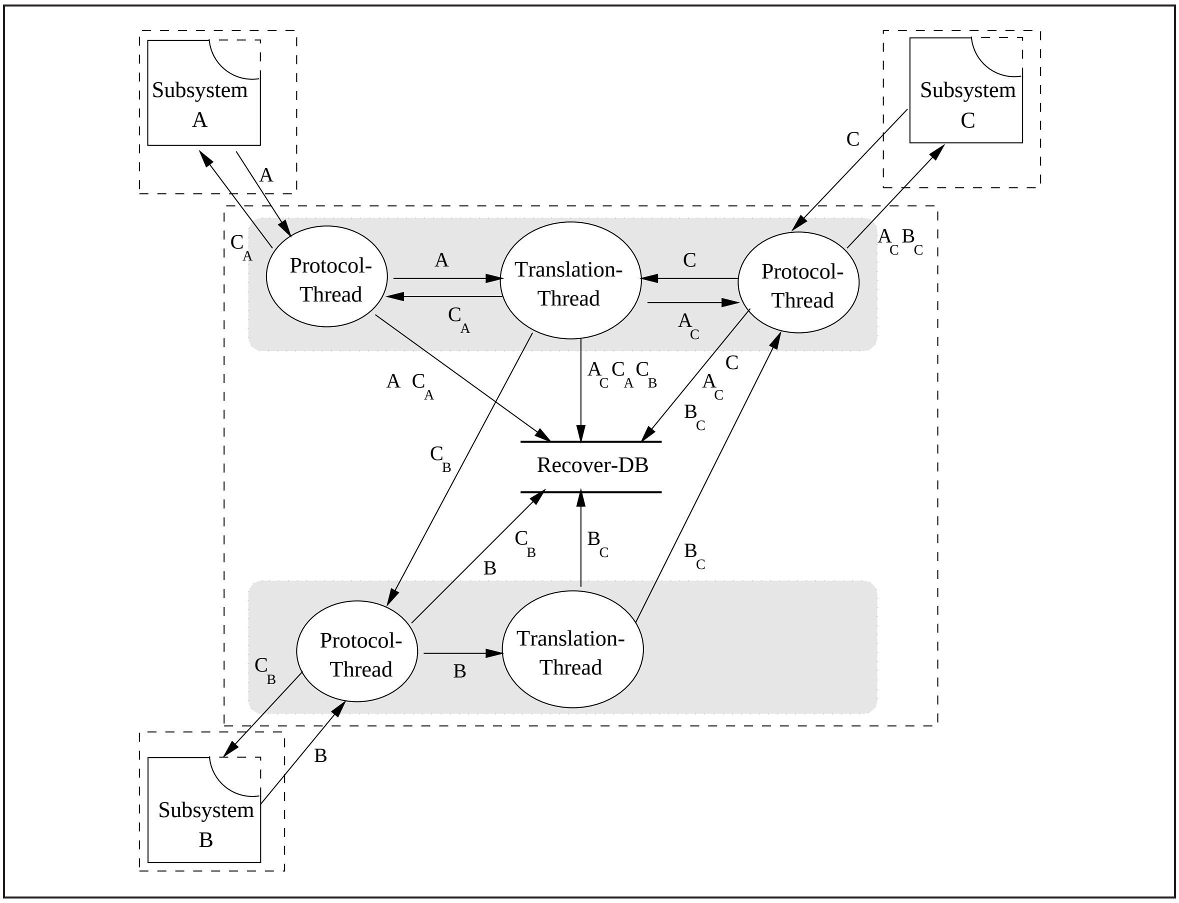


Abbildung 6.3: Nachrichtenfluß innerhalb von CLOVERLEAF^{TM}. Dargestellt werden zwei Serverprozesse (durch die schattierten Bereiche dargestellt) über die drei Subsysteme angeschlossen sind.

notwendigen Transformation der Nachrichten, werden sie an die *Protocol Threads*, die die Verbindung zu den Empfängern unterhalten, übertragen.

Vor und meistens auch nach jedem Verarbeitungsschritt können vom Administrator des Kommunikationsservers TCL-Skripte in den Verarbeitungsprozeß eingehängt werden. Über diese Skripte lassen sich Funktionen realisieren, die von CLOVERLEAF^{TM} nicht angeboten werden (z.B. spezielle Konvertierungsfunktionen oder Regeln für die Nachrichtenidentifikation). Der standardmäßig vorgesehene Nachrichtenfluß läßt sich über diese TCL-Skripte verändern. So kann zum Beispiel ein *Protocol Thread* umgangen werden, indem nach der Transformation ein TCL-Skript die Übertragung der Nachricht vornimmt. Innerhalb der TCL-Skripte kann der Programmierer auf die im Kommunikationsserver definierte Struktur der Nachrichten zugreifen.

Für die Synchronisation des Nachrichtenaustausches zwischen zwei Subsystemen bietet CLOVERLEAF^{TM} eine besondere Unterstützung an. Im Kommunikationsserver lassen sich Nachrichten (*Messages*) und Antworten auf diese Nachrichten (*Replies*) unterscheiden. Eine Kommunikationsverbindung zu einem Subsystem läßt sich nun so konfigurieren, daß CLOVERLEAF^{TM} nach der Übermittlung einer Nachricht an das Subsystem auf eine Antwortnachricht wartet. Erst wenn diese eingetroffen ist, werden weitere Nachrichten an das Subsystem übertragen. Eine Synchronisierung des Nachrichtenaustausches auf Nachrichtenebene zwischen zwei Subsystemen läßt sich so leicht implementieren.

Konfigurierung und Überwachung der Kommunikationsverbindungen

Alle Konfigurationen von CLOVERLEAF^{TM} werden in einer an TCL angelehnten Beschreibungssprache vorgenommen und in ASCII-Dateien gespeichert. Für alle Konfigurationsdateien werden Editoren mit graphischen Oberflächen angeboten, auch für die Definition der Transformationsregeln. Die Regeln für die Identifikation der Nachrichten aus proprietären Kommunikationsprotokollen müssen allerdings in TCL programmiert werden. Für die Implementierung der TCL-Skripte stellt CLOVERLEAF^{TM} eine Entwicklungsumgebung, bestehend aus Editor, Interpreter und Debugger, zur Verfügung.

Die Konfigurierung der Kommunikationsverbindungen kann ebenfalls mit Hilfe eines graphischen Editors erfolgen. Die Darstellung der Kommunikationsverbindungen erfolgt hier in Form eines Netzwerkes (gleiche Darstellung wie im System-Monitor, Abbildung 6.4).

Für die Überwachung der Kommunikationsverbindungen und der Verarbeitungsprozesse im Kommunikationsserver stellt CLOVERLEAF^{TM} einen sogenannten *Netzwerk-Monitor* zur Verfügung. In diesem werden alle Kommunikationsverbindungen in Form eines Netzwerkes dargestellt (siehe Abbildung 6.4). Von hier aus lassen sich alle Verbindungen und alle Serverprozesse steuern und überwachen. Im oberen Abschnitt des Monitors werden die Kommunikationsverbindungen in Form eines Netzwerkes dargestellt. Die Farben der Knoten und Kanten symbolisieren den jeweils aktuellen Zustand der Verbindungen. Im mittleren Abschnitt werden die Serverprozesse aufgelistet. Auch hier symbolisieren die Farben den Zustand eines Prozesses (aktiv oder angehalten). Im untersten Abschnitt werden die zuletzt ausgeführten Kommandos textuell dargestellt.

Für eine flexible Überwachung der Kommunikationsverbindungen werden Alarmierungsfunktionen bereitgestellt. Mit ihnen lassen sich zahlreiche Situationen im Verarbeitungsprozeß sowie im Systemzustand (z.B.: Speicherplatzauslastung) kontrollieren.

Das Logging-Modul von CLOVERLEAF^{TM} erlaubt eine konfigurierbare Protokollierung aller Vorgänge in den Serverprozessen. Je nach Detaillierungsgrad der Protokollierung werden auch die Nachrichten unverschlüsselt mit in die Log-Dateien aufgenommen. Es kann allerdings für alle Kommunikationsverbindungen nur eine Log-Datei angelegt werden. Der Umfang der Log-Informationen läßt sich für jeden Serverprozeß und auch für jeden Subsystem-Agenten (*Protocol Thread*) getrennt konfigurieren.

Neben der Möglichkeit, die Nachrichten mit in die Log-Dateien aufzunehmen, lassen sich für jede Kommunikationsverbindung Nachrichtenarchive anlegen. Diese bestehen aus unverschlüsselten ASCII-Dateien, in denen die Nachrichten in der intern in CLOVERLEAF^{TM} verwendeten Darstellungsform (Erweiterung der eigentlichen Nachricht um Angaben wie Priorität, Bezeichner der *Protocol Threads* des Senders und des Empfängers, etc.) abgelegt werden. Mit Hilfe eines Tools (*Saved Message Administration Tool* SMAT) lassen sich diese Nachrichtenarchive einsehen und die Nachrichten zum Beispiel erneut versenden.

Fehlermanagement und Persistenzsicherung

Für die Behandlung von fehlerhaften Nachrichten und für die Sicherung der Persistenz der Nachrichten im Kommunikationsserver wird ein Netzwerk-Modell-DBMS (RDM von Raima Corp.) verwendet. In einer *Error-Datenbank* werden alle fehlerhaften Nachrichten gespeichert. Eine *Recover-Datenbank* ist für die Persistenzsicherung zuständig. In ihr werden alle Nachrichten kontinuierlich gespeichert. Nach jedem Verarbeitungsschritt wird eine Nachricht in der nun neuen Version in der Datenbank abgelegt, bevor die alte Version der Nachricht gelöscht werden kann. Erst nach der Bestätigung des erfolgreichen Empfangs einer Nachricht durch das entsprechende Subsystem bzw. den Kommunikationsklienten wird die Nachricht endgültig aus der Recover-Datenbank gelöscht. Somit ist sichergestellt, daß eine Nachricht nicht im Kommunikationsserver verloren gehen kann. Nach einem Systemabsturz kann der Verarbeitungsprozeß bei jeder Nachricht wieder an der Stelle beginnen, die zuletzt erfolgreich abgeschlossen wurde.

Für die Verwaltung der Nachrichten durch den Administrator wird ein gemeinsames Modul für beide Datenbanken bereitgestellt.

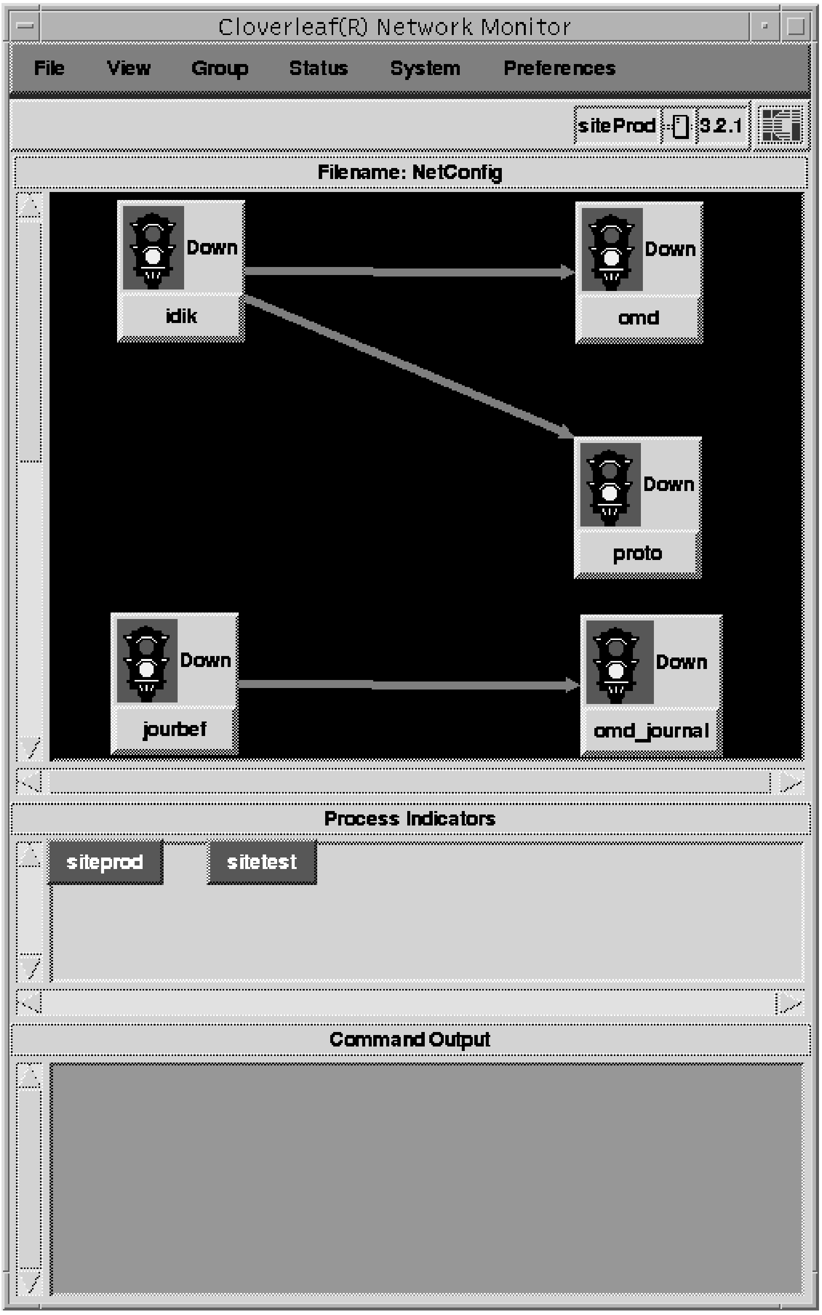


Abbildung 6.4: Benutzeroberfläche des System-Monitors (genannt *Network-Monitor*) von CLOVERLEAF^{TM}. Eine Beschreibung des Monitors befindet sich im Text auf der Seite 58. Die Abbildung stellt eine Konfiguration aus den Uni-Kliniken in Münster dar.

Datenschutz

Beim Schutz der Daten vor einem unbefugten Zugriff verläßt sich CLOVERLEAF^{TM} vollständig auf die Schutzmechanismen des Betriebsystems. Log-Dateien und Nachrichtenarchive lassen sich nicht verschlüsseln. Sollen die Nachrichten zwischen den Subsystemen und dem Kommunikationsserver verschlüsselt ausgetauscht werden, so können in den *Protocol Threads* über die PDL Verschlüsselungsmethoden implementiert werden.

Die einzelnen Module und Tools von CLOVERLEAF^{TM} unterliegen keinem weiteren Zugriffsschutz. Den Benutzern können aber verschiedene Benutzerrollen zugeteilt werden, die ihnen unterschiedliche Befugnisse erteilen. Die Steuerung der Benutzerrollen und der mit ihnen verbundenen Rechte erfolgt über die Gruppenzugehörigkeit eines Anwenders im UNIX-Betriebsystem.

Hardware- und Betriebsystemvoraussetzungen

CLOVERLEAF^{TM} unterstützt zur Zeit die folgenden Plattformen:

Hardware	**Betriebssystem**
Dec	OSF I
HP 9000er	HP-UX
IBM RS 6000	AIX 4.1.x
Intel	Windows NT
SUN Sparc	Solaris 2.5.x

6.1.2 DataGate^{TM}

DATAGATE^{TM} ist ein Produkt der *Software Technologies Corporation* (STC) und wird in Deutschland von der Gesellschaft für Offene Systeme in der Medizin mbH (OSM) Essen und der Hinz Fabrik GmbH Berlin vertrieben. DATAGATE^{TM} ist schon seit längerer Zeit an den Uni-Kliniken in Münster installiert und stand auch für die Untersuchungen im Rahmen dieser Arbeit zur Verfügung. Für eine tiefergehende Analyse des Kommunikationsservers wurde im Rahmen dieser Arbeit eine Kommunikationsverbindung implementiert. Diese Ergebnisse werden im Kapitel 7 dargestellt.

Die folgende Beschreibung von DATAGATE^{TM} bezieht sich auf die Version 3.0 [Sof95b]. Auf die Änderungen und Erweiterungen in der Version 3.1 von DATAGATE^{TM} geht der letzte Abschnitt dieser Darstellung ein. Die in der Beschreibung dargestellten Bildschirmmasken stammen von der in Münster installierten Version 3.0.

Architektur der Serverkomponente

Für die Ausführung der Funktionen der Serverkomponente stellt DATAGATE^{TM} zwei verschiedene Module bereit, das

1. **Servermodul**, das die Aufgaben der Verarbeitungskomponente übernimmt, und die
2. ***Communication Clients***, die die Subsystem-Agenten von DATAGATE^{TM} darstellen, also die Funktionen der Verbindungskomponente ausführen.

Das Servermodul unterteilt sich weiter in einen sogenannten DATAGATE-Server (DG-Server) und einen *Control Brocker*. Während der DG-Server für die Nachrichtenidentifikation und -transformation zuständig ist, übernimmt der Control Broker das Routing sowie den Austausch der Nachrichten zwischen den *Communication Client*s und dem Servermodul.

Die *Communication Clients* stellen kleine, in C programmierte Programme dar. In ihnen lassen sich beliebige Transportprotokolle implementieren. Für viele spezielle Transportprotokolle existieren fertige *Communication Clients*. Da die Quellcodes der *Communication Clients* mit zum Lieferumfang von DATAGATE^{TM} gehören, können sie auch vom Administrator des Kommunikationsservers angepaßt bzw. implementiert werden.

In Abbildung 6.5 werden die Beziehungen zwischen den Modulen der Serverkomponente in OMT-Notation dargestellt. Dabei zeigt sich, daß alle Subsystem-Agenten (*Communication Clients*) an nur ein Servermodul angeschlossen werden, alle Kommunikationsverbindungen also über dieses Modul verlaufen müssen.

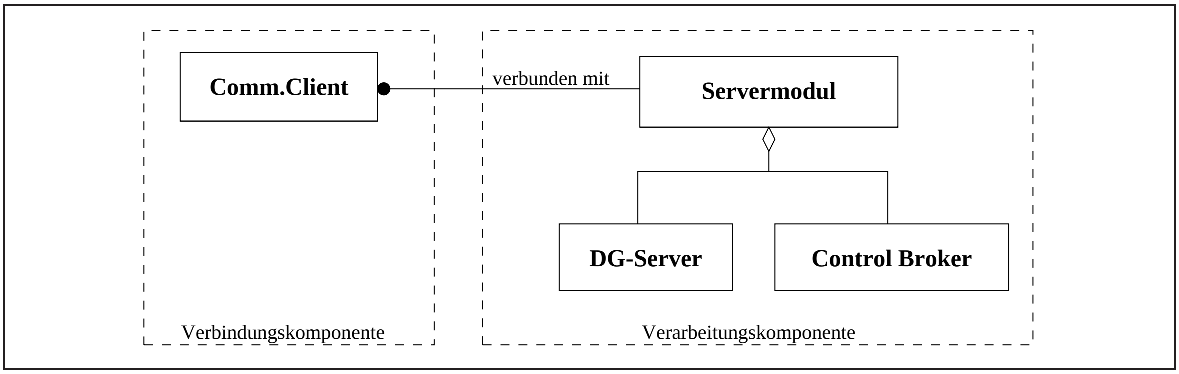


Abbildung 6.5: Darstellung der Module der Serverkomponente von DATAGATE^{TM} in OMT-Notation. Die Zuordnung der Module zur Verbindungs- und Verarbeitungskomponente wird durch die gestrichelt gezeichneten Kästen vorgenommen.

Nachrichtenfluß

Der Nachrichtenaustausch zwischen DATAGATE^{TM} und den Subsystemen erfolgt über die *Communication Clients*, die den Subsystem-Agenten eines Kommunikationsservers entsprechen. In der Regel werden für den Anschluß eines Subsystems zwei Subsystem-Agenten verwendet: Der erste (mit *CC-In* abgekürzt) empfängt die Nachrichten vom Subsystem und leitet sie an das Servermodul weiter. Der zweite (mit *CC-Out* abgekürzt) ist für den umgekehrten Weg zuständig. Er überträgt die Nachrichten vom Servermodul zum Subsystem.

Die Trennung der Subsystem-Agenten in CC-In und CC-Out hat den Vorteil, daß beide Verbindungen getrennt kontrolliert werden können. Eine Synchronisation der Nachrichtenübertragung auf Nachrichtenebene zwischen Servermodul und Subsystem wird durch diese Aufteilung allerdings sehr erschwert. Die Subsystem-Agenten lassen sich aber auch bidirektional anlegen.

Der weitere Nachrichtenfluß in DATAGATE^{TM} gestaltet sich recht einfach (vgl. Abbildung 6.6): Die Subsystem-Agenten übertragen die Nachrichten zum Servermodul, wo sie identifiziert, geroutet und transformiert werden. Für die Nachrichtenidentifizierung und das Routen verwendet DATAGATE^{TM} ausschließlich eine Schlüsselwortmethode (vgl. Abschnitt 4.1.3). Die hierfür notwendigen Identifizierungsregeln werden über eine spezielle Programmiersprache implementiert.

Das Servermodul stellt die Nachrichten nach der Transformation in die Warteschlangen der CC-Outs der Empfänger. Aus diesen werden sie von den entsprechenden Subsystem-Agenten zur Übertragung an die Empfänger entnommen. Warteschlangen für die ankommenden Nachrichten (CC-In) sind standardmäßig nicht vorgesehen, lassen sich aber in den entsprechenden Subsystem-Agenten implementieren.

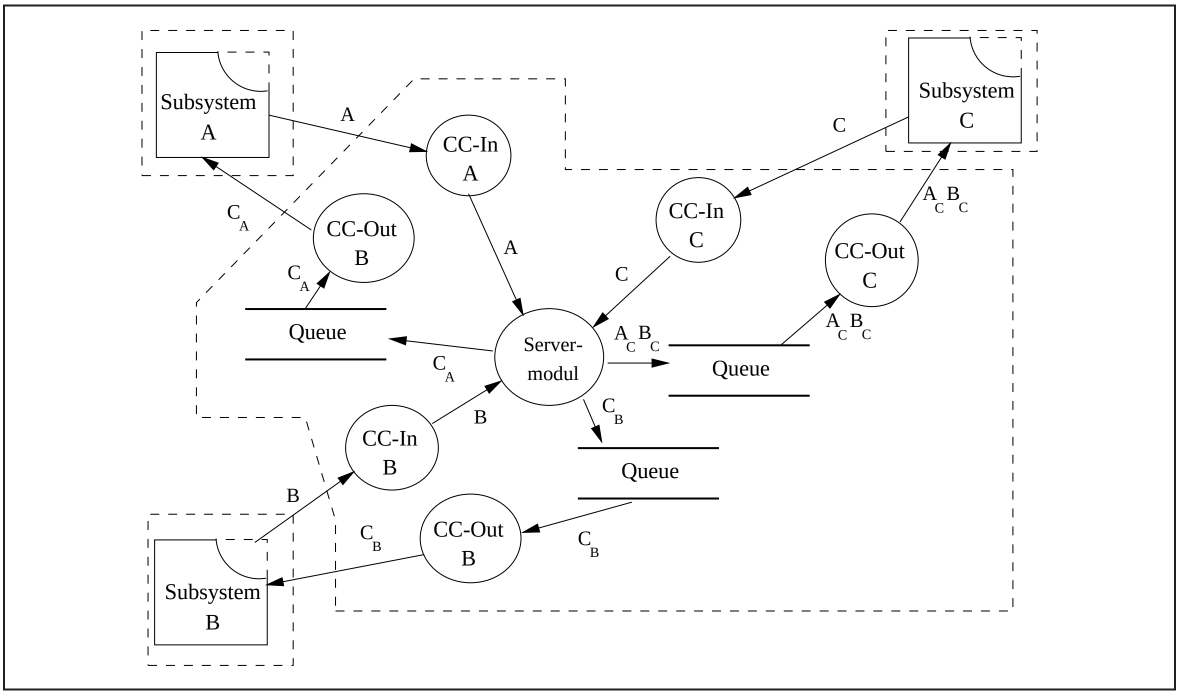


Abbildung 6.6: Darstellung des Nachrichtenflusses über die Module von DataGate^{TM}. Mit *CC-In* und *CC-Out* werden die *Communication Client*s (Subsystem-Agenten) für das Empfangen bzw. Senden der Nachrichten abgekürzt.

Konfigurierung und Überwachung der Kommunikationsverbindungen

Die Konfigurierung der Kommunikationsverbindungen und der Subsystem-Agenten erfolgt über eine Reihe von Konfigurationsdateien. In diesen werden zum Beispiel die Transformationsregeln in einer an C-Funktionen erinnernden Beschreibungssprache definiert und im ASCII-Format abgelegt. Spezielle Editoren für diese Dateien existieren bisher nicht. Lediglich das Nachrichten-Routing läßt sich über einen Editor mit einer graphischen Oberfläche definieren (siehe Abbildung 6.7).

In diesem Editor werden auf der linken Seite alle Subsystem-Agenten der sendenden Subsysteme dargestellt. Wird eines der Icons ausgewählt, erscheinen auf der rechten Seite des Editors alle Subsystem-Agenten, an die von dem ausgewählten Subsystem-Agenten Nachrichten versandt werden. Wird eine Kommunikationsverbindung ausgewählt, indem auch auf der linken Seite ein Subsystem-Agent markiert wird, erscheinen in der Mitte des Editors die über diese Verbindung ausgetauschten Nachrichtentypen (bzw. die Schlüsselwörter, die die Identifizierungsfunktionen liefern). Zusätzlich wird für jede Nachricht der Name der Tabelle angezeigt, in der die Transformationsregeln für diesen Nachrichtentyp bei diesem Empfänger definiert wurden.

Über den System-Monitor von DataGate^{TM} lassen sich alle Kommunikationsverbindungen steuern und überwachen. Die Kommunikationsverbindungen werden allerdings graphisch nicht dargestellt (siehe Abbildung 6.8). Es erfolgt lediglich eine tabellarische Auflistung der Subsystem-Agenten mit Darstellung der Zustände der Verbindungen zu den Subsystemen („*down*“ oder „*up*“). Zusätzlich werden einige Statistiken angezeigt (z.B. Nachrichtendurchsatz). In der obersten Zeile werden das Servermodul (Icon mit „S“) sowie alle angeschlossenen Subsystem-Agenten (Icon mit „C“) in Form von Icons dargestellt. Von hier aus erfolgt der Zugriff auf die Log-Dateien der Subsystem-Agenten sowie die Steuerung der Verbindungen.

Log-Dateien lassen sich für jedes angeschlossene Subsystem getrennt erstellen. Zusätzlich werden die Vorgänge in dem Servermodul in einer eigenen Log-Datei protokolliert. Der Umfang der Log-

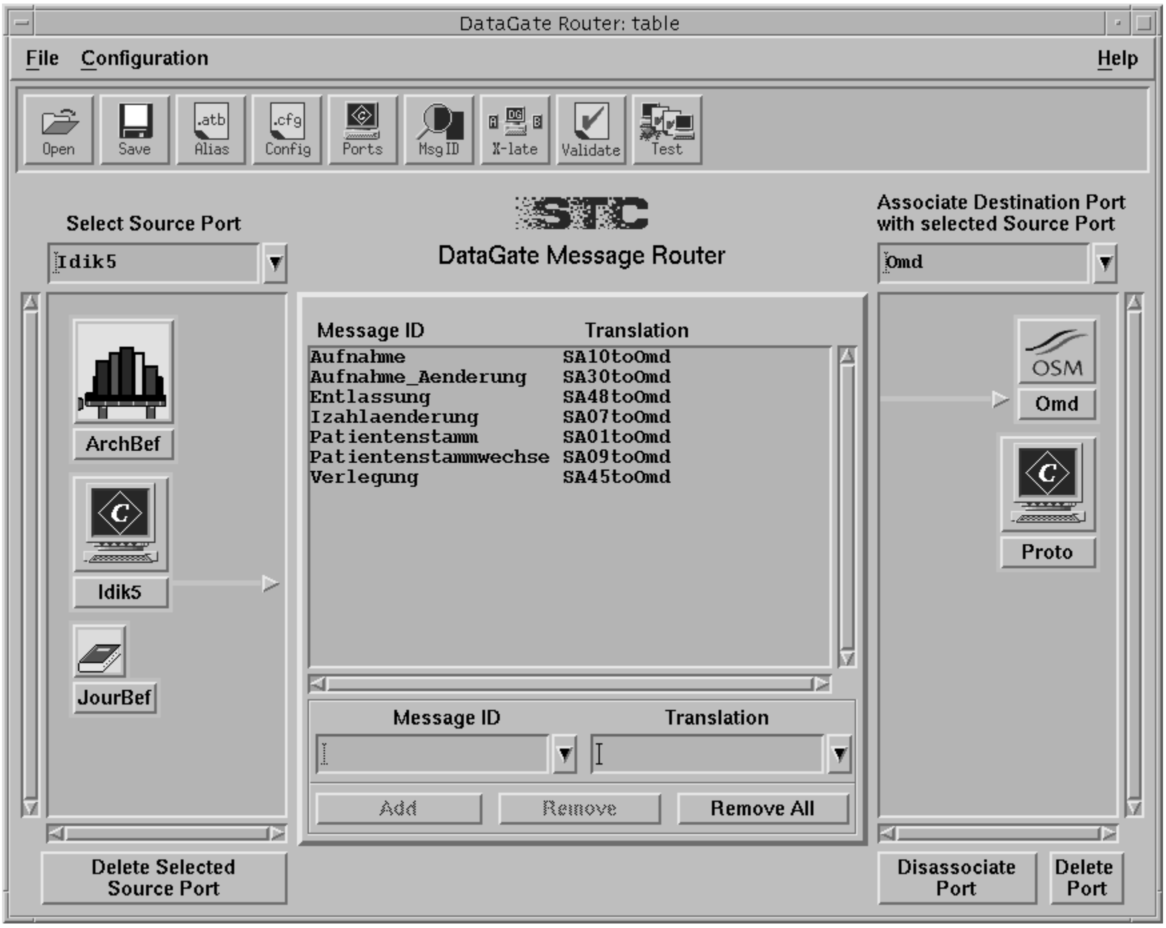


Abbildung 6.7: Darstellung des Editors von DataGate^{TM} für die Konfigurierung des Nachrichten-Routing. Es wird ein Ausschnitt aus der derzeitigen Konfiguration an den Uni-Kliniken in Münster gezeigt. Eine Beschreibung der Oberfläche wird im Text auf der Seite 62 gegeben.

Informationen läßt sich getrennt konfigurieren. Nachrichten werden nicht mit in die Log-Dateien aufgenommen.

Zur Speicherung der Nachrichten lassen sich an jeder Outbound-Queue (Warteschlange für Nachrichten, die an ein Subsystem gesandt werden sollen) Nachrichtenarchive einrichten. Alle Nachrichten, die in die Queue gestellt werden, werden dann auch im Nachrichtenarchiv abgelegt. Die Nachrichtenarchive werden durch unverschlüsselte ASCII-Dateien gebildet.

Sowohl für die Log-Dateien, als auch für die Nachrichtenarchive stellt DataGate^{TM} jeweils ein Modul für die Auswertung der Informationen bzw. Verwaltung der Nachrichten durch den Administrator bereit.

Fehlermanagement und Persistenzsicherung

Module für das Fehlermanagement und die Persistenzsicherung gibt es in DataGate^{TM} nicht. Fehlerhafte Nachrichten werden für alle Kommunikationsverbindungen in einer gemeinsamen Datei gesammelt, in der die Nachrichten in unverschlüsselter Form abgelegt werden. Die Persistenz der Nachrichten wird über das Prinzip *store- and-forward* erreicht. Dazu werden die Nachrichten zwischen den Subsystem-Agenten und dem Servermodul synchron übertragen. Hat das Servermodul eine Nachricht von einem Subsystem-Agenten erhalten, so erhält dieser erst dann eine positive Quittung, wenn

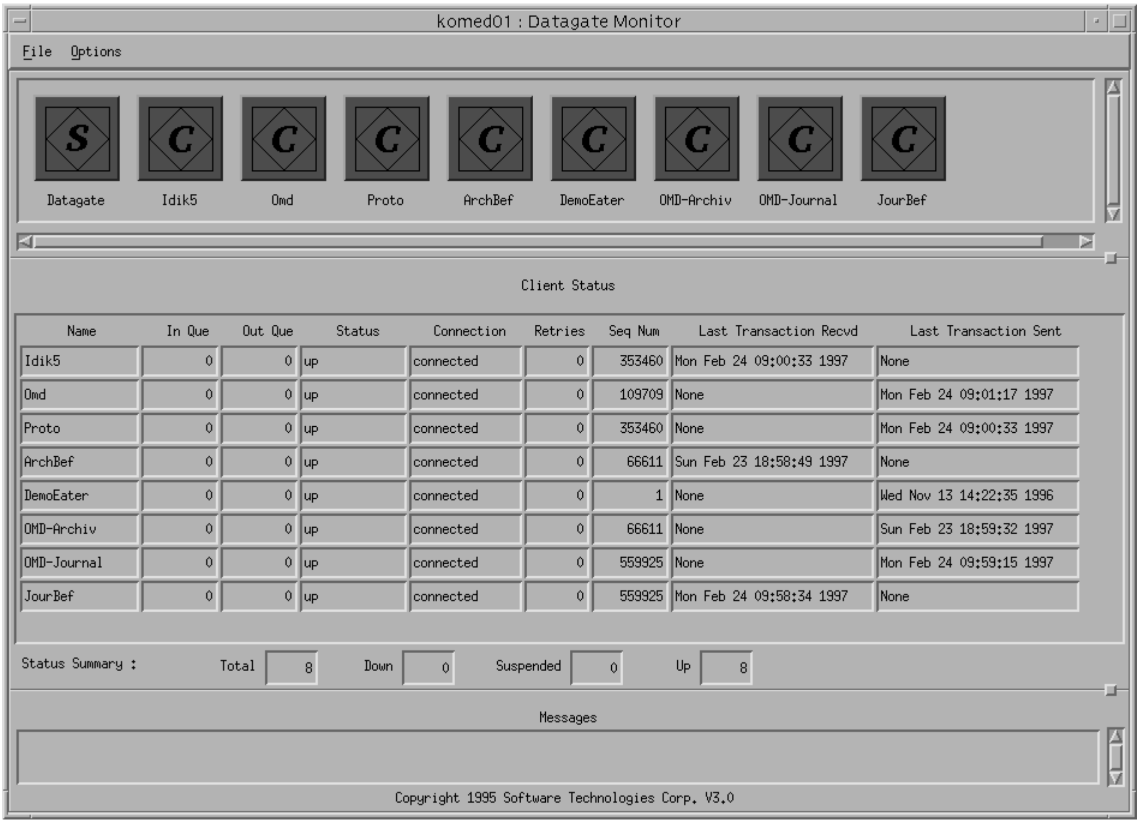


Abbildung 6.8: Abbildung der Benutzeroberfläche des System-Monitors von DataGate^{TM}. Eine Beschreibung des Monitors befindet sich im Text auf der Seite 62.

die Nachricht den Verarbeitungsprozeß durchlaufen hat und in die Warteschlange (Outbound-Queue) des Subsystem-Agenten des Empfängers eingereiht wurde. Eine Nachricht wird erst dann aus einer Outbound-Queue gelöscht, wenn der entsprechende Subsystem-Agent eine erfolgreiche Übertragung der Nachricht zum Empfänger angezeigt hat. Die Outbound-Queues werden auf einem externen Speichermedium gehalten, so daß die Persistenz der Nachrichten gewährleistet ist.

Datenschutz

Auch DataGate^{TM} stützt sich im Schutz der Daten vor einem unbefugten Zugriff vollständig auf die Sicherungsmaßnahmen des Betriebsystems ab. Weder die Log-Dateien noch die Nachrichtenarchive lassen sich in einer verschlüsselten Form erstellen. Auch der Nachrichtenaustausch zwischen den Modulen der Serverkomponente (Subsystem-Agenten und Servermodul), der über TCP-Sockets erfolgt, läßt sich nicht verschlüsseln. Für den Austausch der Nachrichten mit den Subsystemen lassen sich allerdings beliebige Sicherungsmaßnahmen in den Subsystem-Agenten implementieren.

DataGate^{TM} bietet für die Benutzer unterschiedliche Benutzerrollen an, die abgestufte Benutzerprofile zulassen. Von den Modulen ist allerdings nur der System-Monitor durch eine Paßwortabfrage vor einem unbefugten Zugriff geschützt.

Hardware- und Betriebsystemvoraussetzungen

DataGate^{TM} unterstützt zur Zeit die folgenden Hardware- und Betriebsystemplattformen:

Hardware	Betriebsystem
SUN (alle Modelle)	Solaris
IBM RS 6000	AIX 4.1.x
DEC Alpha	OSF
HP 9000	HP UX
SNI RM200/400/600	SINIX 5.1.x
Data General Aviion	DG UX
Intel x86	SVR4.x, Windows NT

Änderungen in der Version 3.1

Mit der DataGate™ Version 3.1 hat sich an der Architektur und dem Nachrichtenfluß nichts geändert. Stark verändert wurde aber die Darstellung der Konfigurationstabellen. Hier wird nun eine neue Beschreibungssprache (*Message Language* MONK) verwendet, die sich stark an Lisp anlehnt. Für die Definition der Identifikations- und Transformationsregeln sind nun spezielle graphische Editoren vorhanden. Auch lassen sich die Nachrichtenstrukturen über einen speziellen Editor im Kommunikationsserver definieren. Identifikations- und Transformationsregeln können über diese definierten Strukturen auf die Datenfelder der Nachrichten zugreifen.

6.1.3 HCS™

Das *Hospital-Communication-System* (HCS) war ursprünglich ein Gemeinschaftsprojekt der Anstalt für kommunale Datenverarbeitung in Bayern (AKDB), Hewlett-Packard Deutschland (HP) und der Gesellschaft für Softwareentwicklung mbH (SoftCon) in Oberhachingen. Der eigentliche Entwickler war und ist die Firma SoftCon. HP zog sich 1995 aus dem Vertrieb von HCS™ zurück, so daß der Kommunikationsserver nun von SoftCon und der AKDB allein weiter entwickelt wird.

Die Darstellung dieses Kommunikationsservers bezieht sich auf den vom Hersteller ausgefüllten Fragebogen sowie auf das Handbuch zur Version 1.1 [Sof96]. Für die Darstellung der Bildschirmmasken stellte der Hersteller des Kommunikationsservers *Sreendumps* zur Verfügung. Die Abbildung entsprechen denen in der Benutzeranleitung [Sof96].

Architektur der Serverkomponente

Die Funktionen der Serverkomponente werden bei dem Kommunikationsserver HCS™ auf zwei verschiedene Module verteilt. Sie werden als *HCS-Local* (HCS-L) und *HCS-Communication* (HCS-C) bezeichnet.

HCS-Local: Das Modul HCS-Local entspricht dem Verarbeitungsmodul eines Kommunikationsservers. Es übernimmt alle Schritte des Verarbeitungsprozesses (Identifikation, Routing und Transformation) der Nachrichten. Von diesem Modul können mehrere Instanzen unterhalten werden, so daß eine verteilte Architektur mit getrennten Verarbeitungswegen aufgebaut werden kann.

HCS™ verwendet HL7 als internes Kommunikationsprotokoll. Daß heißt, daß alle empfangenen Nachrichten in die HL7-Nachrichtenstrukturen abgebildet werden müssen. Nach dem Routing erfolgt dann eine umgekehrte Abbildung der HL7-Nachrichten in die Kommunikationsprotokolle der Empfänger (vgl. auch Abschnitt 3.5).

HCS-Communication: Der Name dieses Moduls verrät auch schon seine Aufgabe: Ein HCS-C Modul entspricht einem Subsystem-Agenten. Jedem Subsystem wird genau ein HCS-C Modul zugeordnet, über die der Nachrichtenaustausch in beide Richtungen (bidirektional) erfolgt. Jedes

HCS-C Modul ist wiederum genau einem HCS-L Modul zugeordnet, innerhalb dessen die Verarbeitung der Nachrichten stattfindet. Die Übertragung der Nachrichten zwischen den HCS-C Modulen und den Subsystemen kann dateibasiert über Dateien oder Pipes erfolgen. Für die dateilose Übertragung werden TCP-Sockets unterstützt.

Abbildung 6.9 zeigt ein Modell der Serverkomponente von HCS^{TM} in OMT-Notation.

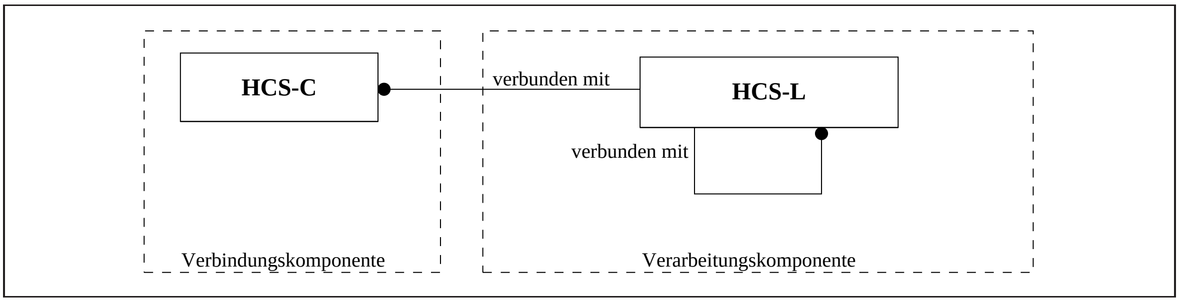


Abbildung 6.9: Darstellung der Serverkomponente von HCS^{TM} in OMT-Notation.

Die Module des Kommunikationsservers lassen sich beliebig auf verschiedene Rechner verteilen. In der Regel werden die Subsystem-Agenten direkt auf den Plattformen der Subsysteme installiert. Die Verteilung der Verarbeitungsmodule ermöglicht eine echt-parallele Verarbeitung der Nachrichten.

Nachrichtenfluß

Der Nachrichtenfluß über die einzelnen Module wird in Abbildung 6.10 gezeigt. Die Subsystem-Agenten (HCS-C Module) nehmen die Nachrichten von den Subsystemen entgegen und senden sie über RPC's oder TCP-Sockets unverändert an ein Verarbeitungsmodul (HCS-L Modul). Dort erfolgt die Transformation der Nachrichten in das interne Kommunikationsprotokoll (HL7). Die anschließende Identifikation der Nachrichten geschieht über die im Standard von HL7 festgelegten Nachrichtentypen. Die Empfänger einer Nachricht werden ebenfalls über die im Standard vorgesehenen Datenfelder bestimmt.

Sind die Subsystem-Agenten der Empfänger ebenfalls mit demselben Verarbeitungsmodul verbunden, wie die der Sender, werden nun die Nachrichten aus dem HL7-Format in die Kommunikationsprotokolle der Empfänger transformiert und an die entsprechenden Subsystem-Agenten übermittelt. Sind die Subsystem-Agenten der Empfänger aber mit anderen Verarbeitungsmodulen verbunden, werden die Nachrichten zuerst an die entsprechenden Module versandt. Dort erfolgt dann die Transformation und die Übermittlung an die zuständigen Subsystem-Agenten der Empfänger.

Nur wenn fehlerhafte Nachrichten vorliegen oder für eine Kommunikationsverbindung das „Mithören“ (siehe System-Monitor, Abbildung 6.12) aktiviert wurde, werden Kopien der ausgetauschten Nachrichten auch an das HCS-M Modul (siehe nächsten Abschnitt) übermittelt. Dort werden sie in der relationalen Datenbank gespeichert. In der Darstellung des Nachrichtenflusses über die Module von HCS^{TM} (Abbildung 6.10) wurde das „Mithören“ für die Kommunikationsverbindung zum Subsystem C aktiviert. Entsprechend werden alle Nachrichten, die von diesem Subsystem stammen bzw. für dieses Subsystem bestimmt sind, von dem zuständigen Verarbeitungsmodul an das HCS-M Modul übertragen.

Konfigurierung und Überwachung der Kommunikationsverbindungen

Für die Konfigurierung und die Überwachung der Kommunikationsverbindungen stellt HCS^{TM} ein Modul mit der Bezeichnung **HCS-Main** (HCS-M) zur Verfügung. Dieses Modul wird innerhalb einer

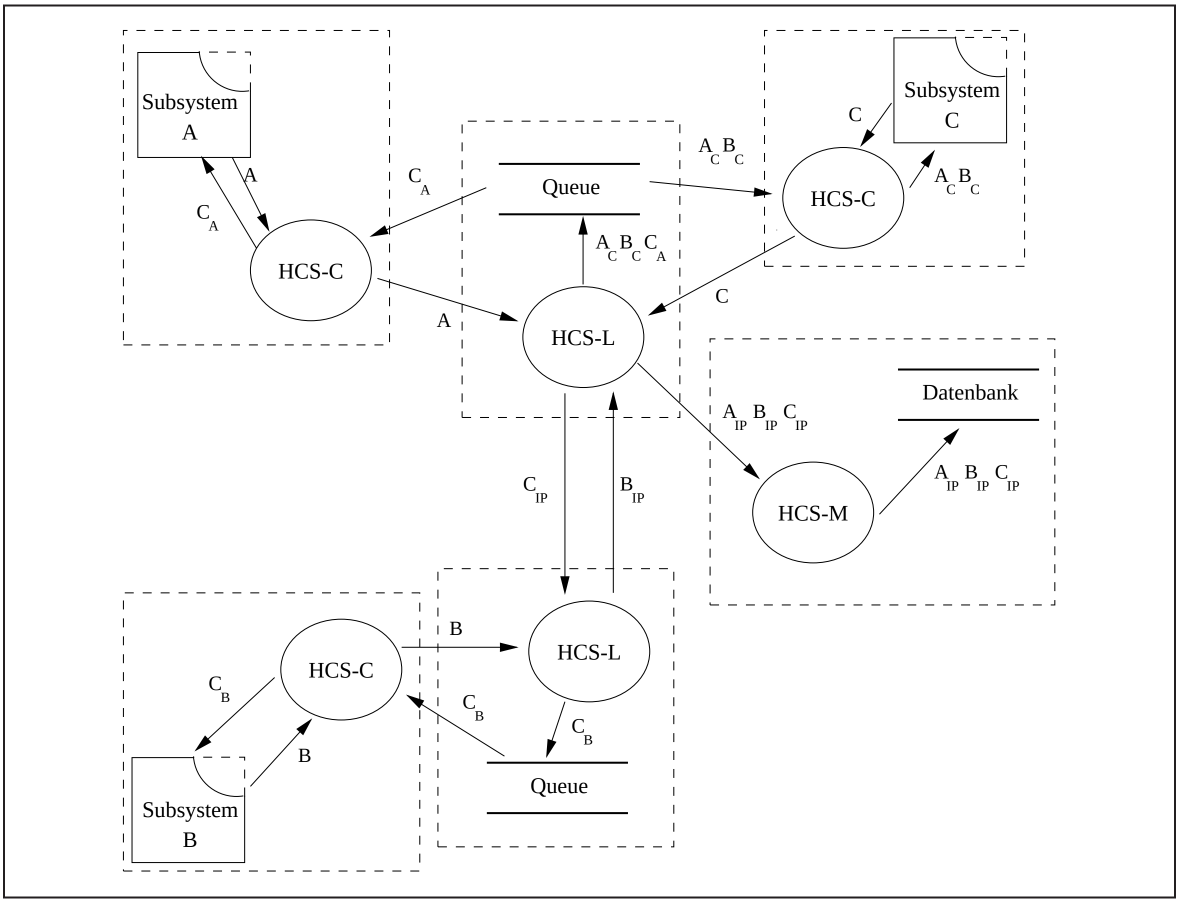


Abbildung 6.10: Darstellung des Nachrichtenflusses über die Module von HCS^{TM}.

Installation von HCS^{TM} genau einmal benötigt. Es stellt die graphische Benutzeroberfläche mit den Funktionen zur Steuerung der übrigen Module zur Verfügung. Die Steuerung des Nachrichtenflusses erfolgt in den entsprechenden Verarbeitungsmodulen und Subsystem-Agenten.

Die Konfigurierung der HCS-Module erfolgt über Konfigurationsdateien, die jeweils lokal bei den entsprechenden Modulen angelegt werden. Die Verwaltung der Konfigurationen erfolgt aber über das HCS-M Modul. Hier sind alle Konfigurationen nochmals in einer relationalen Datenbank (ORACLE^{TM} oder INFORMIX^{TM}) gespeichert. Für die Bearbeitung werden vom HCS-M Modul spezielle Editoren bereit gestellt. Wurden hier Konfigurationen geändert, so erzeugt das HCS-M Modul für die entsprechenden Verarbeitungsmodule oder Subsystem-Agenten aktualisierte Konfigurationsdateien und übermittelt diese an die entsprechenden Module, die dann die neuen Konfigurationen übernehmen können.

Die Empfänger einer Nachricht werden aus den entsprechenden Feldern im `MSH`-Segment der Nachricht entnommen (alle Nachrichten liegen intern als HL7-Nachrichten vor). Zur Kontrolle und Steuerung des Routings kann der Administrator im HCS-M Modul festlegen, welche Subsysteme welche Nachrichten untereinander austauschen dürfen. Der hierfür bereitgestellte Editor wird in Abbildung 6.11 dargestellt.

In diesem Editor lassen sich für jedes Subsystem in den oberen beiden Listen jeweils eine Nachricht und ein potentieller Empfänger auswählen. Im unteren Teil der Maske kann über das Auswahlfeld „`Status`“ bestimmt werden, ob die Nachricht an diesen Empfänger übermittelt werden darf oder nicht. Der untere Teil der Maske dient dem „Mithören“ auf einer Kommunikationsverbindung. Für

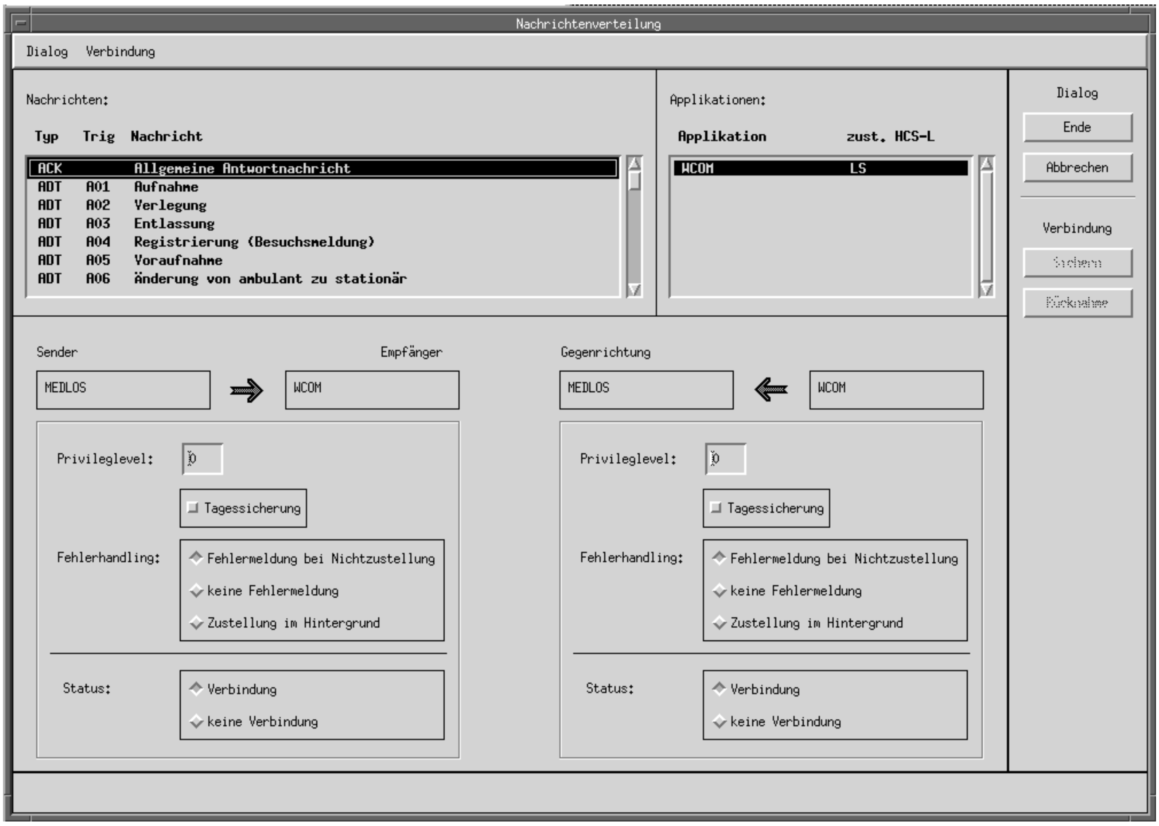


Abbildung 6.11: Darstellung der Benutzeroberfläche des Routing-Editors von HCS^{TM}. Eine Beschreibung befindet sich im Text auf der Seite 67. Die Abbildung ist dem Handbuch [Sof96] entnommen.

das „Mithören“ werden jeweils Kopien der versandten Nachrichten von den Verarbeitungsmodulen an das HCS-M Modul geschickt, wo sie in der Datenbank abgelegt und in der Maske angezeigt werden.

Das HCS-M Modul stellt auch die Funktionen eines System-Monitors bereit. Allerdings sind die Funktionen dabei auf mehrere unterschiedliche Bildschirm-Masken verteilt. Eine graphische Darstellung der Kommunikationsverbindungen existiert zur Zeit noch nicht. Die Verbindungen werden in Form von Listen dargestellt, in den die an der Verbindung beteiligten HCS-Module und Subsysteme aufgelistet werden (siehe Abbildung 6.12). Dazu werden in einer Liste alle verfügbaren HCS-L Module (Verarbeitungsmodule) angezeigt. Wird hier ein Modul selektiert, erscheinen in einer weiteren Liste alle diesem Verarbeitungsmodul zugeordneten Subsystem-Agenten. Zusätzlich werden die mit diesen Subsystem-Agenten verbundenen Subsysteme und der Status der Verbindung („aktiv“ oder „abgemeldet“) angezeigt.

Für die Überwachung der Kommunikationsverbindungen werden Log-Dateien und Nachrichtenarchive in Form von ASCII-Dateien jeweils auf den Rechnern derVerarbeitungsmodule geführt. In den Nachrichtenarchiven werden alle Nachrichten abgelegt, die von den angeschlossenen Subsystem-Agenten eingelesen werden. Die Speicherung erfolgt in einer unverschlüsselten Form. Jeweils um Mitternacht werden die dann 24 Stunden alten Nachrichten aus den Archiven gelöscht. Für die Verwaltung der Nachrichten in diesen Archiven durch den Administrator stellt das HCS-M Modul Funktionen über eine graphische Oberfläche bereit.

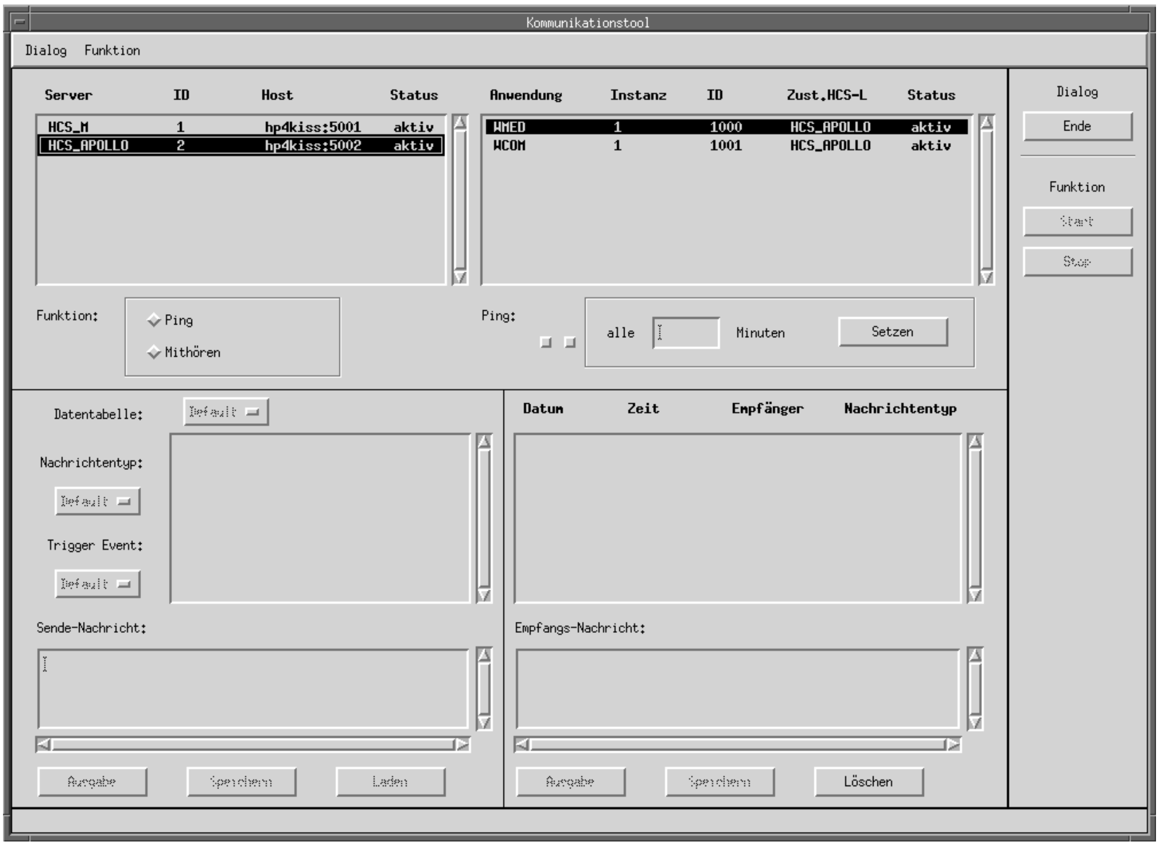


Abbildung 6.12: Darstellung der Benutzeroberfläche des System-Monitors von HCS^{TM}. Eine Beschreibung der Darstellung befindet sich im Text auf der Seite 68. Die Abbildung ist dem Handbuch [Sof96] entnommen.

Fehlermanagement und Persistenzsicherung

Alle fehlerhaften Nachrichten, die in den Verarbeitungsmodulen anfallen, werden zum HCS-M Modul gesandt, wo sie in der Datenbank gespeichert werden. Der Administrator kann über das HCS-M Modul auf diese Nachrichten zugreifen und sie gegebenenfalls modifizieren und erneut versenden.

Die Übertragung der Nachrichten über die Subsystem-Agenten und Verarbeitungsmodule erfolgt nach dem Prinzip *store-and-forward*. Nachrichten, die nicht an ein Subsystem versandt werden können (z.B. wegen einer unterbrochenen Verbindung), werden im entsprechenden Verarbeitungsmodul in einer Queue zwischengespeichert. Die Übertragung der Nachricht erfolgt, sobald die Verbindung zum Subsystem wieder hergestellt wurde. Konnte die Nachricht erfolgreich übertragen werden, so wird sie aus der Queue entfernt. Mit dieser Queue wird die Persistenz der Nachrichten im Kommunikationsserver gesichert. Konnte eine Nachricht nicht zugestellt werden, kann zusätzlich der Sender über eine Acknowledgement-Nachricht von diesem Sachverhalt informiert werden.

Datenschutz

Auch HCS^{TM} setzt in Bezug auf den Schutz der Nachrichten vor unbefugtem Zugriff keine neuen Akzente. Der Schutz wird vollständig an das Betriebsystem bzw. das DBMS übertragen. Der Austausch der Nachrichten zwischen den Modulen kann nur unverschlüsselt erfolgen. Da hier aber DCE/RCP's verwendet werden, sind die Nachrichten relativ gut geschützt. Auch der Nachrichtenaustausch zwischen den Subsystem-Agenten und den Subsystemen erfolgt unverschlüsselt. Ab HCS^{TM} Version 1.1

besteht aber die Möglichkeit, die Nachrichten vor dem Senden bzw. unmittelbar nach dem Empfang in den Subsystem-Agenten durch externe Prozesse bearbeiten zu lassen. Inwieweit hier eine Ver- bzw. Entschlüsselung der Nachrichten erfolgen kann, konnte im Rahmen dieser Arbeit nicht geklärt werden.

Hardware- und Betriebsystemvoraussetzungen

Die Installation von HCS^{TM} kann auf den folgenden Hardware- und Betriebsystemplattformen erfolgen:

Hardware	**Betriebsystem**
HP 9000	HP-UX 9/10
Siemens Nixdorf RM400/600	SINIX 5.4

Weitere Voraussetzungen für die Installation:

- OSF/Motif
- DCE V 1.2 (muß auf allen Clients installiert sein, wenn die Übertragung der Nachrichten nicht per TCP/IP erfolgen soll)
- Oracle 7 oder Informix 7

6.1.4 HL7-Connection-Server^{TM}

Der HL7-Connection-Server^{TM} wurde von der **`prompt!`** Medizinische Informationssysteme GmbH entwickelt. **`prompt!`** ist durch ein Joint Venture aus den Geschäftsbereichen „Gesundheitswesen“ der Partner-Consult GmbH Hamburg, und dem DV-Bereich „Medis“ der Firma Philips Medizin Systeme hervorgegangen.

Die hier vorgestellte Kurzbeschreibung des HL7-Connection-Server^{TM} stützt sich auf das umfangreiche Informationsmaterial [pro95], das vom Hersteller zur Verfügung gestellt wurde. Die in der Beschreibung dargestellten Bildschirmmasken sind diesem Informationsmaterial entnommen.

Architektur der Serverkomponente

Das Servermodul, das die Funktionen der Serverkomponente übernimmt, setzt sich aus drei verschiedenen Prozessen zusammen:

1. Der sogenannte **Daemon** empfängt die Nachrichten von den Subsystemen. Dieser Prozeß existiert im System nur einmal und ist für den Empfang der Nachrichten von allen angeschlossenen Subsystemen zuständig. Der *Daemon* übernimmt somit die Aufgaben eines Subsystem-Agenten, der für alle Subsysteme zuständig ist.

2. Für jede Nachricht, die der *Daemon* erhält, startet er einen sogenannten **Empfangsprozeß**, der für die Verarbeitung der Nachricht zuständig ist. Der HL7-Connection-Server^{TM} unterstützt ausschließlich HL7 als Kommunikationsprotokoll. Alle Nachrichten müssen in dieser Form vorliegen. Nachrichten, die in proprietären Kommunikationsprotokollen vorliegen, müssen gegebenenfalls in externen Schnittstellen in das HL7-Format umgewandelt werden. Somit entfällt für die Nachrichtenverarbeitung die Transformationen der Nachrichtenstrukturen. Es werden lediglich Transformationen zwischen den HL7-Versionen und eine Schlüsselkonvertierung einzelner HL7-Felder (z.B. Anpassung der Codierung der Angabe des Geschlechts eines Patienten) durchgeführt.

3. Der Empfangsprozeß übergibt die Nachricht einem sogenannten **Sendeprozeß**. Dieser übernimmt die Übertragung der Nachricht an den oder die Empfänger. Auch dieser Prozeß übernimmt somit die Aufgaben eines Subsystem-Agenten, der für mehrere Subsysteme zuständig ist.

Die Aufteilung des Servermoduls in die drei Prozesse wird in OMT-Notation in Abbildung 6.13 dargestellt.

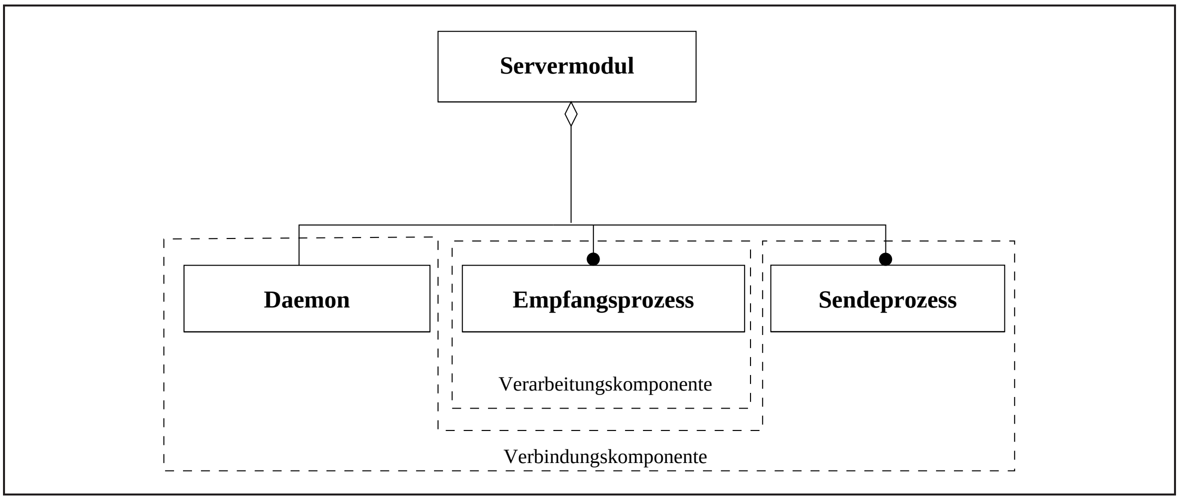


Abbildung 6.13: Darstellung der Serverkomponente des HL7-Connection-Server^{TM} in OMT-Notation. Die einzelnen Prozesse werden durch die gestrichelt gezeichneten Kästen der Verarbeitungs- und Verbindungskomponente zugeordnet.

Nachrichtenfluß

Der Nachrichtenfluß über den HL7-Connection-Server^{TM} wird in Abbildung 6.14 dargestellt. Erscheint eine neue Nachricht im Kommunikationsserver, so wird vom *Daemon* ein neuer Empfangsprozeß generiert und die Nachricht von ihm an diesen Prozeß übergeben. Dort erfolgt die Verarbeitung der Nachricht, also die Nachrichtenidentifikation, das Routing und gegebenenfalls die Transformation in eine andere HL7-Version bzw. die Schlüsselkonvertierung einzelner HL7-Felder. Das Routing erfolgt über die im HL7-Standard definierten Nachrichtentypen anhand von Routing-Tabellen. Die Übermittlung der Nachricht an die Empfänger wird von einem Sendeprozeß übernommen. Empfangs- und Sendeprozeß speichern nach jedem Verarbeitungsschritt die Nachrichten in einer relationalen Datenbank.

Die Empfangs- und Sendeprozesse einer Nachricht existieren nur solange, bis die Nachricht erfolgreich an alle Empfänger übermittelt werden konnte. Für jede Nachricht werden eigene Empfangs- und Sendeprozesse generiert.

Die Übertragung der Nachrichten werden im HL7-Connection-Server^{TM} in Form der im Abschnitt 4.2.2 dargestellten komplexen Aktionen durchgeführt. Diese komplexen Aktionen werden hier als *Transaktionen* bezeichnet. Die einzelnen komplexen Aktionen sind weitestgehend beliebig definierbar, die Implementierung kann aber nur vom Hersteller vorgenommen werden.

Konfigurierung und Überwachung der Kommunikationsverbindungen

Alle Konfigurationen des Kommunikationsservers werden in einer relationalen Datenbank gespeichert. Das Konfigurationsmodul (als *DB-Edit* bezeichnet) stellt Bildschirmmasken für die Definition und

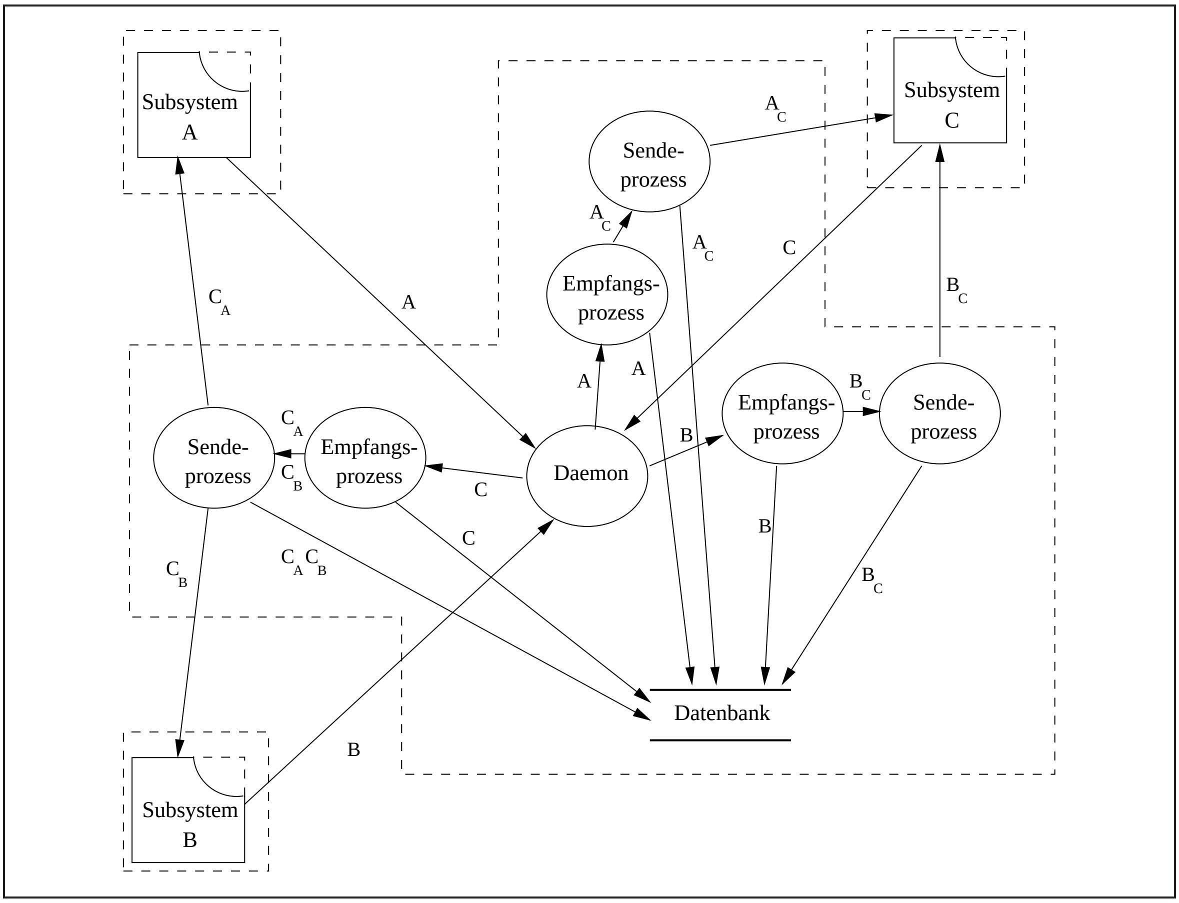


Abbildung 6.14: Darstellung des Nachrichtenflusses über die Module des HL7-Connection-Server^{TM}. Die Prozesse „Empfangsprozeß" sowie „Sendeprozeß" sind als temporäre Prozesse aufzufassen, die beim Eintreffen einer Nachricht vom *Daemon* erzeugt werden und nur für die Dauer der Verarbeitung dieser Nachricht existieren.

Änderung der einzelnen Tabellen bereit.

Da HL7 ein Multicasting einzelner Nachrichten nicht vorsieht, die komplexen Aktionen aber auf Multicasting angewiesen sind, werden die Empfänger einer Nachricht nur über den Nachrichtentyp bestimmt. Für jedes Subsystem lassen sich für jeden Nachrichtentyp die Empfänger der Nachrichten angeben (siehe Abbildung 6.15).

Alle Kommunikationsprozesse werden in Form von Transaktionsprotokollen in der Datenbank gespeichert. In diesen Protokollen werden für jede Nachricht der Sender bzw. Empfänger, der Nachrichtentyp, Datum und Uhrzeit, sowie der Übertragungsstatus („OK" oder „Error") angezeigt. Der System-Monitor des HL7-Connection-Server^{TM} erlaubt dem Administrator sämtliche Fehlermeldungen und Transaktionsprotokolle einzusehen. Dabei kann auch auf den Inhalt der einzelnen Nachrichten zugegriffen werden. In Abbildung 6.16 ist der System-Monitor des HL7-Connection-Server^{TM} abgebildet.

Ein graphische Darstellung der Kommunikationsverbindungen zwischen den Subsystemen ist im System-Monitor nicht möglich. Diese Informationen lassen sich lediglich mit Hilfe des entsprechenden Editors des Konfigurationsmoduls (siehe Abbildung 6.15) mühsam ermitteln.

Abbildung 6.15: Darstellung der Benutzeroberfläche des Routing-Editors des HL7-Connection-Server^{TM}. Es werden jeweils für Sender (im Textfeld `MODUL_ID` angegeben) und Empfänger (Textfeld `MODUL_ID1`) getrennt der Nachrichtentyp (Textfeld `EVENT_ID`), die HL7-Versionsnummer (`VERSION_ID`) sowie das Nachrichtenverarbeitungskennzeichen (`Prozessing_ID`) angegeben. Die Abbildung wurde aus [pro95] entnommen.

Fehlermanagement und Persistenzsicherung

Es konnte im Rahmen dieser Arbeit nicht ermittelt werden, ob der HL7-Connection-Server^{TM} über ein Fehlermanagement von fehlerhaften Nachrichten verfügt. Grundsätzlich werden aber alle Nachrichten in der angeschlossenen Datenbank gespeichert. Fehler in der Nachrichtenübertragung werden vom Daemon bzw. den Sendeprozessen abgefangen.

Das Konzept der *Transaktionen* (komplexe Aktionen), das für die Übertragung der Nachrichten verwendet wird, garantiert eine Persistenz der Nachrichten im Kommunikationsserver. Sobald eine Nachricht empfangen und an den Empfangsprozeß weitergeleitet wurde, wird diese in der Datenbank gespeichert (vgl. Abbildung 6.14). Erst dann erfolgt eine Bestätigung des Empfangs an das sendende Subsystem. Das Ende einer komplexen Aktion wird durch die Bestätigung des Empfängers der Nachrichten angezeigt. Bei einem Systemausfall werden auf die Transaktionssicherungsmechanismen des verwendeten DBMS zurückgegriffen. Was sich hinter diesen Sicherungsmaßnahmen verbirgt, konnte nicht ermittelt werden. Unklar blieb auch, wann die Nachrichten, die zu einer Transaktion gehören, wieder aus der Datenbank gelöscht werden.

Datenschutz

Eine Verschlüsselung der Nachrichten für die Übertragung zwischen den Subsystemen und dem Kommunikationsserver ist nicht möglich. Hier verläßt sich der HL7-Connection-Server^{TM} vollständig auf die Sicherungsmaßnahmen des Betrieb- und Netzwerksystems. Allerdings werden die Nachrichten im Kommunikationsserver nur innerhalb der Datenbank gespeichert, wo sie zusätzlich durch das DBMS geschützt werden.

Inwieweit für die einzelnen Module des Kommunikationsservers ein Zugriffsschutz besteht, konnte im Rahmen dieser Arbeit nicht ermittelt werden. Aus den vorliegenden Informationen konnte aber

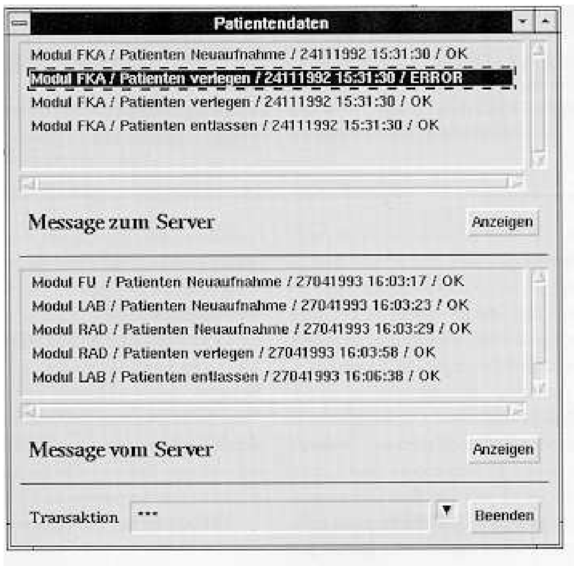


Abbildung 6.16: Darstellung der Benutzeroberfläche des System-Monitors des HL7-CONNECTION-SERVER™. Eine graphische Darstellung der Kommunikationsverbindungen ist nicht möglich. Angezeigt werden lediglich die über den Kommunikationsserver ausgetauschten Nachrichten. Die Abbildung wurde aus [pro95] entnommen.

entnommen werden, daß den Benutzern des Kommunikationsservers unterschiedliche Befugnisse in Form von Benutzerrollen zugeteilt werden können.

Hardware- und Betriebsystemvoraussetzungen

Der HL7-CONNECTION-SERVER™ unterstützt zur Zeit die folgenden Hardware- und Betriebsystemplattformen. Weitere Portierung sind auf Anfrage erhältlich.

Hardware	Betriebsystem
DEC 5000/240	Ultrix 4.2
SUN SparcServer	Solaris
Siemens RM 400	Sinix

Für die Datenhaltung innerhalb des Kommunikationsservers werden die folgenden relationalen DBMS unterstützt:

- Ingres 6.4
- Oracle 7
- Infomix

Auch hier sind weitere Portierungen auf Anfrage erhältlich.

6.1.5 pro7-System™

Das PRO7-SYSTEM™ stellt das jüngste Produkt in dieser Reihe der Kommunikationsserver dar. Er ist aus einem Gemeinschaftsprojekt der Abteilung Softwaretechnik des ICD e.V., der PRO DV Software

GmbH und der ROKD GmbH entstanden. Eine erste Version wird im Laufe dieses Jahres verfügbar sein. Die hier vorgestellte Kurzbeschreibung stützt sich auf den beim Hersteller ausgefüllten Fragebogen sowie auf [Jun95] und [GS97].

Architektur der Serverkomponente

Die Funktionen der Serverkomponente werden durch drei verschiedene Prozesse ausgeführt:

1. Für den Anschluß der Subsysteme an den Kommunikationsserver sind die sogenannten **Adapter** zuständig. Sie entsprechen in ihrer Funktionalität den Subsystem-Agenten und werden für jedes Subsystem in der Regel individuell entwickelt. Die Implementierung erfolgt in der Programmiersprache C. Für die Schnittstelle zum Kommunikationsserver wird lediglich ein Code-Gerüst bereitgestellt. Die Implementierung der Schnittstelle zum Subsystem muß komplett programmiert werden. Für bestimmte Schnittstellen stehen fertige Subsystem-Agenten zur Verfügung, die individuell angepaßt werden können.
2. Das PRO7-SYSTEM^{TM} verwendet eine Erweiterung von HL7 als internes Kommunikationsprotokoll, die als *pro7* bezeichnet wird. Dieses Kommunikationsprotokoll berücksichtigt sowohl die deutschen Anpassungen von HL7 als auch die Möglichkeit, beliebige Daten wie zum Beispiel Röntgenbilder auszutauschen. Die notwendigen Transformationen der Nachrichten in das interne Kommunikationsprotokoll werden durch die sogenannten **Konverter** durchgeführt. Diese sind bidirektional ausgelegt, führen also auch die Transformation einer HL7-Nachricht in das Kommunikationsprotokoll des Empfängers durch. Auch die Konverter müssen vom Anwender programmiert werden. Hier erfolgt die Implementierung in einer speziellen Programmiersprache, die als *Protocol Definition Language* (PDL) bezeichnet wird. Für die Generierung der Konverter steht ein spezieller Compiler zur Verfügung.
3. Der **pro7-Kern** stellt den zentralen Prozeß in der Nachrichtenverarbeitung dar. Hier erfolgt die Nachrichtenidentifikation und das Routing. Innerhalb des pro7-Kerns liegen alle Nachrichten im internen Kommunikationsprotokoll vor. Die Strukturtransformationen werden in den Konvertern durchgeführt. Im pro7-Kern kann eine Schlüsseltransformation einzelner HL7-Felder durchgeführt werden (z.B. Konvertierung des Schlüssels für die Codierung des Geschlechts eines Patienten).

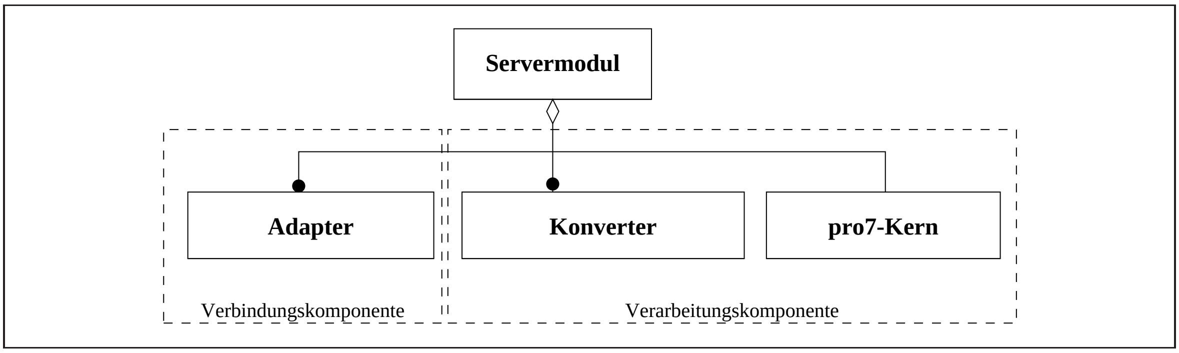


Abbildung 6.17: Darstellung der Serverkomponente des PRO7-SYSTEM^{TM} in OMT-Notation. Die gestrichelten Kästen deuten die Zugehörigkeit der Prozesse zur Verarbeitungs- und Verbindungskomponente an.

Die Serverkomponente besteht somit aus genau einem pro7-Kern und einer variablen Anzahl an Adaptern und Konvertern. Zwischen diesen Modulen werden die Nachrichten über DCE/RPC's [Sch93] ausgetauscht. Ist ein Subsystem in der Lage, eine Kommunikation über DCE/RPC's durchzuführen, so kann dieses Subsystem auch ohne Adapter an den Kommunikationsserver angeschlossen werden. Verwendet das Subsystem darüberhinaus HL7 als Kommunikationsprotokoll, so kann auch der Konverter entfallen und die Nachrichten direkt an den pro7-Kern übermittelt werden. Abbildung 6.17 stellt die Beziehungen zwischen den Serverprozessen in OMT-Notation dar.

Die intermodulare Kommunikation über DCE/RPC's ermöglicht auch eine Verteilung der Module der Serverkomponente auf mehrere Rechner. Da die Kommunikation über DCE/RPC's auf der Grundlage des *Distributed Computing Environment* (DCE) der *Open Software Foundation* (OSF) realisert wurde, müssen bei einer Verteilung der Module alle Rechner DCE unterstützen. Auch bei einer Verteilung der Module kann aufgrund der zentralen Stellung des pro7-Kerns nur eine serielle Nachrichtenverarbeitung erfolgen.

Nachrichtenfluß

Der Nachrichtenfluß über den Kommunikationsserver gestaltet sich recht einfach (siehe Abbildung 6.18). Die Adapter (Subsystem-Agenten) nehmen die Nachrichten von den Subsystemen entgegen und übermitteln sie unverändert an die Konverter weiter. Dort erfolgt die Transformation der Nachrichten in das interne Kommunikationsprotokoll pro7. Die Konverter übergeben nach der erfolgreichen Konvertierung die Nachrichten an den pro7-Kern, der sie sofort in einer Datenbank speichert. Erst jetzt erfolgt über die Rückgabewerte der DCE/RPC's ein Acknowledgement für die Übertragung der Nachrichen, das bis zu den Adaptern durchgereicht wird. Die Übertragung der Nachrichten vom Subsystem zum Kommunikationsserver erfolgt somit synchron.

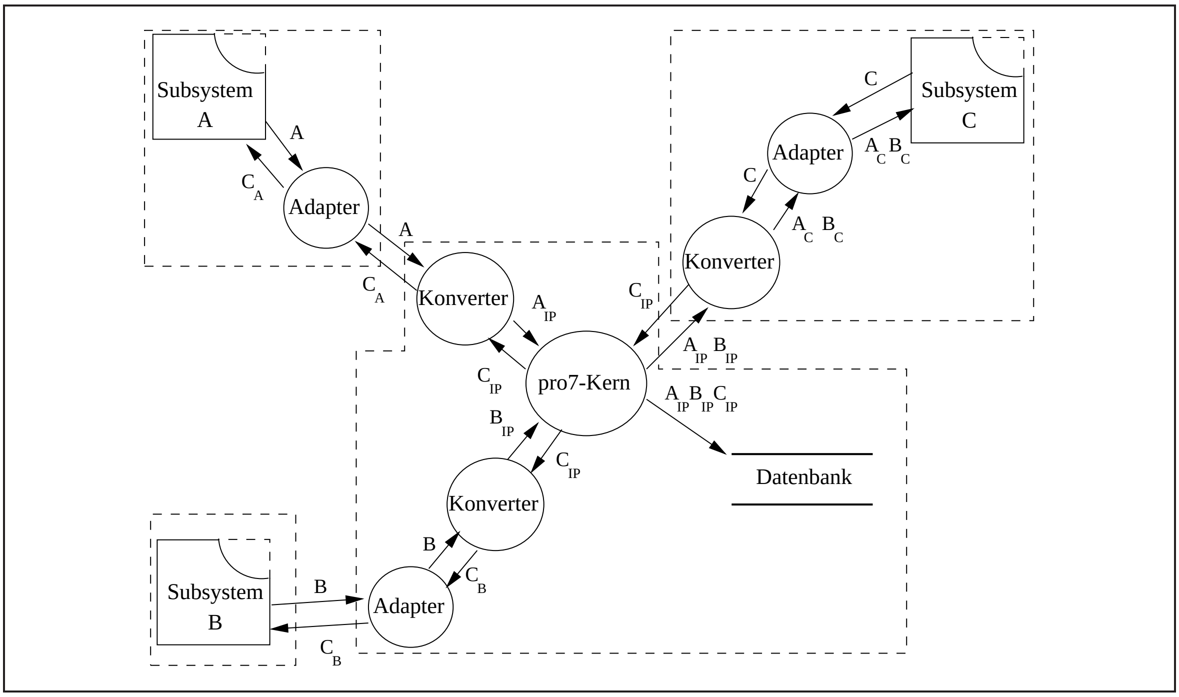


Abbildung 6.18: Darstellung des Nachrichtenflusses über den Kommunikationsserver pro7-System^{TM}. Das Datenflußdiagramm zeigt eine mögliche Verteilung der Module des Kommunikationsservers auf unterschiedliche Rechner.

Im pro7-Kern werden die Nachrichten geroutet und gegebenenfalls Schlüsselkonvertierungen einzelner Felder vorgenommen. Die Kopien der Nachrichten, die beim Routing erzeugt werden, stellen nun neue Nachrichten dar und werden ebenfalls in der Datenbank hinterlegt. Die Übertragung der Nachrichten zu den Empfängern erfolgt ebenfalls über die Konverter, in denen die Transformation in die Kommunikationsprotokolle der Empfänger stattfindet, und die Adapter, die die Nachrichten an die Subsysteme übergeben. Auch diese Übertragung erfolgt über die DCE/RPC's synchronisiert. Eine Synchronisation der Nachrichtenübertragung zwischen den Subsystemen kann über Acknowledgement-Nachrichten erfolgen, die in den Adaptern generiert bzw. ausgewertet werden.

Unter bestimmten Umständen können die Adapter und/oder die Konverter für den Anschluß eines Subsystems wegfallen. Ist das Subsystem, bzw. der entsprechende Kommunikationklient (vgl. Abschnitt 3.3), in der Lage, HL7-Nachrichten über DCE/RPC's zu übertragen, so läßt sich dieses Subsystem direkt an den pro7-Kern anschließen.

Konfigurierung und Überwachung der Kommunikationsverbindungen

Die Konfigurationen der Kommunikationsverbindungen werden im PRO7-SYSTEM™ in einer relationalen Datenbank gespeichert. Bisher wird `Oracle`™ als relationales DBMS unterstützt.

Da alle Nachrichten innerhalb des Kommunikationsservers im HL7-Format vorliegen, werden die Empfänger einer Nachricht über die im Standard definierten Datenfelder ermittelt. Um den Nachrichtenaustausch zwischen den Subsystemen kontrollieren zu können, werden im pro7-Kern für jedes Subsystem unterstützte und nicht unterstützte Nachrichten festgelegt. Ein Subsystem kann dann nur die von ihm unterstützen Nachrichten empfangen. Alle weiteren Nachrichten werden vom Kommunikationsserver zurückgewiesen bzw. nicht übermittelt.

Da im HL7-Standard ein Multicasting einzelner Nachrichten nicht vorgesehen ist, können die Nachrichten im pro7-Kern auch über den Nachrichtentyp geroutet werden. Für jedes Subsystem läßt sich eine Multicasting-Tabelle verwalten, in der für jeden unterstützten Nachrichtentyp die Empfänger eingetragen werden. Für einen Empfänger muß der entsprechende Nachrichtentyp als unterstützt definiert worden sein. Diese Tabellen entsprechen am ehesten den Routing-Tabellen der anderen Kommunikationsserver. Der Editor für die Bearbeitung dieser Tabellen wird in Abbildung 6.19 dargestellt.

Ein System-Monitor, der eine graphische Darstellung der Kommunikationsverbindungen ermöglicht, existiert im PRO7-SYSTEM™ nicht. Es werden lediglich die einzelnen angeschlossenen Subsysteme mit ihren Subsystem-Agenten (Adaptern) und Konvertern, sowie dem Verbindungsstatus tabellarisch dargestellt (siehe Abbildung 6.20 auf Seite 79).

Zur Kontrolle der Kommunikationsverbindungen werden alle Nachrichten in der relationalen Datenbank gespeichert. Log-Dateien im klassischen Sinne werden nicht erstellt. Die Informationen über den Verarbeitungsprozeß werden direkt mit den einzelnen Nachrichten verbunden und mit diesen gespeichert. Das pro7-Kommunikationsprotokoll sieht hierfür spezielle Datenfelder in den Nachrichtenstrukturen vor.

Fehlermanagement und Persistenzsicherung

Grundsätzlich werden alle Nachrichten in der relationalen Datenbank gespeichert. Fehlerhafte Nachrichten werden besonders markiert und nicht an die Empfänger ausgeliefert. Der Administrator kann über ein Modul diese Nachrichten einsehen, gegebenenfalls korrigieren und erneut versenden. Die Speicherung der Nachrichten in der Datenbank dient somit auch der Persistenzsicherung der Nachrichten. Zusätzlich werden die Nachrichten innerhalb des Kommunikationsservers über die DCE/RPC's nur synchron übertragen.

Die Nachrichten müssen in der Datenbank durch den Administrator explizit gelöscht werden. Auch dann, wenn eine Nachricht erfolgreich übermittelt werden konnte. Ist der Speicherplatz der Datenbank verbraucht, werden alle weiteren Nachrichten vom PRO7-SYSTEM™ zurückgewiesen.

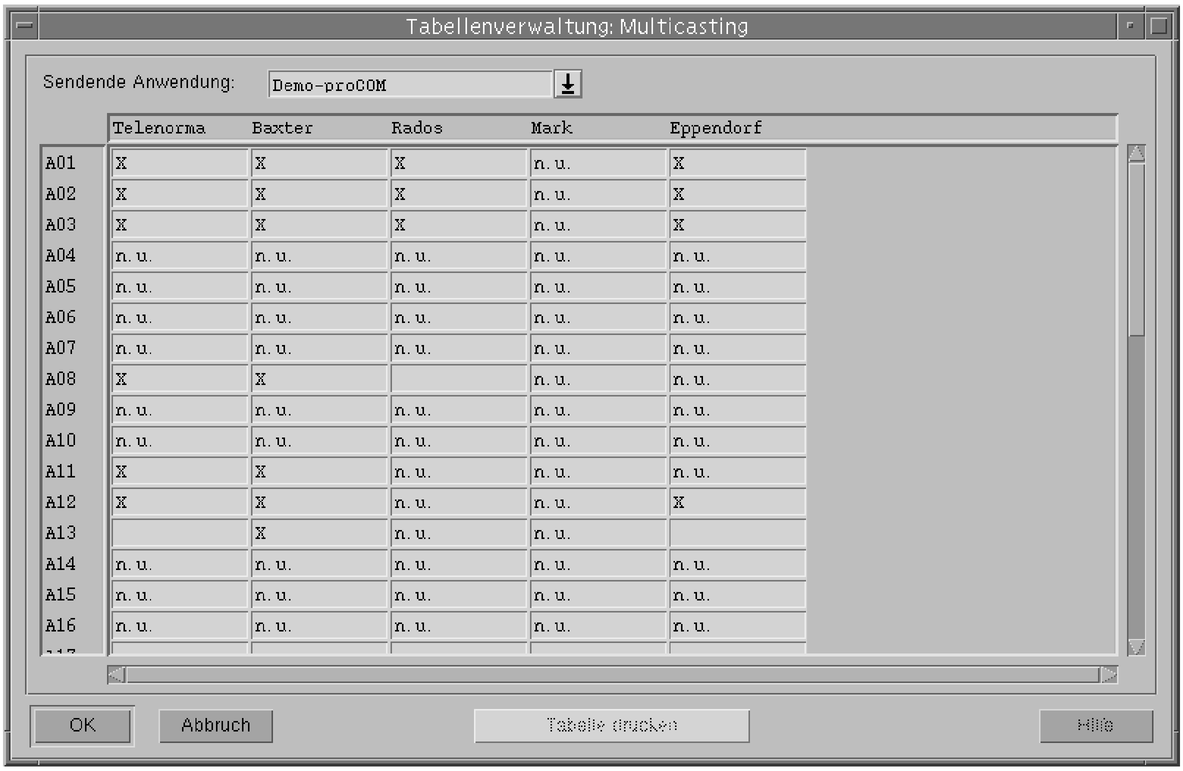


Abbildung 6.19: Darstellung der Benutzeroberfläche des Routing-Editors des PRO7-SYSTEM^{TM}. Für jedes Subsystem können hier die Empfänger der vom Subsystem unterstützten Nachrichten angegeben werden. Der Empfänger muß diesen Nachrichtentyp ebenfalls unterstützten.

Datenschutz

Die Nachrichten liegen innerhalb des Kommunikationsservers an keiner Stelle in einer direkt lesbaren Form vor (z.B. als ASCII-Datei). Innerhalb der Datenbank werden die Nachrichten durch das DBMS geschützt. Die für die Speicherung notwendigen Tabellen sind nur über die vom Kommunikationsserver angebotenen Schnittstellen einsehbar und veränderbar. Ein direkter Zugriff über das DBMS ist nicht möglich. Für die Übertragung der Nachrichten von und zu den Subsystemen lassen sich aufgrund der flexiblen Architektur der Subsystem-Agenten (Adapter) beliebige Sicherungsmechanismen implementieren.

Hardware- und Betriebsystemvoraussetzungen

Das PRO7-SYSTEM^{TM} benötigt die folgenden Hardware- und Betriebsystemvoraussetzungen:

Hardware	**Betriebsystem**
Sun Sparc-Station	Solaris

Weitere Portierungen sind auf Anfrage erhältlich. Voraussetzung ist allerdings eine für diese Systeme implementierte DCE-Version. Für die Installation ist desweiteren das relationale DBMS ORACLE^{TM} Voraussetzung.

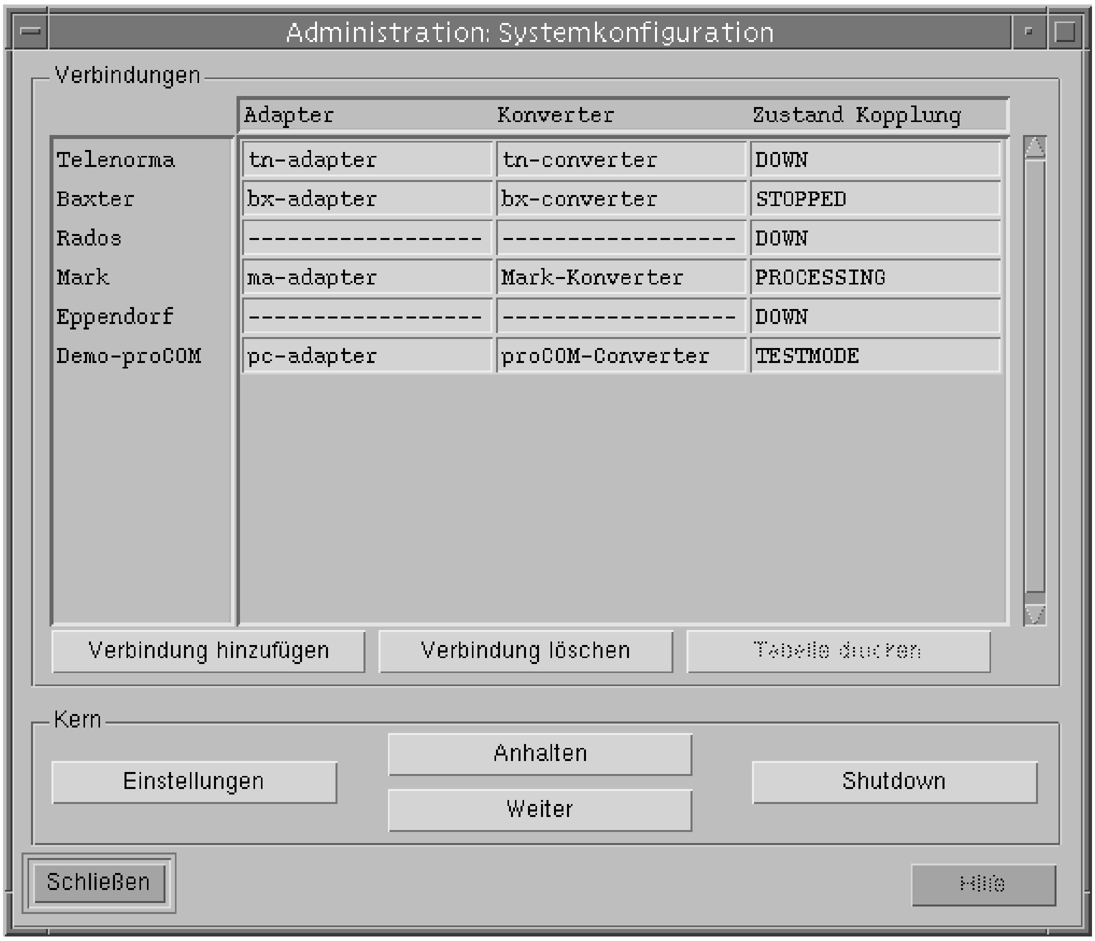


Abbildung 6.20: Darstellung der Benutzeroberfläche des System-Monitors des PRO7-SYSTEM^{TM}.

6.2 Einordnung der Kommunikationsserver in die Taxonomie

In diesem Abschnitt folgt nun die Einordnung der betrachteten Kommunikationsserver in die im Kapitel 5 entwickelte Taxonomie für Kommunikationsserver. Die Taxonomie mit ihren drei Dimensionen *Verteiltheit der Serverkomponente*, *Nebenläufigkeit der Nachrichtenverarbeitung* und *Heterogenität der Module* und der Einordnung der Kommunikationsserver wird in Abbildung 6.21 dargestellt.

Cloverleaf^{TM}

Die Serverkomponente von CLOVERLEAF^{TM} besteht aus einem oder mehreren Serverprozessen, die sich wiederum in *Protocol-Threads* (Subsystem-Agenten), einen *Translation-Thread* (Verarbeitungsmodul) und einen *Command-Thread* (Steuerungsmodul) aufspalten. Alle *Threads* und alle Serverprozesse müssen auf einem Rechner lokalisiert sein. Eine Verteilung auf mehrere Rechner ist nicht möglich. Da jeder Serverprozeß aber einen eigenständigen, vollständigen Verarbeitungsweg für Nachrichten bereitstellt, kann eine quasi-parallele Nachrichtenverarbeitung stattfinden. Somit ist CLOVERLEAF^{TM} in die Klasse der **zentralen, lokal-nebenläufigen Kommunikationsserver** einzuordnen.

DataGate^{TM}

Im Prinzip lassen sich die Subsystem-Agenten des Kommunikationsservers DATAGATE^{TM}, die *Communication Clients*, auf verschiedene Rechner verteilen. Allerdings müssen die Subsystem-Agenten dann selbständig um eine Aufrechterhaltung der Verbindung zum Servermodul sorgen.

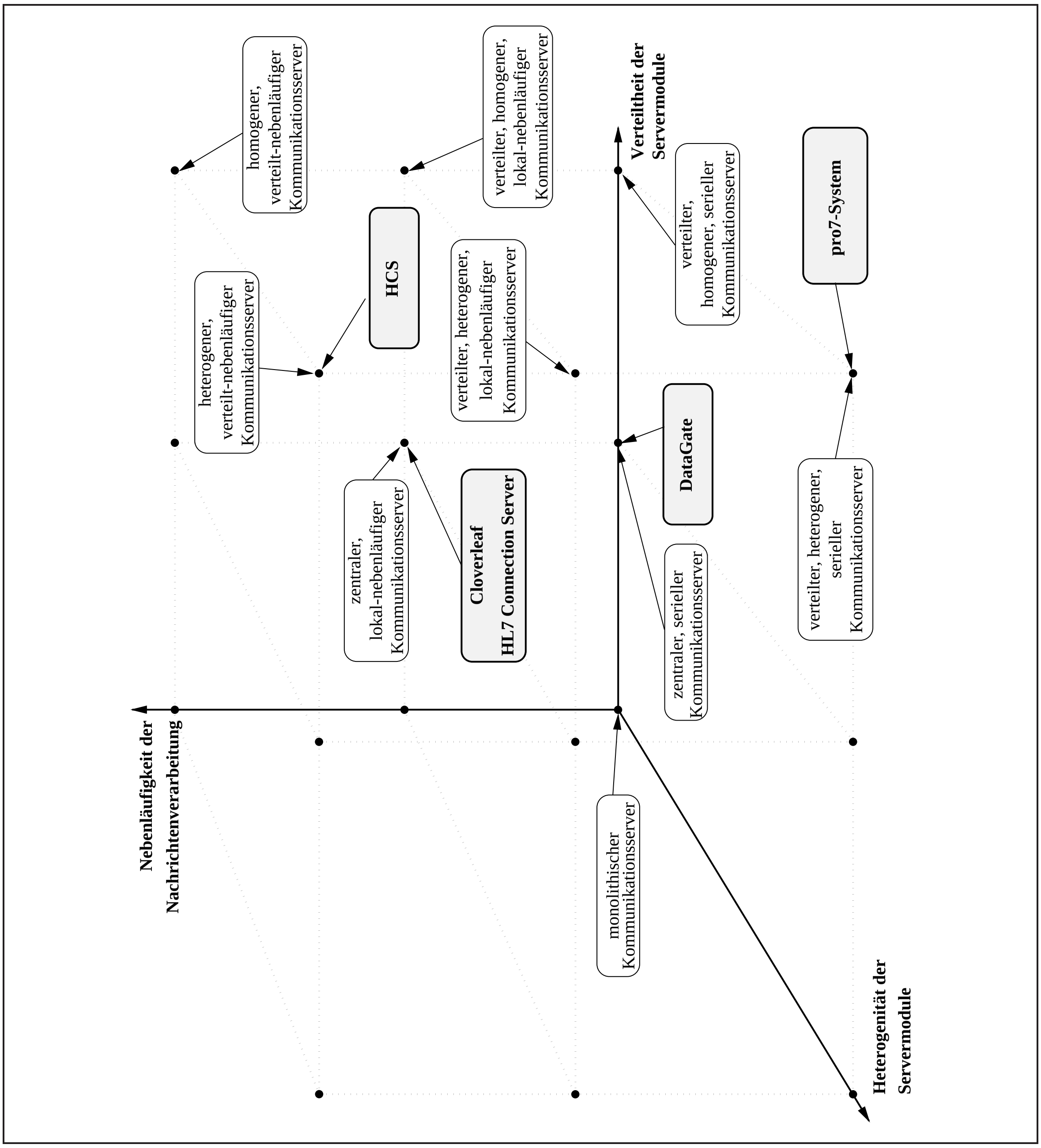


Abbildung 6.21: Einordnung der betrachteten Kommunikationsserver CLOVERLEAF^{TM}, DATAGATE^{TM}, HCS^{TM}, HL7-CONNECTION-SERVER^{TM} und PRO7-SYSTEM^{TM} in die Taxonomie für Kommunikationsserver.

Bei einer nicht-verteilten Anordnung wird dies vom Servermodul übernommen. Auch können bei einer verteilten Anordnung die Log-Dateien der Subsystem-Agenten vom System-Monitor aus nicht eingesehen werden. Aufgrund dieser Einschränkungen ist bei DataGate™ wohl eine nicht-verteilte Architektur der Serverkomponente vorzuziehen. Da für alle Nachrichten nur ein Verarbeitungsweg existiert (der zentrale Serverprozeß), ist DataGate™ somit in die Klasse der **zentralen, seriellen Kommunikationsserver** einzuordnen.

HCS™

Der Kommunikationserver HCS™ ist in die Klasse der **heterogenen, verteilt-nebenläufigen Kommunikationsserver** einzuordnen. Alle Module der Serverkomponente lassen sich auf die unterschiedlichsten Rechner verteilen. Im Prinzip läßt sich für jedes Subsystem auch ein eigener Nachrichtenverarbeitungsweg installieren, wenn für jeden Subsystem-Agenten (HCS-*Communication*) ein eigenes Verarbeitungsmodul (HCS-*Local*) vorgesehen wird. Eine parallele Nachrichtenverarbeitung ist somit möglich. Unter der Einschränkung, daß alle Rechner DCE/RPC's unterstützen, können die einzelnen Module auch auf unterschiedlichen Plattformen installiert sein. Das zentrale HCS-*Main* Modul ist an der Nachrichtenübertragung und -verarbeitung nicht beteiligt.

HL7-Connection-Server™

Die Serverkomponente des HL7-Connection-Server™ setzt sich aus dem *Daemon* und einer variablen Anzahl von *Empfangs-* und *Sendeprozessen* zusammen. Alle Prozesse müssen auf einem Rechner lokalisiert sein. Da für jede Nachricht in Form der Empfangs- und Sendeprozesse eigenständige Verarbeitungswege geschaffen werden, können die Nachrichten quasi-parallel verarbeitet werden. Der HL7-Connection-Server™ ist somit ebenfalls in die Klasse der **zentralen, lokal-nebenläufigen Kommunikationsserver** einzuordnen.

pro7-System™

Die Module des pro7-System™ lassen sich unter denselben Voraussetzungen wie bei HCS™ auf unterschiedliche Rechner verteilen. Aufgrund der zentralen Stellung des Verarbeitungsmoduls (pro7-Kern) ist aber nur eine serielle Nachrichtenverarbeitung möglich. Somit läßt sich das pro7-System™ in die Klasse der **verteilten, heterogenen, seriellen Kommunikationsserver** einordnen.

Kapitel 7

Kommunikationsverbindungen im Uni-Klinikum Münster

Am Uni-Klinikum in Münster wird seit einiger Zeit der Kommunikationsserver DATAGATE^{TM} eingesetzt. Über ihn wurden bisher drei Kommunikationsverbindungen implementiert. Außerdem stand der Kommunikationsserver für eine ausführliche Analyse zur Verfügung. Zur Vertiefung der Analyse wurden im Rahmen dieser Arbeit Kommunikationsverbindungen über DATAGATE^{TM} implementiert.

In diesem Kapitel werden die Ergebnisse dieser Implementation vorgestellt. Zu Beginn werden im Abschnitt 7.1 ab Seite 82 einige Kommunikationsbeziehungen im Uni-Klinikum Münster beispielhaft dargestellt. Im Abschnitt 7.2 ab Seite 83 folgt die Beschreibung der Kommunikationsverbindungen, die bisher über den Kommunikationsserver DATAGATE^{TM} implementiert wurden. Abschnitt 7.3 ab Seite 88 stellt schließlich die neuen Kommunikationsverbindungen vor, die im Rahmen dieser Arbeit über DATAGATE^{TM} implementiert wurden.

7.1 Darstellung der Kommunikationsbeziehungen zwischen einigen Abteilungen in einem Krankenhaus

In diesem Abschnitt sollen beispielhaft Kommunikationsbeziehungen zwischen einigen Abteilungen im Uni-Klinikum Münster dargestellt werden. Für die Analyse und Beschreibung der Beziehungen wurden sechs Abteilungen des Uni-Klinikums ausgewählt. Ausschlaggebend für die Auswahl dieser Abteilungen waren die folgenden Punkte:

1. Die Abteilungen mußten für die Unterstützung ihrer Routinetätigkeiten ein EDV-System einsetzen,
2. die Abteilungen mußten physikalisch an das Kliniknetz angeschlossen sein und
3. ein elektronischer Datenaustausch zwischen den EDV-Systemen dieser Abteilungen mußte bereits bestehen oder aber zumindest geplant sein.

Die Wahl fiel auf die Abteilungen, die bereits eine elektronische Verbindung zwischen ihren EDV-Systemen etabliert hatten bzw. die in der Liste der geplanten Verbindungen die höchste Priorität erhalten hatten. Die einzelnen Abteilungen sollen hier kurz mit ihren EDV-Systemen beschrieben werden:

1. **Patientendatenverwaltung**
   Die Patientendatenverwaltung setzt schon seit langem das PDV-System IDIK^{TM} von ehemals

Krupp Atlas Datensysteme ein. Es stellt das zentrale PDV-System des Klinikums dar. Die Stammdaten aller neu- und wiederaufgenommenen Patienten werden in diesem System über eine zentrale Aufnahmestelle erfaßt.

2. **Institut für Klinische Radiologie**
   In diesem Institut wird das System RADOS^{TM} der Firma `prompt!` Medizinische Informationssysteme GmbH eingesetzt.

3. **Institut für Klinische Chemie und Laboratoriumsmedizin**
   Das Zentrallaboratorium des Klinikums setzt für die Verwaltung der Untersuchungsergebnisse das System OLIS^{TM} der OSM Gesellschaft für offene Systeme in der Medizin mbH ein.

4. **Institut für Transfusionsmedizin**
   Hier wird das System PROTRANSMED^{TM} der Firma Elters eingesetzt. Es wird unter anderem für die Verwaltung der Befunde und der Stammdaten der behandelten Patienten eingesetzt.

5. **Institut für Nuklearmedizin**
   Die Nuklearmedizin setzt für die Verwaltung ihrer Befunde und der Stammdaten der behandelten Patienten das System MACDOC^{TM} der Firma Mac Software Design ein.

6. **Eine stationäre Abteilung**
   Die Vernetzung der stationären Abteilungen des Uni-Klinikums Münster war zu Beginn dieser Arbeit noch sehr unzureichend bzw. gar nicht erfolgt. Auch wurde auf den Stationen keine EDV-Systeme zur Unterstützung der Behandlung der Patienten und des Stationsmanagements eingesetzt. Am Institut für Medizinische Informatik und Biomathematik wird zur Zeit das Stationssystem OMD^{TM} der OSM Gesellschaft für offene Systeme in der Medizin mbH getestet. Dieses System soll in Zukunft in einigen stationären Abteilungen probeweise eingesetzt werden.

Zwischen diesen Abteilungen bestehen eine Reihe von Kommunikationsbeziehungen. Die ausgetauschten Daten beziehen sich alle auf die Behandlung der Patienten in den Abteilungen. Ausgetauscht werden Stammdaten, Untersuchungs- bzw. Behandlungsaufträge sowie die Befunde der Untersuchungen. Der Austausch der Daten erfolgt bisher überwiegend auf dem Postweg. Nur zwischen dem PDV-System und den Systemen in der Radiologie, der Transfusionsmedizin, dem Zentrallabor und dem Stationssystem bestehen jeweils elektronische Verbindungen, über die die Stammdaten der in den Abteilungen behandelten Patienten übertragen werden. Desweiteren werden die Befunde des Zentrallabors auf elektronischem Wege an das Stationssystem übermittelt. Die Verbindungen zwischen dem PDV-System und dem Stationssystem, sowie die zwischen dem Laborsystem und dem Stationssystem wurden über einen Kommunikationsserver realisiert. Sie werden im Abschnitt 7.2 genauer dargestellt. Alle übrigen elektronischen Verbindungen stellen Punkt-zu-Punkt-Verbindungen zwischen den Systemen dar.

Die Kommunikationsbeziehungen zwischen den Abteilungen werden in Abbildung 7.1 dargestellt. Die Station mit dem Befunddokumentationssystem OMD^{TM} ist stellvertretend für eine stationäre Abteilung des Klinikums zu sehen. Dargestellt werden die ausgetauschten Daten und die für den Austausch verwendeten Datenträger. So stellt eine durchgezogene Linie eine elektronische Verbindung dar, während eine gestrichelt gezeichnete Linie für einen Austausch der Daten auf dem Postweg steht (vgl. auch Legende der Datenflußdiagramme im Abschnitt A.2). Das Format der ausgetauschten Daten wird in dem Diagramm jeweils hinter der Datenklasse angegeben (z.B.: Patientenstammdaten:*Formular*). Die einzelnen Datenklassen sollen hier nicht weiter verfeinert werden. Die verwendeten Datenformate werden in Tabelle 7.1 beschrieben.

7.2 Beschreibung der bisher über einen Kommunikationsserver realisierten Verbindungen

Zwei der in Abbildung 7.1 dargestellten Kommunikationsverbindungen wurden im Vorfeld dieser Arbeit auf elektronischem Wege über einen Kommunikationsserver realisiert:

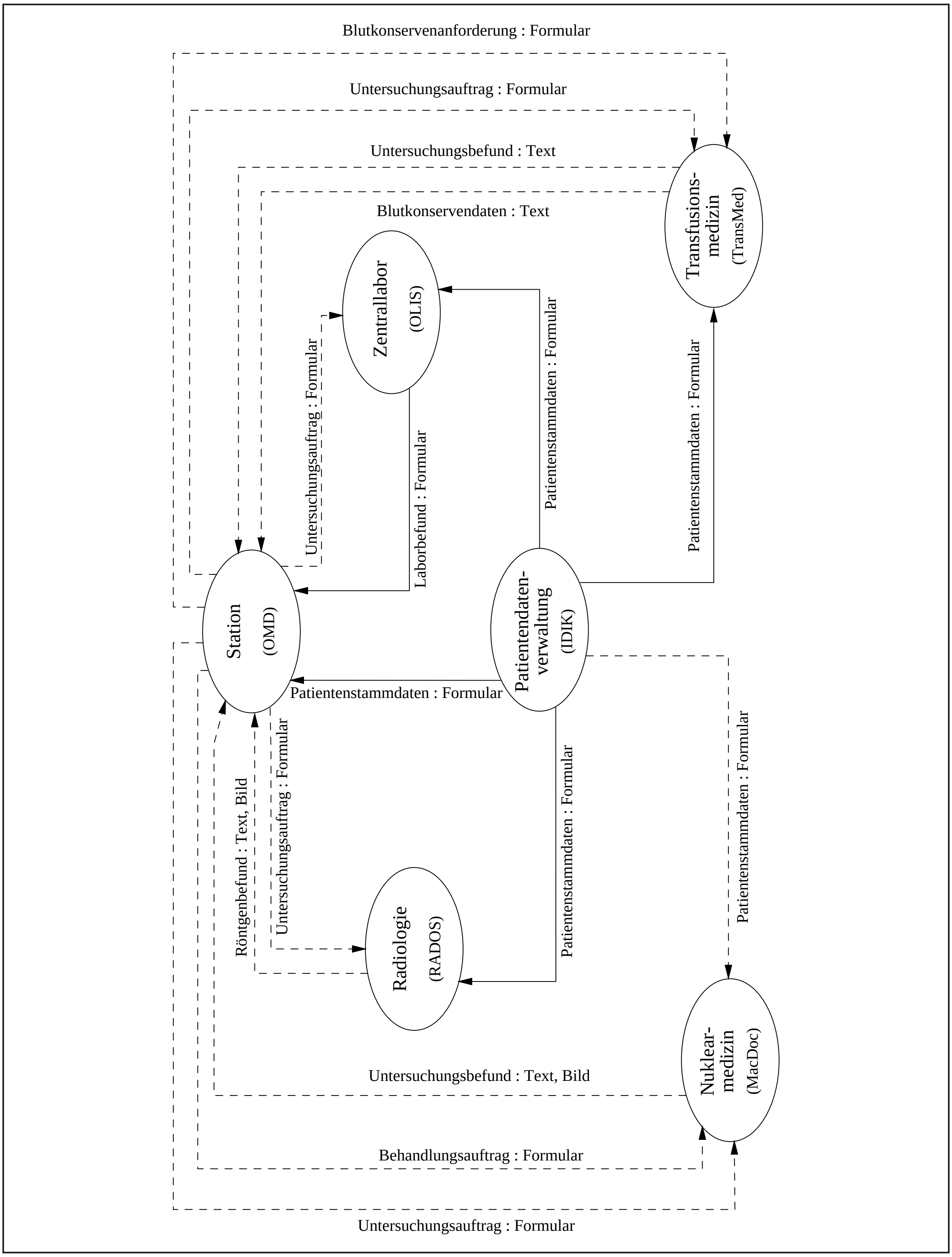


Abbildung 7.1: Darstellung der Kommunikationsbeziehungen zwischen einigen Abteilungen am Uni-Klinikum Münster. Bei den einzelnen Abteilungen ist jeweils in Klammern das eingesetzte EDV-System aufgeführt. Die Legende zu diesem Datenflußdiagramm befindet sich im Anhang A.2 auf Seite 110.

Datenformat	Beschreibung
Bilder	Graphiken, Diagramme, Röntgenbilder, Ultraschallbilder, etc.
Formular	Ein Dokument mit definierten Text- oder Datenfeldern, zum Teil maschinenlesbar.
Text	Ein Dokument mit einem Text in einer nicht festgelegten Form.

Tabelle 7.1: Beschreibung der in Abbildung 7.1 verwendeten Datenformate.

- Die Verbindung PDV-System → Stationssystem zur Übertragung der Stammdaten der neuaufgenommenen Patienten und

- die Verbindung Laborsystem → Stationssystem zur Übertragung der Befunde aus dem Zentrallabor.

Als Kommunikationsserver wird DataGate^{TM} der Firma STC eingesetzt. Abbildung 7.2 gibt einen schematischen Überblick über die Installation der Kommunikationsverbindungen. In den folgenden Abschnitten werden beide ausführlich beschrieben.

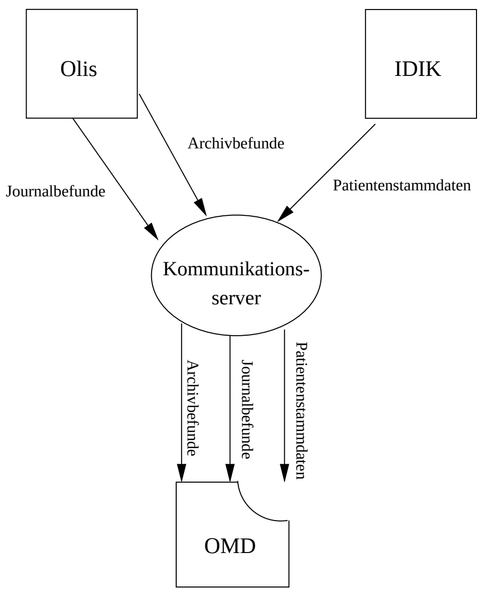


Abbildung 7.2: Darstellung der Kommunikationsverbindungen Idik^{TM} → OMD^{TM} und Olis^{TM} → OMD^{TM} über den Kommunikationsserver DataGate^{TM}.

7.2.1 Übertragung der Patientenstammdatensätze vom PDV-System zum Stationssystem

Generierung der Nachrichten im PDV-System

Das PDV-System schreibt alle Datensätze, die bei administrativen Transaktionen in der IDIK^{TM}-Datenbank entstehen (z.B. Neuaufnahmen, Verlegung, Entlassung von Patienten), in eine Datei mit dem Namen `tadhei` (siehe auch Abbildung 7.3). Diese Datensätze sind nach Satzarten gegliedert, die jeweils einer der administrativen Transaktionen entsprechen. Über die Satzarten ist eine genaue Struktur der Datensätze und die Bedeutung der einzelnen Datenfelder definiert.

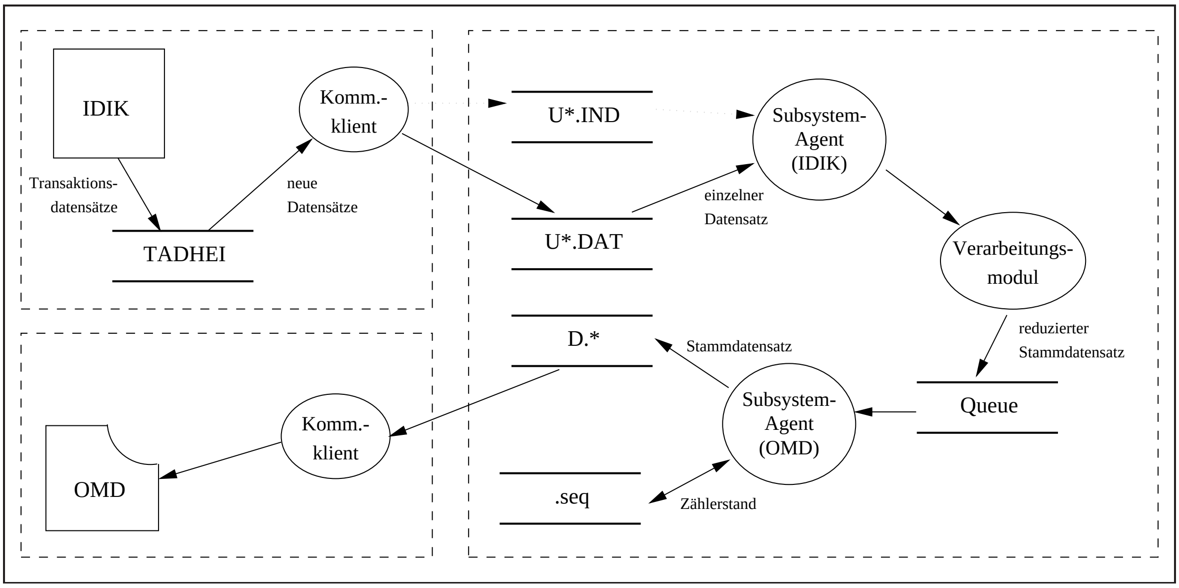


Abbildung 7.3: Darstellung der Patientendatenübertragung vom PDV-System IDIK^{TM} zum Stationssystem OMD^{TM} über den Kommunikationsserver DATAGATE^{TM}, wie sie am Uni-Klinikum Münster realisiert wurde.

Ein Kommunikationsklient liest stündlich die neu hinzugekommenen Datensätze aus dieser Datei aus, schreibt sie in eine Datei mit einem Namen der Form `U*.dat` ($*$ steht für eine fortlaufende Dezimalzahl) und überträgt diese per FTP in ein Verzeichnis des Rechners des Kommunikationsservers. Da die Übertragung der Datei einige Zeit in Anspruch nehmen kann, wird nach erfolgreicher Übertragung eine leere Datei mit dem gleichen Namen, aber der Endung `*.ind` erzeugt und ebenfalls vom Kommunikationsklienten per FTP übertragen. Diese Datei soll dem Kommunikationsserver das Ende der Übertragung der ersten Datei anzeigen.

Der Nachrichtenaustausch zwischen dem PDV-System und dem Kommunikationsserver erfolgt also dateibasiert. Eine Nachricht entspricht hier genau einem der Datensätze in der Datei `U*.dat`. Mit dieser Datei werden somit mehrere Nachrichten gleichzeitig an den Kommunikationsserver übermittelt. Das Kommunikationsprotokoll wird aus den Strukturen der Datensätze des PDV-Systems gebildet. Die Nachrichtentypen entsprechen den Satzarten der Datensätze.

Verarbeitung der Nachrichten im Kommunikationsserver

Der Subsystem-Agent `IDIK` des Kommunikationsservers (siehe Abbildung 7.3) überprüft ständig, ob in dem entsprechenden Verzeichnis eine neue Datei der Form `U*.ind` erscheint. Wurde eine neue Datei übertragen, so liest der Subsystem-Agent aus der zugehörigen Datei `U*.dat` die Datensätze satzweise ein und übermittelt sie zum Verarbeitungsmodul des Kommunikationsservers. Wurde von dem Verarbeitungsmodul der erfolgreiche Empfang der letzten Nachricht aus einer Datei bestätigt, so wird die Datei mit der zugehörigen Acknowledgement-Datei (`U*.ind`) vom Subsystem-Agenten gelöscht.

Im Verarbeitungsmodul werden die einzelnen Nachrichten anhand der Satzart des in der Nachricht enthaltenen Datensatzes identifiziert. Da der Empfänger der Nachrichten, das Stationssystem OMD^{TM}, nicht alle möglichen Satzarten verarbeiten kann, werden nur die Nachrichten mit den von OMD^{TM} unterstützten Satzarten an das Stationssystem weitergeleitet. Die übrigen Nachrichten werden vernichtet.

Aus den an OMD^{TM} weitergeleiteten Nachrichten werden nicht alle Datenfelder benötigt. Für jede Satzart sind daher im Verarbeitungsmodul Transformationsregeln definiert, die aus den Nachrichten die entsprechenden Datenfelder herausschneiden und in die von OMD^{TM} erwartete Nachrichtenstruktur transformieren.

Nach der Transformation werden die Nachrichten von dem Verarbeitungsmodul an den Subsystem-Agenten `OMD` des Stationssystems übermittelt. Dieser erzeugt für jede Nachricht eine Datei der Form `D.*` (der $*$ steht hier für eine fortlaufende Hexadezimalzahl) und schreibt die Nachrichten in diese Dateien. Die nächste freie Zahl für die Numerierung ermittelt der Subsystem-Agent anhand eines Zählers, der in einer Datei `.seq` gespeichert ist. Dieser Zähler wird nach der Übertragung der Nachricht in die Datei `D.*` vom Subsystem-Agenten aktualisiert.

Genaugenommen erzeugt der Subsystem-Agent `OMD` nicht direkt die Dateien in der Form `D.*`, sondern schreibt die Nachrichten erst in Dateien der Form `T*.tmp`. Diese Dateien werden dann vom Subsystem-Agenten in `D.*` umbenannt. Auf diese Weise wird verhindert, daß der Kommunikationsklient des Stationssystems nicht schon während des Schreibvorganges auf die Datei zugreifen kann. Sie wird für ihn erst dann „sichtbar", wenn der Schreibvorgang abgeschlossen ist.

Übernahme der Nachrichten im Stationssystem

Für die Übernahme der Nachrichten in das Stationssystem OMD^{TM} ist ein weiterer Kommunikationsklient zuständig. Dieser hat über ein gemountetes Verzeichnis direkten Zugriff auf die vom Subsystem-Agenten `OMD` erzeugten Dateien. Sobald eine Nachricht aus einer Datei eingelesen wurde, wird die Datei vom Kommunikationsklienten gelöscht.

In Abbildung 7.3 wird der hier beschriebene Nachrichtenfluß über den Kommunikationsserver in Form eines Datenflußdiagramms dargestellt.

7.2.2 Übertragung der Befunde aus dem Zentrallabor zum Stationssystem

Das Laborsystem OLIS^{TM} versendet zwei unterschiedliche Befundtypen: *Journalbefunde* und *Archivbefunde*. Während die Journalbefunde nur die Untersuchungsergebnisse einer Untersuchung enthalten, werden in den Archivbefunden Untersuchungsergebnisse mehrerer, zeitlich getrennter Untersuchungen dargestellt. Archiv- und Journalbefunde werden in OLIS^{TM} auf unterschiedliche Weise erzeugt. Auch in OMD^{TM} werden sie unterschiedlich behandelt. Bei beiden Subsystemen sind somit je zwei verschiedene Kommunikationsklienten notwendig: Jeweils ein Kommunikationsklient für die Generierung bzw. Übernahme der Befunde.

Die Übertragung der Befunde aus dem Zentrallabor erfolgt jeweils auf die gleiche Weise, wie die Übertragung der Patientenstammdaten (siehe Abschnitt 7.2.1). Lediglich die verwendeten Dateinamen und Verzeichnisse sind unterschiedlich. Für jede einzelne Kommunikationsverbindung ist ein Subsystem-Agent zuständig. Die Verbindungen und der Nachrichtenaustausch werden in Abbildung 7.4 gezeigt.

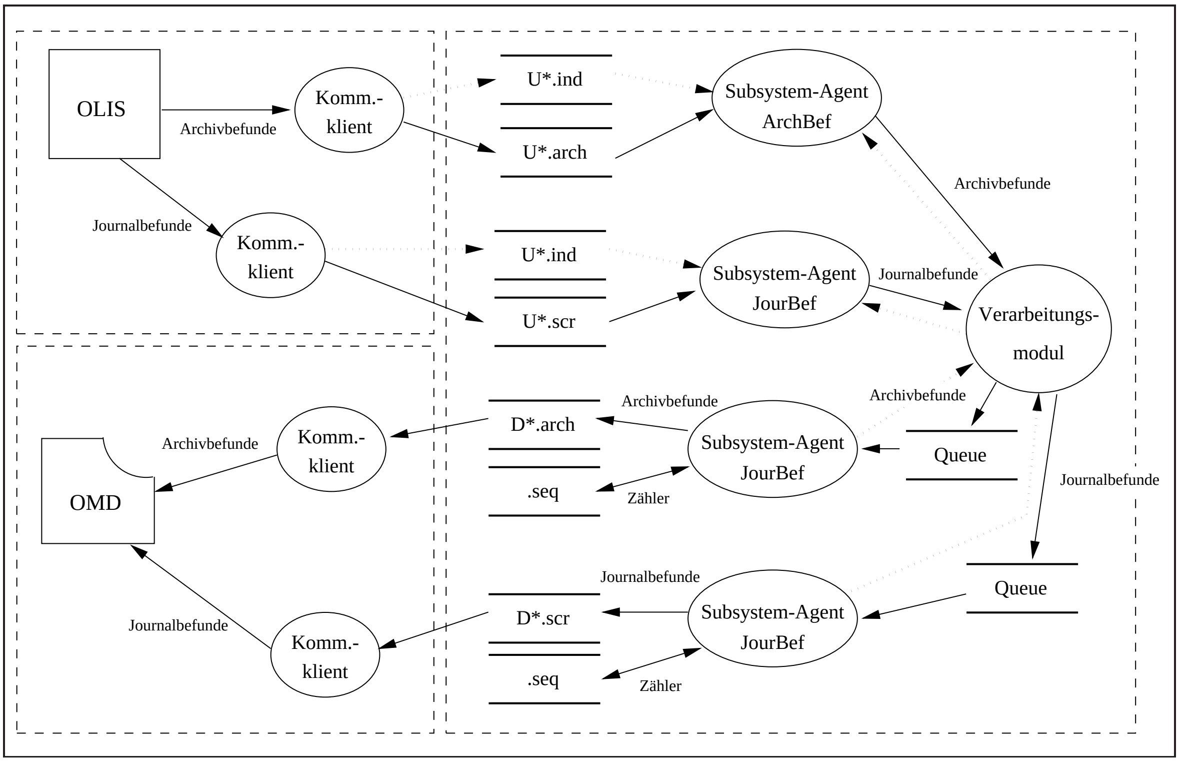


Abbildung 7.4: Darstellung der Befundübertragung aus dem Laborsystem OLIS^{TM} über den Kommunikationsserver DATAGATE^{TM} zum Stationssystem OMD^{TM}, wie sie zur Zeit am Uni-Klinikum in Münster realisiert ist.

7.3 Beschreibung der neu implementierten Kommunikationsverbindungen

Im Rahmen dieser Arbeit sollten drei weitere der in Abbildung 7.1 auf Seite 84 dargestellten Kommunikationsbeziehungen als elektronische Verbindungen über den Kommunikationsserver DATAGATE^{TM} realisiert werden. Im einzelnen waren dies die folgenden Verbindungen:

1. Übertragung der Patientenstammdaten aus dem PDV-System zum Abteilungssystem der Transfusionsmedizin
(TRANSMED^{TM}),

2. Übertragung der Patientenstammdaten aus dem PDV-System zum Abteilungssystem der Nuklearmedizin (MACDOC^{TM}) und

3. Übertragung der Befunde vom System TRANSMED^{TM} zum Stationssystem OMD^{TM}.

Die Übertragung der Stammdaten aus dem PDV-System sollte dabei jeweils durch die Abteilungssysteme über eine Anfrage (*Query*) nach den Stammdaten eines bestimmten Patienten getriggert werden. Es handelt sich hierbei also um eine Abfrage von Patientenstammdaten aus den Abteilungssystemen über den Kommunikationsserver in der Datenbank des PDV-Systems. Diese Abfrage sollte dabei *interaktiv* durchführbar sein. Das heißt, daß ein Anwender des Abteilungssystems durch die Eingabe einer Identifikationsnummer des gewünschten Patienten die Abfrage startet. Das Abteilungssystem generiert eine entsprechende Anfrage (*Query*), die in Form einer Nachricht an den Kommunikationsserver gesandt wird. Dieser leitet die Anfrage an das PDV-System weiter und erhält von diesem eine

Nachricht zurück, in der der Stammdatensatz des gewünschten Patienten enthalten ist (*Reply*). Diese Nachricht wird vom Kommunikationsserver an das Abteilungssystem weitergeleitet, wo der Stammdatensatz dem Anwender angezeigt wird.

In Abbildung 7.5 werden die drei Kommunikationsverbindungen schematisch dargestellt. Dabei werden aus Gründen der Übersichtlichkeit die Kommunikationsklienten der Subsysteme sowie die Subsystem-Agenten des Kommunikationsservers nicht abgebildet.

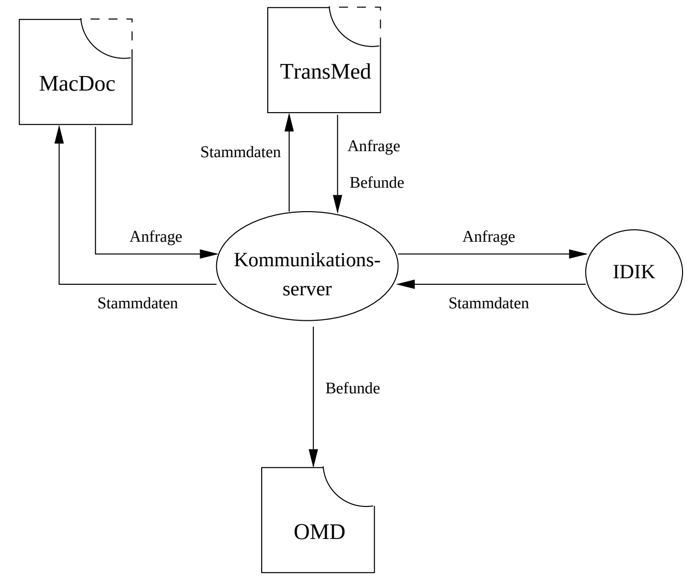


Abbildung 7.5: Schematische Darstellung der im Rahmen dieser Arbeit über einen Kommunikationsserver zu implementierenden Kommunikationsverbindungen.

Während der Bearbeitungszeit dieser Arbeit war es aufgrund fehlender Ressourcen im Bereich der Transfusionsmedizin nicht möglich, Befunde in TransMed™ für eine Übertragung zum Stationssystem bereitzustellen. Daher mußte auf die Implementierung dieser Kommunikationsverbindung verzichtet werden. Auch die Übertragung der Patientenstammdaten zu den Systemen der Nuklearmedizin (MacDoc™) und Transfusionsmedizin (TransMed™) konnte aufgrund von noch nicht abgeschlossenen Vertragsverhandlungen mit den Herstellern nicht in der oben beschriebenen Form realisiert werden.

Um nun aber doch eine der Kommunikationsverbindungen zu realisieren, wurde im Rahmen dieser Arbeit ein Programm entwickelt, das eine Simulation der Kommunikationsklienten der Abteilungssysteme MacDoc™ bzw. TransMed™ erlaubte. Die Entwicklung dieses Programms sowie die Implementierung der Kommunikationsverbindung zum PDV-System und den dafür notwendigen Kommunikationsklienten werden in den folgenden Abschnitten dargestellt.

7.3.1 Abfrage der Patientenstammdaten im PDV-System von einem Abteilungssystem aus

Wie bereits erwähnt, soll diese Kommunikationsverbindung eine interaktive Abfrage eines historischen (alten) Patientenstammdatensatzes von einem Abteilungssystem aus ermöglichen. Da es sich um eine interaktive Abfrage handelt, der Anwender im Abteilungssystem also durch Eingabe einer Patientenidentifikationsnummer die Abfrage triggert und auf eine Antwort (Stammdaten des Patienten) wartet, ist diese Kommunikationsverbindung als zeitkritisch zu betrachten. Besonders kurze Antwortzeiten sowie eine Synchronisation der mit der Abfrage verbundenen Nachrichtenübertragungen müssen erreicht werden. Dabei reicht es nicht, nur die Übertragung der Nachrichten zwischen dem Abteilungssystem und dem Kommunikationsserver zu synchronisieren. Kann der Kommunikationsserver aus irgendeinem Grund die Anfrage aus dem Abteilungssystem nicht unmittelbar an das PDV-System weiterleiten, so muß der Anwender des Abteilungssystems von diesem Sachverhalt informiert werden können, damit das Abteilungssystem nicht blockiert wird. Somit muß eine Synchronisation des Nachrichtenaustausches zwischen den Kommunikationsklienten des Abteilungssystems und des PDV-Systems stattfinden.

Der eingesetzte Kommunikationsserver DataGate™ bietet für eine Unterstützung einer Synchronisation zweier Subsysteme auf Nachrichtenebene keine Funktionalität an. Daher wurden für die beiden Abteilungssysteme TransMed™ und MacDoc™ jeweils nur für diese Abfrage der Stammdaten reservierte Kommunikationsverbindungen zum Kommunikationsserver implementiert. So konnte sichergestellt werden, daß Nachrichten von anderen Subsystemen den Synchronisationsprozeß nicht stören können.

Um einen möglichst schnellen Nachrichtenaustausch zu erreichen, wurde für die Übertragung der Nachrichten zwischen den Kommunikationsklienten der Abteilungssysteme und den Subsystem-Agenten des Kommunikationsservers eine direkte Interprozeßkommunikation über TCP-Sockets gewählt. Im folgenden werden nun die einzelnen Kommunikationsverbindungen sowie der für die Abfrage und Synchronisation notwendige Nachrichtenaustausch über diese Verbindungen beschrieben.

Abbildung 7.6 bietet einen Überblick über den Nachrichtenaustausch. Dabei werden die Nachrichten mit der Anfrage als *Query* und die Nachrichten mit der Antwort als *Reply* bezeichnet. Die Strukturen dieser Nachrichten (`PatDatQuery` bzw. `PatDatReply`) sind im Anhang D dargestellt. Da aus den oben beschriebenen Gründen die Abteilungssysteme TransMed™ sowie MacDoc™ nicht angeschlossen werden konnten, wird in der Abbildung die tatsächliche Implementierung mit der Simulation der Abteilungssysteme durch einen speziellen Kommunikationsklienten dargestellt. Dieser Kommunikationsklient wird als *IdikClient* bezeichnet und ist jeweils auf dem Rechner des Kommunikationsservers installiert.

Generierung der Anfrage im Kommunikationsklienten *IdikClient*

Der Kommunikationsklient *IdikClient* ermöglicht eine Simulation der interaktiven Abfrage der Stammdaten eines Patienten aus einem Abteilungssystem heraus. Aus einer über die Tastatur oder einer Datei eingelesenen Aufnahmenummer eines Patienten generiert er eine Nachricht der Form `PatDatQuery` (siehe Anhang D.1.1). Diese Aufnahmenummer dient als Identifikationsnummer für den entsprechenden Stammdatensatz im PDV-System. Eine ausführliche Beschreibung der Funktionalität von *IdikClient* befindet sich im Abschnitt 7.3.3 sowie im Anhang E.2.

Übertragung der Anfrage über den Kommunikationsserver

Der Kommunikationsklient *IdikClient* überträgt die Anfrage-Nachricht über eine TCP-Socket an den Subsystem-Agenten `CC_TransMed_In` des Kommunikationsservers, der sie an das Verarbeitungsmodul weiterleitet (siehe Abbildung 7.6). Die Synchronisation der Nachrichtenübertragung zwischen dem Kommunikationsklienten und dem Subsystem-Agenten erfolgt auf der Ebene des Transportprotokolls (TCP-Socket) gemäß dem *HL7 Lower Level Protocol* (siehe Anhang D.2).

Im Verarbeitungsmodul wird die Nachricht über das in der Struktur `PatDatQuery` vorgesehene Datenfeld `Nachrichtentyp` (Feldnummer 1) als eine Anfrage an das PDV-System identifiziert und an den Subsystem-Agenten des Kommunikationsklienten des PDV-Systems (`CC_IDIK_Out`) weitergeleitet. Eine Transformation der Nachricht muß nicht erfolgen. Die graphische Darstellung der entsprechenden Routing-Tabelle aus DataGate™ wird in Abbildung 7.7 auf Seite 92 dargestellt.

Sobald das Verarbeitungsmodul des Kommunikationsservers die Nachricht in die Queue des Subsystem-Agenten `CC_IDIK_Out` gestellt hat, erfolgt auf Transportprotokollebene über den Subsystem-Agenten

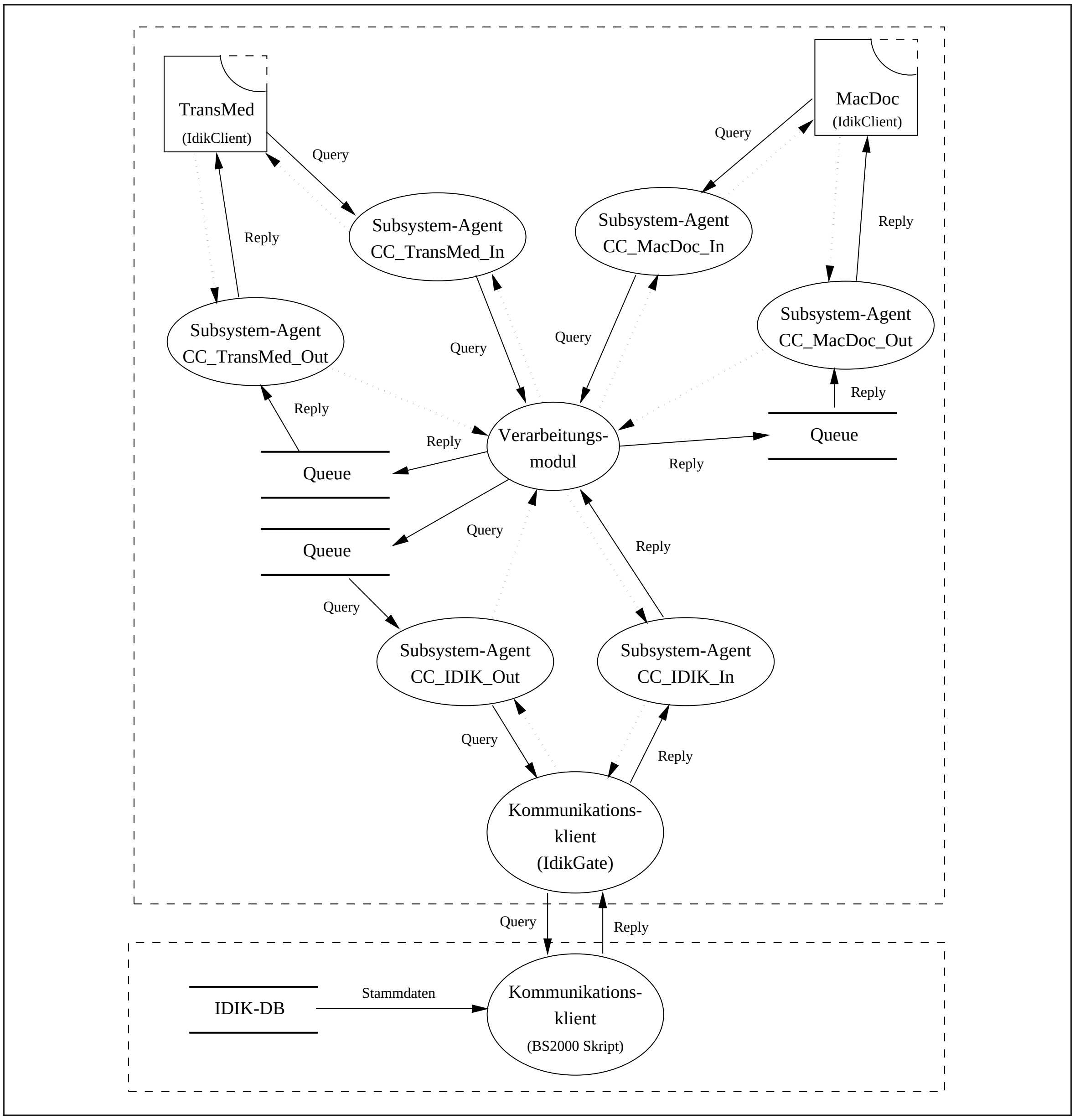


Abbildung 7.6: Darstellung des Nachrichtenaustausches für die Abfrage der Patientenstammdaten im PDV-System. Die Nachrichten mit der Anfrage werden als *Query* und die Nachrichten mit der Antwort als *Reply* bezeichnet. Da im Rahmen dieser Arbeit die Abteilungssysteme, von denen aus die Abfrage der Stammdaten getriggert wird, nicht angeschlossen werden konnten, werden hier nur die entsprechenden Kommunikationsklienten dargestellt, die für die Simulation der Abfrage entwickelt wurden. Sie werden als *IdikClient* bezeichnet. Die gepunkteten Linien deuten den Kontrollfluß an, der für die Synchronisierung der Nachrichtenübertragung auf Transportprotokollebene erfolgt.

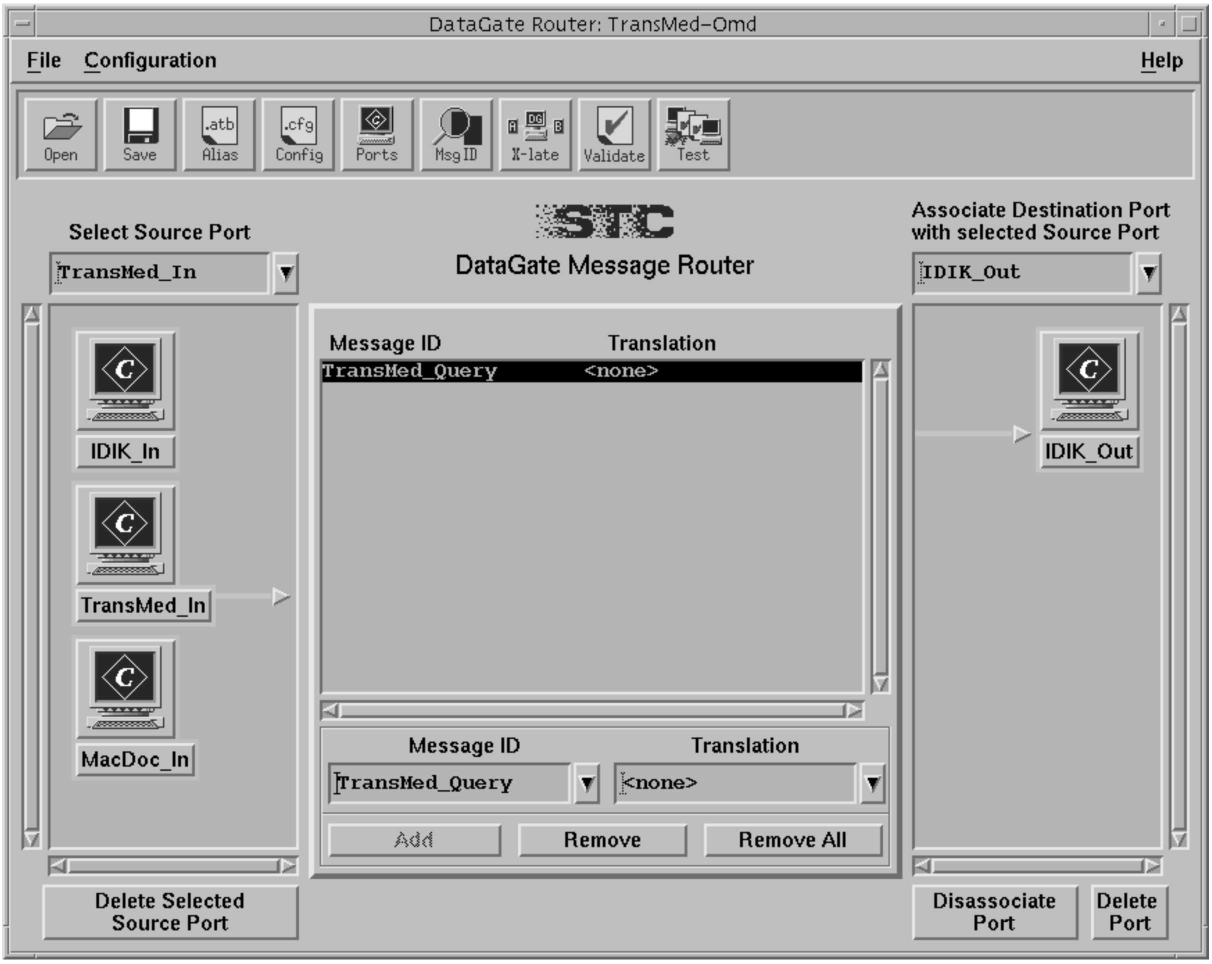


Abbildung 7.7: Graphische Darstellung der Routing-Tabelle in DataGate^{TM} für die Abfrage der Stammdaten im PDV-System.

`CC_TransMed_In` eine Bestätigung des erfolgreichen Empfangs der Nachricht an den Kommunikationsklienten *IdikClient*.

Der Subsystem-Agent `CC_IDIK_Out` entnimmt die Anfrage aus der Queue und übermittelt sie über eine TCP-Socket an den Kommunikationsklienten *IdikGate* des PDV-Systems. Die Synchronisation dieser Nachrichtenübertragung erfolgt ebenfalls gemäß dem *HL7 Lower Level Protocol*. Der Kommunikationsklient bestätigt allerdings erst dann den erfolgreichen Empfang der Anfrage, wenn eine entsprechende Antwortnachricht generiert werden konnte. Der Subsystem-Agent reicht eine positive Bestätigung an das Verarbeitungsmodul weiter, das dann die Anfrage-Nachricht aus der Queue löscht. Bei einer negativen Bestätigung wird die Anfrage vom Subsystem-Agenten erneut an den Kommunikationsklienten übertragen.

Empfang der Anfrage und Generierung der Antwort im Komunikationsklienten des PDV-Systems

Der Kommunikationsklient des PDV-Systems Idik^{TM} für die Abfrage eines historischen Patientenstammdatensatzes besteht aus zwei Prozessen.

1. Der erste Prozeß übernimmt den Austausch der Nachrichten mit den Subsystem-Agenten des Kommunikationsservers sowie alle Maßnahmen, die für die Synchronisation der Nachrichtenübertragungen notwendig sind. Dieser Prozeß ist auf der Plattform des Kommunikationsservers installiert und wird als *IdikGate* bezeichnet.

2. Der zweite Prozeß ist auf der Plattform des PDV-Systems lokalisiert. Dieser Prozeß übernimmt die Generierung der Datenbankoperationen, die für die Abfrage der Stammdaten aus der Datenbank des PDV-Systems notwendig sind. Er übernimmt von dem ersten Prozeß die Anfrage-Nachricht der Form `PatDatQuery` und liefert als Antwort eine Nachricht der Form `PatDatReply` zurück.

Die Aufteilung des Kommunikationsklienten auf zwei Prozesse war notwendig, da auf der Plattform des PDV-Systems (Siemens-Rechner mit BS 2000) mit vertretbarem Aufwand keine Möglichkeit bestand, einen Kommunikationsklienten zu entwickeln, der eine direkte Interprozeßkommunikation mit den Subsystem-Agenten des Kommunikationsservers ermöglicht hätte. So stellt der Prozeß auf der Plattform des PDV-Systems lediglich ein Skript dar, das über eine vom Prozeß auf der Plattform des Kommunikationsservers automatisierte TELNET-Verbindung aufgerufen wird. Einzelheiten zu dieser Form der Interprozeßkommunikation sind im Abschnitt 7.3.2 beschrieben.

Übertragung der Antwort über den Kommunikationsserver zum Kommunikationsklienten *IdikClient*

Die Übertragung der Antwort-Nachricht vom Kommunikationsklient des PDV-Systems über den Kommunikationsserver zum Kommunikationsklient des Abteilungssystems (*IdikClient*) erfolgt in der gleichen Weise, wie die Übertragung der Anfrage-Nachricht. Den Empfänger der Nachricht kann das Verarbeitungsmodul des Kommunikationsservers über das Datenfeld `Zurück-An` (Feldnummer 4 in der Nachrichtenstruktur `PatDatReply`) der Nachricht ermitteln. Eine entsprechende Identifizierungsregel für die Identifizierung einer Antwort-Nachricht an TRANSMED^{TM} in DATAGATE^{TM} wird beispielhaft für die übrigen Identifizierungsregeln in Abbildung 7.8 dargestellt. Eine Transformierung der Nachricht ist auch hier nicht erforderlich.

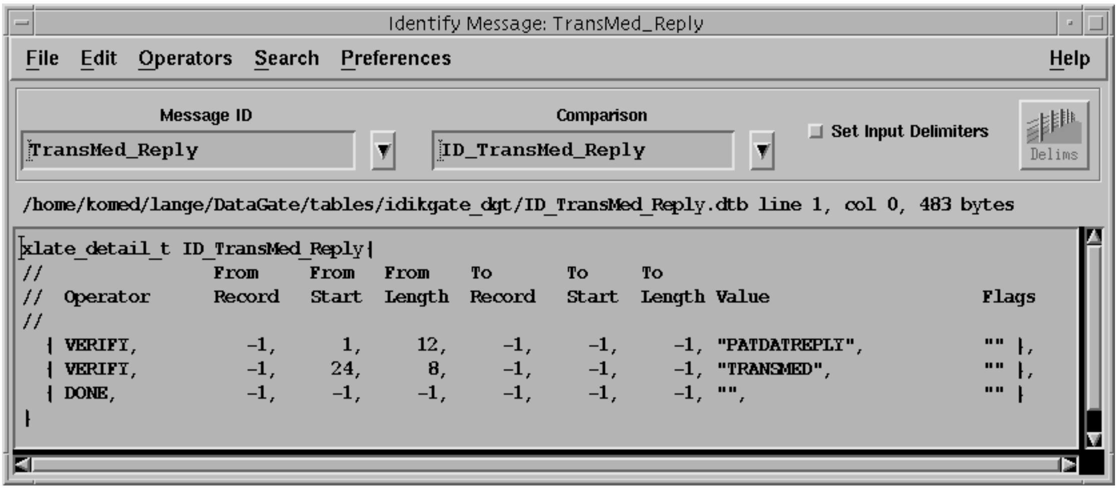


Abbildung 7.8: Darstellung der Nachrichtenidentifikationstabelle für die Nachrichten `PatDatReply` an das Abteilungssystem TRANSMED^{TM}.

Sobald die Antwort-Nachricht im Kommunikationsklienten *IdikClient* angekommen ist, wird ihr Inhalt – entweder die Stammdaten des Patienten oder eine Fehlermeldung (siehe Strukturbeschreibung von `PatDatReply` im Anhang D.1.2) – zur Simulation der interaktiven Abfrage angezeigt.

7.3.2 Entwicklung des Kommunikationsklienten *IdikGate*

Für den Aufbau einer Kommunikationsverbindung zwischen dem Kommunikationsserver und dem PDV-System IDIK^{TM} zur Abfrage der Patientenstammdaten mußte im Rahmen dieser Arbeit ein Kommunikationsklient entwickelt werden. Dieser sollte einen dateilosen Nachrichtenaustausch mit dem Kommunikationsserver ermöglichen, damit die Übertragungen der Nachrichten möglichst verzögerungsfrei erfolgen kann.

Da auf der Plattform des PDV-Systems (Siemens-Rechner mit BS 2000) eine Entwicklung eines solchen Kommunikationsklienten nicht möglich war, eine Abfrage der Stammdaten in der Datenbank des PDV-Systems aber nur von dieser Plattform aus erfolgen kann, wurde der Kommunikationsklient auf zwei Prozesse aufgeteilt (vgl. auch Abschnitt 7.3.1): Ein Prozeß realisiert auf der Plattform des Kommunikationsservers den dateilosen Nachrichtenaustausch mit dem Kommunikationsserver und ein weiterer Prozeß auf der Plattform des PDV-Systems generiert die für die Abfrage der Stammdaten notwendigen Datenbankoperationen.

Die Kommunikation zwischen diesen beiden Prozessen erfolgt über eine automatisierte TELNET-Verbindung. Hierunter ist das folgende zu verstehen: Der Prozeß auf der Plattform des Kommunikationsservers – im folgenden als *IdikGate* bezeichnet – baut über das UNIX-Programm TELNET eine Verbindung zum Rechner des PDV-Systems auf. Ist die Verbindung etabliert, ruft *IdikGate* auf dem Rechner des PDV-Systems ein Programm auf, das die Abfrage der Stammdaten in der PDV-Datenbank durchführt. Dieses Programm ist genau der Teil des Kommunikationsklienten, der auf der Plattform des PDV-Systems implementiert wurde. Als Eingabe erhält das Programm die Anfrage-Nachricht (`PatDatQuery`), die *IdikGate* vom Kommunikationsserver empfangen hat. Der Rückgabewert des Programms entspricht einer Nachricht vom Typ `PatDatReply` und enthält die gewünschten Stammdaten. Diese Ausgabe des Programms ist von *IdikGate* über die TELNET-Verbindung abzufangen und als Antwort-Nachricht an den Kommunikationsserver zuversenden.

In diesem Abschnitt wird die Entwicklung von *IdikGate* beschrieben. Der zweite Prozeß des Kommunikationsklienten, das Programm auf der Plattform des PDV-Systems, wurde von einem der Administratoren des PDV-Systems implementiert.

Anforderungen

Der überwiegende Teil der Anforderungen an den Prozeß *IdikGate* des Kommunikationsklienten wurden bereits in den vorherigen Abschnitten erläutert. Hier sollen diese und weitere Anforderungen nochmals stichpunktartig aufgestellt werden.

- Der Nachrichtenaustausch mit dem Kommunikationsserver ist über ein dateiloses Transportprotokoll zu realisieren. Für die Übertragung sind TCP-Sockets und das *HL7 Lower Level Protocol* vorgesehen.
- Der Nachrichtenaustausch mit dem Kommunikationsserver ist so zu synchronisieren, daß der Sender der Anfrage in jedem Fall eine Antwort-Nachricht der Form `PatDatReply` erhält. Auch dann, wenn eine Abfrage in der PDV-Datenbank zur Zeit nicht möglich ist. Für diesen Fall ist im Kommunikationsprotokoll eine spezielle Nachricht vorgesehen (siehe Nachrichtenstrukturbeschreibung von `PatDatReply` im Anhang D.1.2).
- Für den Aufruf des Prozesses auf der Plattform des PDV-Systems ist von *IdikGate* eine elektronische Verbindung über das TELNET-Programm zum Rechner des PDV-Systems aufzubauen und zu unterhalten.
- Alle Vorgänge im Kommunikationsklienten sollen, soweit möglich, in einer Log-Datei protokolliert werden.
- Zusätzlich sollen alle Anfrage- und Antwort-Nachrichten von *IdikGate* in einer Datei archiviert werden.

- Während der festgelegten Wartungszeiten des PDV-Systems ist eine Abfrage nicht möglich. Zu diesen Zeiten muß die Verbindung zum PDV-Rechner von *IdikGate* selbständig auf- bzw. abgebaut werden.

Aus diesen Anforderungen wurde eine Benutzeranleitung für das Programm *IdikGate* geschrieben. Diese befindet sich im Anhang E.1.

Systemarchitektur

IdikGate wird über zwei Subsystem-Agenten mit dem Kommunikationsserver verbunden: Einer für die Übertragung der Anfrage-Nachrichten und einer für die Übertragung der Antwort-Nachrichten (siehe auch Abbildung 7.6 auf Seite 91). Entsprechend müssen in *IdikGate* zwei TCP-Sockets (*Ports*) für den Nachrichtenaustausch unterhalten werden.

In-Port: Über diese Socket werden die Anfrage-Nachrichten empfangen. Sie wird von *IdikGate* als „Server" eingerichtet. Der entsprechende Subsystem-Agent des Kommunikationsservers (`CC_IDIK_Out`) baut die Verbindung zu dieser Socket auf.

Out-Port: Über diese Socket werden die Antwort-Nachrichten zum Kommunikationsserver übertragen. Sie wird von dem Subsystem-Agenten `CC_IDIK_In` eingerichtet. Hier muß der Kommunikationsklient den Aufbau der Verbindung übernehmen.

Die Verbindung zu dem Rechner des PDV-Systems wird schließlich als **Telnet-Port** bezeichnet.

Der Nachrichtenfluß zwischen dem Kommunikationsserver und den Prozessen des Kommunikationsklienten wird mit Hilfe eines Datenflußmodells modelliert. Es ist in Abbildung 7.9 dargestellt. In diesem Diagramm wird der Kommunikationsserver nur in Form der beiden Subsystem-Agenten repräsentiert. In dem Datenflußmodell wurde auf die Darstellung des Datenspeichers „Log-Datei" aus Gründen der Übersichtlichkeit verzichtet. Auf diesen Datenspeicher greifen alle Prozesse schreibend zu, indem sie dort die Log-Informationen ablegen. Der Prozeß „Abfrage-Skript" stellt das Programm des Kommunikationsklienten dar, der auf der Plattform des PDV-Systems implementiert wurde und über die TELNET-Verbindung aufgerufen wird.

Für die Darstellung der Funktionsanalyse wurde die Notation für das dynamische Modell aus OMT gewählt (vgl. [RBP+91], Anhang A.1). Das Ergebnis ist in Abbildung 7.10 dargestellt. Dieses dynamische Modell entspricht einem erweiterten Zustandsübergangsdiagramm. Jeder Zustand (*State*) wird neben der eindeutigen Bezeichnung durch eine Angabe der Aktion erweitert, die innerhalb des Zustandes ausgeführt wird. Die Annotationen an den Kanten (Zustandsübergänge) beschreiben die Ereignisse, die Auslöser für diesen Zustandsübergang sind. Kanten ohne Annotation stellen die regulären Zustandsübergänge dar. Dies sind Übergänge, die im „Normalfall" ausgeführt werden. Alle anderen Übergänge beschreiben somit Ausnahmesituationen. Zur besseren Übersichtlichkeit sind in der Abbildung 7.10 die Kanten der regulären Übergänge dicker als die der Ausnahmekanten gezeichnet.

Für eine ausführliche Beschreibung der Funktionalität von *IdikGate* sei auf das Handbuch im Anhang E.1 verwiesen.

Implementierung

Für die Implementierung des Prozesses *IdikGate* des Kommunikationsklienten wurde ANSI-C verwendet. Auf eine Abbildung des Quellcodes wird aus Platzgründen verzichtet.

Für die Automatisierung der TELNET-Verbindung zum Rechner des PDV-Systems konnte auf das Programm EXPECT zurückgegriffen werden. Dieses Programm ermöglicht eine Automatisierung normalerweise interaktiver Programme (wie z.B. TELNET). Auf eine ausführliche Darstellung dieses Programmes wird hier verzichtet und auf die Literatur (z.B. [Lib94]) verwiesen. EXPECT wurde auch für die Implementierung der Alarmierungsfunktion (siehe Zustand „Alarmierung" in Abbildung 7.10) verwendet.

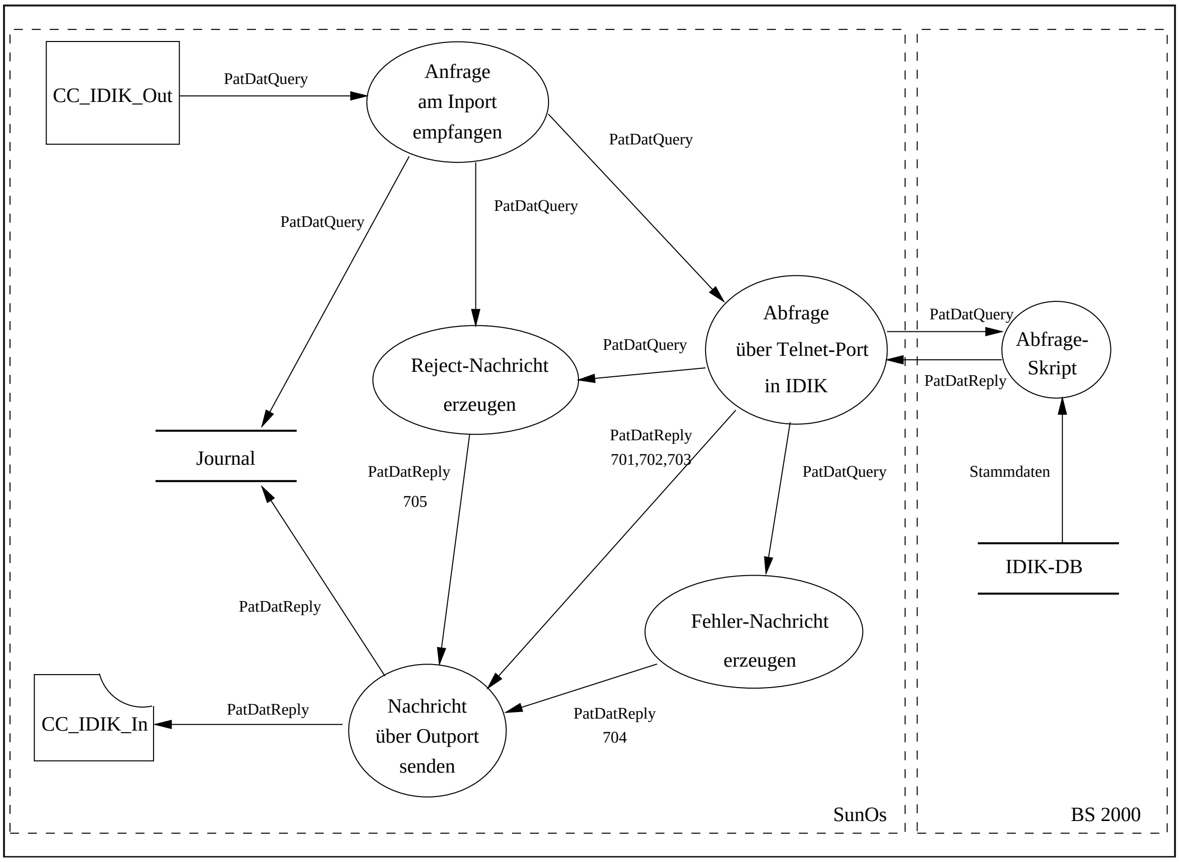


Abbildung 7.9: Datenflußdiagramm des Kommunikationsklienten für die Abfrage der Stammdaten im PDV-System. Der Datenspeicher „Log-Datei“ wird aus Gründen der Übersichtlichkeit nicht dargestellt. Alle Prozesse legen in diesem Log-Informationen ab. Die Zahlen 701 - 705 bei den Nachrichten der Form `PatDatReply` geben den jeweiligen Nachrichtentyp (Feldnummer 2) der Nachricht an (siehe Strukturbeschreibung im Anhang D.1.2).

7.3.3 Entwicklung des Kommunikationsklienten *IdikClient*

Wie bereits erwähnt, konnten während der Bearbeitungszeit dieser Arbeit die Abteilungssysteme MacDoc^{TM} und TransMed^{TM} nicht an den Kommunikationsserver angeschlossen werden. Um aber dennoch die Kommunikationsverbindung zur Abfrage eines historischen Patientenstammdatensatzes testen zu können, wurde im Rahmen dieser Arbeit ein Kommunikationsklient entwickelt, der eine Simulation der interaktiven Abfrage der Stammdaten ermöglicht. Die Entwicklung dieses Kommunikationsklienten wird im folgenden kurz dargestellt.

Anforderungen

Da *IdikClient* nur zum Testen der interaktiven Abfrage benutzt werden soll, lassen sich die Anforderungen klar formulieren. *IdikClient* soll es einem Anwender ermöglichen, Aufnahmenummern über die Tastatur einzugeben und unmittelbar danach Auszüge des zugehörigen Stammdatensatzes am Bildschirm einzusehen. Zusätzlich soll der vollständige Stammdatensatz in einer Datei gespeichert werden. Neben der Eingabe der Aufnahmenummer über die Tastatur soll auch das Einlesen mehrerer Aufnahmenummern aus einer Eingabedatei möglich sein.

Da *IdikClient* zum Testen der Kommunikationsverbindung zum PDV-System eingesetzt werden soll,

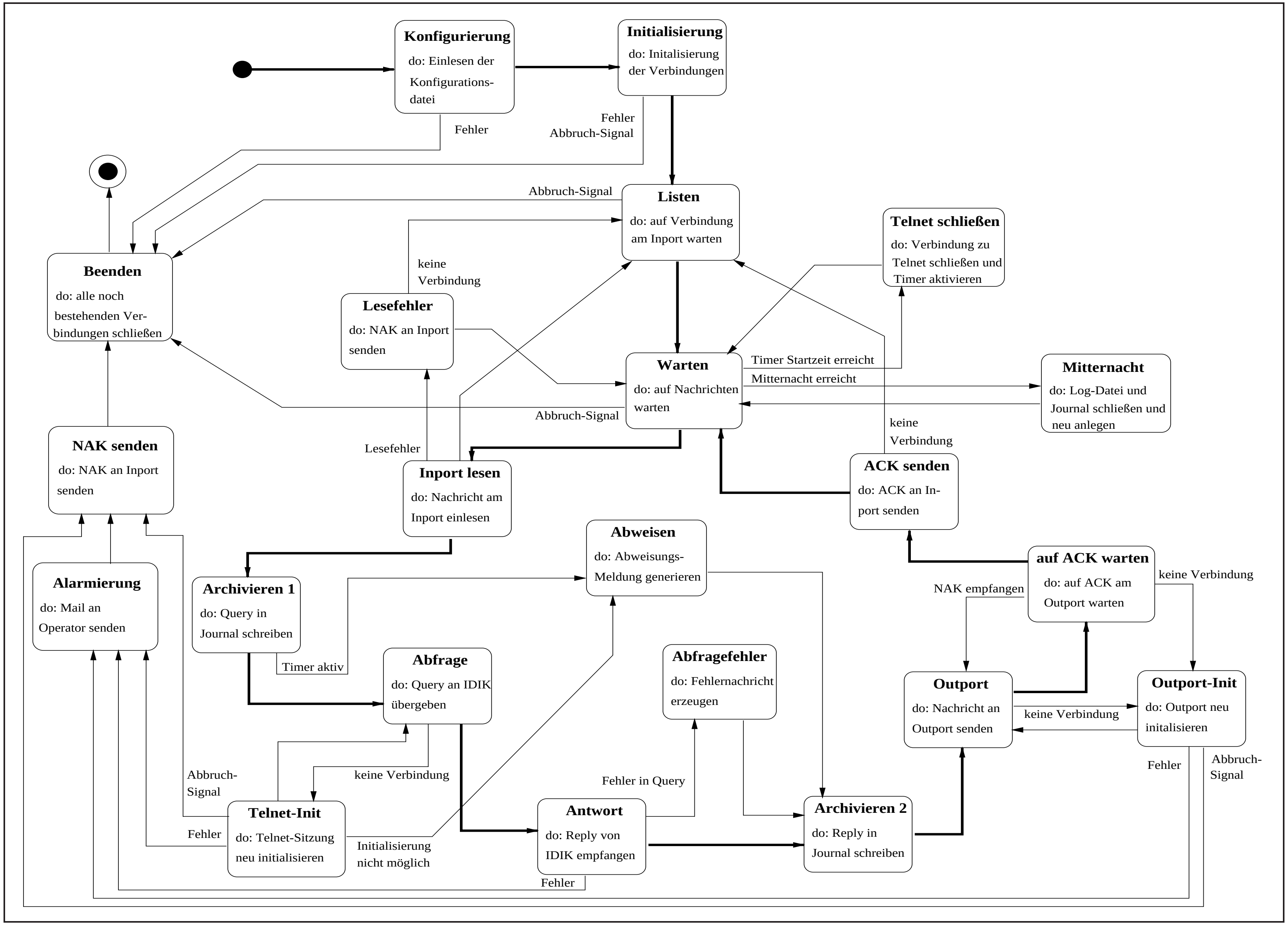


Abbildung 7.10: Dynamisches Modell des Kommunikationsklienten *IdikGate* für die Abfrage der Stammdaten im PDV-System. Die regulären Zustandsübergänge sind durch Fettdruck hervorgehoben.

ist ein ausführliches Logging aller Vorgänge notwendig. Dies soll durch die Verwaltung von Log-Dateien durch *IdikClient* geschehen.

Da geplant war, daß der Test der Kommunikationsverbindung auch von Mitarbeitern der betroffenen Abteilungen durchgeführt werden sollte, wurde eine Benutzeranleitung für *IdikClient* geschrieben. Sie ist im Anhang E.2 dargestellt.

Systemarchitektur

Auch *IdikClient* wird über zwei Subsystem-Agenten mit dem Kommunikationsserver verbunden. Die Schnittstellen (TCP-Sockets) werden analog zu den entsprechenden Schnittstellen in *IdikGate* als `Outport` (für die Anfrage-Nachrichten) und `Inport` (für die Antwort-Nachrichten) bezeichnet. Der Datenfluß innerhalb von *IdikClient* ist in Abbildung 7.11 dargestellt.

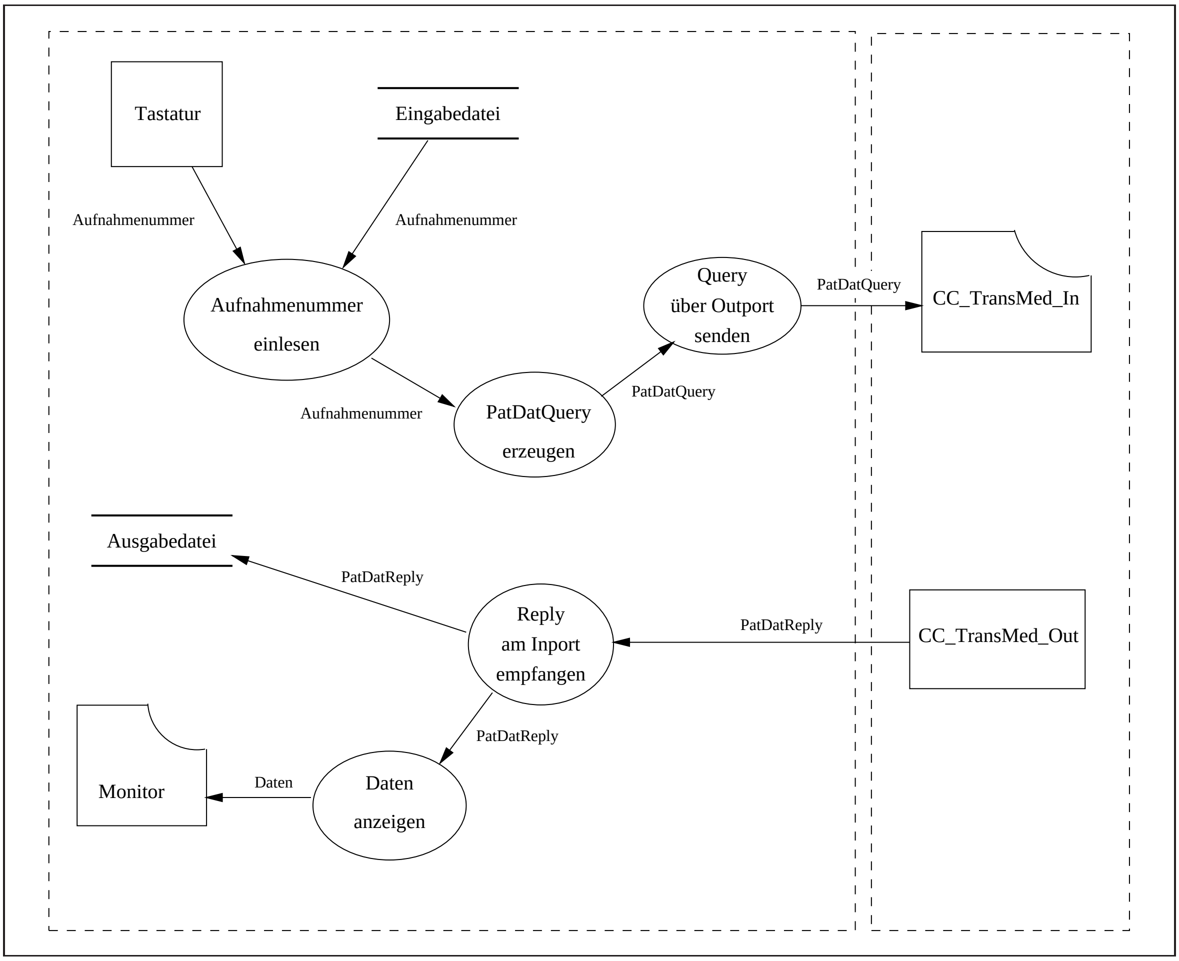


Abbildung 7.11: Datenflußdiagramm des Kommunikationsklienten *IdikClient*.

Da *IdikClient* ausschließlich für eine interaktive Abfrage der Stammdaten gedacht ist, ist die Abfrage blockierend implementiert. Dies bedeutet, daß *IdikClient* nach dem Versenden der Anfrage `PatDatQuery` auf die zugehörige Antwort `PatDatReply` wartet. Antworten, die einen Stammdatensatz zu einer anderen Aufnahmenummer enthalten, werden direkt in der Ausgabedatei gespeichert. Sie werden nicht am Bildschirm angezeigt. Solche Antworten können entstehen, wenn der Kommunikationsklient nach der Eingabe einer Aufnahmenummer abgebrochen wurde, bevor der zugehörige

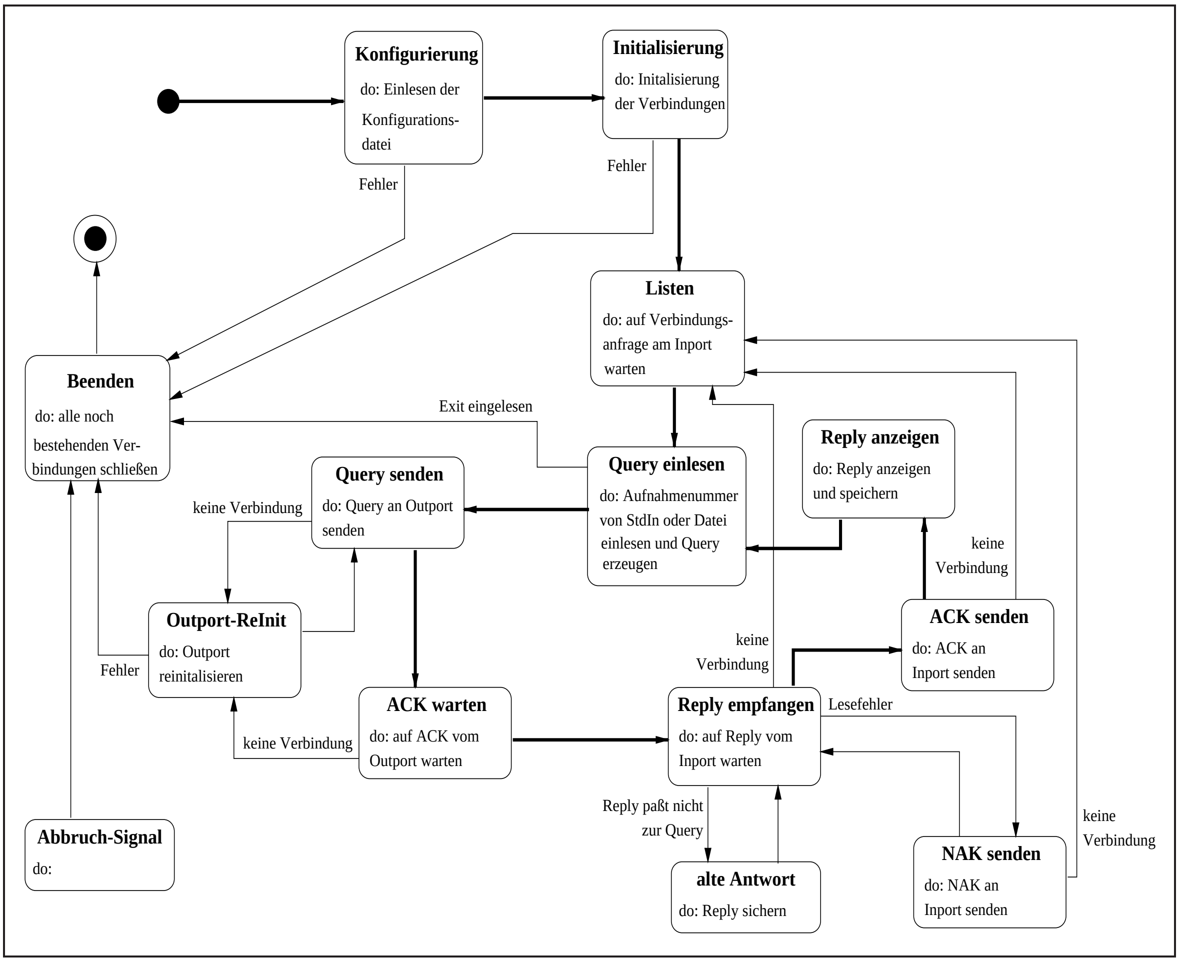


Abbildung 7.12: Dynamisches Modell zum Kommunikationsklienten *IdikClient*. Die regulären Zustandsübergänge sind durch Fettdruck hervorgehoben.

Stammdatensatz empfangen wurde. Während des Wartens auf die Antwort vom Kommunikationsserver können keine weiteren Eingaben vom Anwender vorgenommen werden. Die weitere Funktionalität von *IdikClient* geht aus dem dynamischen Modell in Abbildung 7.12 hervor. Auch hier wurde die OMT-Notation verwendet.

Implementierung

IdikClient wurde ebenfalls wie *IdikGate* in ANSI-C programmiert. Auf eine Abbildung des Quellcodes wird ebenfalls aus Platzgründen verzichtet.

Kapitel 8

Diskussion der Ergebnisse

In diesem Kapitel sollen die Ergebnisse dieser Diplomarbeit diskutiert werden. Nach einer Analyse des Vergleichs der Kommunikationsserver (Abschnitt 8.1) und der implementierten Kommunikationsverbindungen (Abschnitt 8.2 ab Seite 102) werden im Abschnitt 8.3 ab Seite 104 einige weitere Ansätze zur Unterstützung der Kommunikation in einem KKS durch Kommunikationsserver diskutiert.

8.1 Diskussion des Vergleichs der Kommunikationsserver

Bei dem Vergleich der Kommunikationsserver ist seit Untersuchungen von Heitmann eine deutliche Marktkonzentrierung zu beobachten. Von denen in [Hei96] untersuchten sechs verschiedenen Kommunikationsservern, sind heute nur noch vier auf dem Markt vertreten. Die Hersteller der übrigen beiden Produkte bieten nun Kommunikationsserver anderer Hersteller an. Zwar ist mit dem PRO7-SYSTEM^{TM} ein weiterer Kommunikationsserver hinzugekommen, doch bleibt abzuwarten, inwieweit er sich auf dem heiß umkämpften Markt durchsetzen wird. Zusammen mit dem Kommunikationsserver HCS^{TM} ist das PRO7-SYSTEM^{TM} technisch als sehr fortschrittlich anzusehen. Als Marktführer sind zur Zeit die beiden Kommunikationsserver CLOVERLEAF^{TM} und DATAGATE^{TM} anzusehen. Sie stellen wohl auch die am weitesten ausgereiften Produkte dar.

Eine grobe Auswertung des Vergleichs wird im folgenden stichpunktartig gegeben:

- Für die Konfigurierung der Kommunikationsverbindungen bieten alle betrachteten Kommunikationsserver eine graphische Oberfläche an. Die Art und Wiese der Darstellung der einzelnen Konfigurationen ist aber sehr unterschiedlich. Dies gilt vor allem für die Darstellung der Kommunikationsverbindungen (vgl. Abbildungen der Bildschirmmasken im Kapitel 6).
- Keiner der Kommunikationsserver bietet bisher eine Unterstützung multimedialer Daten an.
- Die Kontrolle der Kommunikationsverbindungen durch den Administrator des Kommunikationsservers erfährt bei allen Produkten durch Log-Dateien und Nachrichtenarchive eine gute Unterstützung. Statistiken über zum Beispiel die Auslastung des Kommunikationsservers und über weitere Parameter werden bisher allerdings kaum unterstützt. Umfangreiche Alarmfunktionen bietet lediglich der Kommunikationsserver CLOVERLEAF^{TM} an.
- Eine externe Dokumentation der Kommunikationsverbindungen auf Papier ist bei allen Kommunikationsservern nur sehr eingeschränkt möglich. Der Ausdruck der Konfigurationen oder der Bildschirmmasken reicht in der Regel für eine übersichtliche und aussagekräftige Dokumentation nicht aus.
- Die Persistenzsicherung innerhalb des Kommunikationsservers wird bei allen Produkten gewährleistet. Der hierfür aufgebrachte Aufwand ist allerdings sehr unterschiedlich. Er reicht von einem

einfachen Speichern der Nachrichten nach dem Prinzip *store-and-forward* in Queues bis hin zu einer komplizierten Verwaltung der Nachrichten in *Recover*-Datenbanken. Allerdings zeigen die Erfahrungen, die in Münster mit den dort implementierten Verbindungen gewonnen werden konnten, daß auch die beste Persistenzsicherung durch eine unsichere Nachrichtenübertragung zwischen den Kommunikationsklienten und den Subsystem-Agenten zunichte gemacht werden kann (siehe Abschnitt 8.2).

- Das Management von fehlerhaften Nachrichten wird nur bei dem Kommunikationsserver CLOVERLEAF^{TM} umfangreich unterstützt. Hier ist allerdings abzuwägen, inwieweit fehlerhafte Nachrichten im Betrieb von getesteten Kommunikationsverbindungen überhaupt noch auftreten können. In der Regel werden die Nachrichten durch die Kommunikationsklienten automatisiert aufgebaut. Sind diese ausführlich getestet, so treten Fehler wie ein fehlerhafter Strukturaufbau der Nachrichten normalerweise nicht mehr auf. Auch ist die Frage, inwieweit die Subsysteme bzw. die Kommunikationsklienten etwas mit den Fehlermeldungen des Kommunikationsservers anfangen können. Das beste Fehlermanagement nützt nichts, wenn der Verursacher des Fehlers die Hinweise auf den Fehler ignoriert.

- Das Testen der Kommunikationsverbindungen wird nur bei den Kommunikationsservern CLOVERLEAF^{TM} und DATAGATE^{TM} so unterstützt, daß die Tests unabhängig von dem laufenden Betrieb durchgeführt werden können.

- Der Schutz der Nachrichten vor einem unbefugten Zugriff wird von allen Kommunikationsservern auf das Betriebsystem abgewälzt. Keines der Produkte bietet Verschlüsselungsmethoden für das Versenden der Nachrichten an. Zwar lassen sich bei den meisten Kommunikationsservern durch spezielle Subsystem-Agenten Verschlüsselungsmethoden für den Nachrichtenaustausch mit den Subsystemen realisieren, auf den Nachrichtenaustausch zwischen den Modulen des Kommunikationsservers hat der Administrator aber keinen Einfluß. Der Schutz der Nachrichten durch die Verwendung von RPC's bei dem Kommunikationsserver HCS^{TM} und dem PRO7-SYSTEM^{TM} ist hier hervorzuheben. Da beide Produkte aber eine verteilte Architektur unterstützen, bedeutet die Abstützung auf RPC's auch eine gewisse Unflexibilität, da nun alle eingesetzten Plattformen auch RPC's unterstützen müssen.

  Auch hier muß man sich die Frage stellen, inwieweit Nachrichten innerhalb eines Kommunikationsservers durch besondere Maßnahmen vor einem unbefugten Zugriff zu schützen sind. Normalerweise bietet das Betriebsystem hier ausreichende Möglichkeiten. Als Schwachpunkte sind die dateibasierte Nachrichtenübertragung sowie die Speicherung der Log-Dateien und Nachrichtenarchive in unverschlüsselten ASCII-Dateien zu nennen. Hier sind die Nachrichten für jeden, der Zugang zum Rechner des Kommunikationsservers hat oder sich verschafft, einsehbar.

- Der Anschluß eines Datenbankmanagementsystems wird von keinem Kommunikationsserver durch eine spezielle Schnittstelle unterstützt. Zwar benutzen einige Kommunikationsserver intern Datenbanken, diese lassen sich aber nicht für andere Zwecke wie zum Beispiel eine zentrale Befunddatenbank nutzen. Bei den Kommunikationsservern CLOVERLEAF^{TM}, DATAGATE^{TM} und dem PRO7-SYSTEM^{TM} läßt sich eine solche Schnittstelle zu einem DBMS durch spezielle Subsystem-Agenten allerdings realisieren.

  Es bleibt die Frage, ob die Verwaltung einer umfangreichen, zentralen Befunddatenbank in das Aufgabengebiet eines Kommunikationsservers paßt. Mit den in der Definition 2.4 auf Seite 7 beschriebenen Aufgaben eines Kommunikationsservers hat sie nichts mehr zu tun und gehört eigentlich in das Aufgabengebiet eines eigenständigen Subsystems. Aber auch für die Unterstützung der Kommunikation zwischen den Subsystemen des KKS läßt sich eine Datenbank sinnvoll einsetzen. So zum Beispiel zur Speicherung des aktuellen Aufenthaltortes eines Patienten. Der Kommunikationsserver wäre dann selbständig in der Lage, zum Beispiel Befunde aus der Radiologie an die Krankenstation zu senden, auf der sich der entsprechende Patient zur Zeit befindet.

8.2 Analyse der implementierten Kommunikationsverbindungen

In diesem Abschnitt sollen die an den Uni-Kliniken in Münster über den Kommunikationsserver DATAGATE™ implementierten Kommunikationsverbindungen analysiert und über die Erfahrungen berichtet werden, die bisher mit diesen Verbindungen gewonnen werden konnten.

Bei den in Münster implementierten Kommunikationsverbindungen werden vier verschiedene Transportprotokolle eingesetzt:

1. Eine dateibasierte Übertragung der Nachrichten über das Programm FTP von dem PDV-System und dem Laborsystem zum Kommunikationsserver,

2. eine dateibasierte Übertragung über ein gemountetes Verzeichnis zwischen dem Kommunikationsserver und dem Stationssystem,

3. eine dateilose Übertragung über TCP-Sockets zwischen den Kommunikationsklienten *IdikGate* bzw. *IdikClient* und dem Kommunikationsserver sowie

4. eine dateilose Übertragung über eine automatisierte TELNET-Sitzung zwischen den Prozessen des Kommunikationsklienten des PDV-Systems zur Abfrage der Patientenstammdaten.

Die Vor- und Nachteile der vier Transportprotokolle sowie die bisher im Betrieb aufgetretenen Probleme sollen nun im folgenden dargestellt werden.

1. **Dateibasierte Übertragung der Nachrichten über das Programm Ftp**
   Die Übertragung der Stammdaten der neuaufgenommenen Patienten aus dem PDV-System sowie die Übertragung der Befunde aus dem Zentrallabor erfolgt dateibasiert über das Programm FTP [Tan96]. Dazu erzeugen die entsprechenden Kommunikationsklienten in den beiden Abteilungssystemen jeweils eine Datei, in die die Nachrichten geschrieben werden. Diese Dateien werden dann jeweils mit Hilfe des Programms FTP zum Rechner des Kommunikationsservers übertragen. Dort werden die Nachrichten von den Subsystem-Agenten aus den Dateien entnommen und dem Verarbeitungsmodul des Kommunikationsservers übergeben (vgl. auch Abschnitt 7.2.1 und 7.2.2).

   Die Übertragung der Nachrichten per FTP hat sich – im Vergleich zu den anderen drei Transportprotokollen – als recht unzuverläßig erwiesen. Mehrmals kam es zu einem Verlust oder zu einer Verdoppelung von Nachrichten im Kommunikationsserver. Bisher konnten zum einen die fehlende Synchronisation der Übertragung zwischen den Kommunikationsklienten und den Subsystem-Agenten, sowie die Tatsache, daß in einer Datei mehrere Nachrichten übertragen werden als Ursache der Fehler ermittelt werden.

   Zu einem Verlust von Nachrichten kam es stets dann, wenn der Speicherplatz auf der Festplatte des Rechners des Kommunikationsservers für die Nachrichtendatei nicht ausreichte. Der Kommunikationsklient des PDV-Systems bzw. des Laborsystems überträgt die Dateien mit den Nachrichten ohne vorher zu überprüfen, ob der Plattenplatz auf dem Zielrechner ausreichend ist. Auch eine Kontrolle, ob die Übertragung vollständig erfolgt ist, wird nicht vorgenommen. Reicht der Plattenplatz nicht aus, so gehen die Nachrichten in der entsprechenden Datei verloren. Abhilfe könnte hier eine Kontrolle der Übertragung auf Vollständigkeit durch den Kommunikationsklienten schaffen. Auch eine Synchronisierung der Übertragung mit dem Subsystem-Agenten würde einen Verlust der Nachrichten verhindern. Erhält der Kommunikationsklient innerhalb einer gewissen Zeitspanne vom Subsystem-Agenten keine Bestätigung des Empfangs der Nachrichten, so kann dieser von einer fehlgeschlagenen Übertragung ausgehen und diese wiederholen.

   Die Tatsache, daß in einer Datei mehrere Nachrichten übertragen werden, hat sich als Ursache für die Verdoppelung der Nachrichten im Kommunikationsserver herausgestellt. Der Grund für die Verdoppelung liegt in der Arbeitsweise des verwendeten Subsystem-Agenten. Dieser liest die

Nachrichten aus der Datei einzeln ein. Wurde eine Nachricht eingelesen, so wird diese an das Verarbeitungsmodul des Kommunikationsservers weitergegeben. Erst wenn dieses den Empfang bestätigt hat, wird eine weitere Nachricht aus der Datei eingelesen. Die Datei wird gelöscht, wenn die letzte Nachricht erfolgreich an das Verarbeitungsmodul übergeben wurde. Fällt der Subsystem-Agent aber zum Beispiel aufgrund eines Systemabsturzes aus, bevor alle Nachrichten aus der Datei eingelesen und die Datei gelöscht werden konnte, so beginnt der Subsystem-Agent nach der Wiederherstellung seiner Funktionstüchtigkeit mit dem Einlesen der Nachrichten wieder am Anfang der Datei. Alle Nachrichten, die vor dem Ausfall bereits eingelesen wurden, werden nun erneut an das Verarbeitungsmodul übertragen und somit verdoppelt. Abhilfe würde hier eine Markierung oder ein Löschen der bereits erfolgreich übertragenen Nachrichten in der Datei schaffen.

Eine Verdoppelung der Nachrichten läßt sich aber auch dann nicht vollständig ausschließen, wenn pro Datei nur eine Nachricht vorgesehen ist. Der Subsystem-Agent kann auch hier ausfallen, bevor die Datei gelöscht werden konnte. Auch eine Bestätigung der Bestätigung einer erfolgreichen Übertragung bietet keine absolute Sicherheit. Letztendlich läßt sich nur die Wahrscheinlichkeit einer Verdoppelung der Nachrichten minimieren. Hier muß der Aufwand und die damit verbundene Performanz der Nachrichtenübertragung gegen die Wahrscheinlichkeit der Verdoppelung und den Schaden, den eine doppelte Nachricht beim Empfänger anrichtet, abgewogen werden.

Ein wichtiger Nachteil der Übertragung von Nachrichten über FTP ist der damit verbundene Zeitaufwand. Die Herstellung der Verbindung zwischen den Rechnern und der Dateiaustausch benötigt unter Umständen viel Zeit. Vor allem dann, wenn der Austausch zu Zeiten großer Netzauslastungen erfolgt oder große Dateien mit vielen Nachrichten übertragen werden. In Münster wurden über dieses Transportprotokoll nur nicht-zeitkritische Kommunikationsverbindungen realisiert. Der Zeitfaktor spielte hier also keine Rolle.

2. **Dateibasierte Übertragung der Nachrichten über ein gemountetes Verzeichnis**
   Bei dieser dateibasierten Übertragungsmethode können sowohl das Subsystem, als auch der Kommunikationsserver über ein gemountetes Verzeichnis [Tan96] direkt auf die Datei mit den Nachrichten zugreifen (vgl. Abschnitt 7.2.1 und 7.2.2). Auch hier können im Prinzip die selben Probleme auftreten, wie bei einer Übertragung der Dateien per FTP. Bei der Installation in Münster traten während der analysierten Zeit allerdings keine der oben genannten Fehler auf.

   Durch den Wegfall der Übertragung der Dateien läßt sich dieser Nachrichtenaustausch schneller als der mit FTP gestalten. Allerdings sind auch hier die in Münster realisierten Kommunikationsverbindungen als nicht-zeitkritisch zu betrachten, so daß eine Bestimmung der durchschnittlichen Übertragungszeit der Nachrichten nicht vorgenommen wurde.

3. **Dateilose Übertragung der Nachrichten über TCP-Sockets**
   Die interaktive Abfrage der Patientenstammdaten von einem Abteilungssystem aus war von vornherein als zeitkritisch anzusehen. Daher wurden die hierfür notwendigen Kommunikationsverbindungen über dateilose Transportprotokolle realisiert (vgl. Abschnitt 7.3.1). Die dabei verwendeten TCP-Sockets [Tan96] erbrachten ein recht gutes Antwort-Zeit-Verhalten (durchschnittlich 2.23 Sekunden bei 60 Anfragen).

   Ein Nachteil der dateilosen Kommunikation über Sockets ist allerdings die fehlende Authentifizierung der Kommunikationspartner. Richtet ein Subsystem-Agent für den Empfang und das Senden der Nachrichten eine Socket als Server ein, so kann im Prinzip jedes andere Programm bei bekannter Portnummer eine Verbindung zu dieser Socket aufbauen. Notwendig ist hier, daß nach dem Aufbau einer Verbindung eine Authentifizierung des Kommunikationspartners stattfindet. Diese kann zum Beispiel über den Austausch von Paßwörtern geschehen. Der Subsystem-Agent überträgt nur dann Nachrichten, wenn er zuvor ein Paßwort vom Kommunikationspartner erhalten hat. Kann der Kommunikationspartner sein Paßwort nicht senden, so wird die Verbindung vom Subsystem-Agenten abgebrochen.

4. **Dateilose Übertragung der Nachrichten über eine automatisierte Telnet-Sitzung**
   Der doch etwas ungewöhnliche Weg, Nachrichten zwischen zwei Programmen über eine automatisierte TELNET-Verbindung auszutauschen, gestaltete sich unkomplizierter, als zu Beginn der

Implementierung erwartet wurde. Eine große Hilfe bat hier das Programm EXPECT [Lib94], mit dessen Hilfe die Automatisierung der TELNET-Sitzung sehr einfach zu realisieren war. Dennoch traten einige Probleme während des Betriebs auf, die sich aus der automatisierten Steuerung des eigentlich interaktiv ausgelegten Programms auf der Seite des PDV-Rechners (desweiteren als *Abfrageskript* bezeichnet) durch das Programm auf der Seite des Kommunikationsservers (*IdikGate*) ergaben.

Zum einen konnten nicht alle möglichen Reaktionen des Abfrageskripts auf fehlerhafte Eingaben adäquat in *IdikGate* behandelt werden. Dies lag vor allem an einer fehlenden Dokumentation zum Abfrageskript. Fehlerreaktionen konnten nur durch „Ausprobieren“ oder Rückfragen beim Programmierer ermittelt werden, so daß es während der Testphase immer wieder zu zeitraubenden Fehleranalysen gekommen ist. Eine große Hilfe waren hier die ausführlichen Log-Dateien, die von *IdikGate* angelegt werden.

Ein weiterer Grund lag in der Steuerung der TELNET-Sitzung durch *IdikGate*. Das Abfrageskript mußte durch Eingabe eines bestimmten Befehls korrekt beendet werden, bevor die Verbindung zwischen den Rechnern unterbrochen wurde. Ansonsten war bei einem erneuten Aufruf des Abfrageskripts dessen Funktionalität gestört. Wurde die Verbindung aber vorzeitig unterbrochen, war ein korrektes Beenden des Abfrageskriptes durch *IdikGate* nicht mehr möglich.

Aufgrund dieser Probleme sollte ein Nachrichtenaustausch über eine automatisierte TELNET-Sitzung in einem Krankenhaus nicht zur Regel werden. Ist die einzige Alternative aber ein dateibasierter Nachrichtenaustausch, so ist sie bei zeitkritischen Kommunikationsverbindungen doch sinnvoll einsetzbar.

Das Problem der Authentifizierung der Kommunikationsklienten gegenüber dem Kommunikationsserver tritt nicht nur bei der Nachrichtenübertragung über TCP-Sockets auf. Auch bei den anderen drei vorgestellten Transportprotokollen ist es bisher nicht zufriedenstellend gelöst. Lediglich bei der Übertragung der Nachrichten über FTP und die automatisierte TELNET-Sitzung erfolgt über eine Paßwortabfrage eine gewisse Kontrolle des Kommunikationspartners. Bei der Methode, bei der die Nachrichten über ein gemountetes Verzeichnis ausgetauscht werden, kann im Prinzip jeder, der Zugang zum entsprechenden Rechner des Subsystems (hier das Laborsystem) oder zum Rechner des Kommunikationsservers hat, auf die Nachrichten in dem Verzeichnis zugreifen. Dies ist auch ein Grund, warum in den Uni-Kliniken in Münster einige Abteilungen eine Übertragung per FTP vorziehen. Am dringlichsten ist aber eine Lösung des Problems bei der dateilosen Übertragung über zum Beispiel TCP-Sockets. Hier kann – bei bekannter Portnummer und bekanntem Kommunikationsprotokoll – im Prinzip jeder eine Verbindung zum Kommunikationsserver herstellen und zum Beipiel Patientenstammdaten abfragen.

Man muß aber auch hier den Aufwand der Authentifizierung gegen die tatsächliche Gefahr eines unbefugten Zugriffs abwägen. Die Netzwerke in den Krankenhäusern sind häufig nach außen abgeschirmt, so daß eine „Gefahr“ nur von den Subsystemen des KKS ausgehen kann. Auch sind die Kommunikationsprotokolle und die Kommunikationsverbindungen in ihren Einzelheiten im allgemeinen nur wenigen Personen bekannt. Wesentlich größer dürfte die Gefahr eines unerlaubten „Mithörens“ der Nachrichten während des Versendens sein. Hier ist eine Verschlüsselung der Nachrichten, vor allem über ungeschützte Verbindungen, umso dringender erforderlich.

8.3 Ausblick

Kommunikationsserver lassen sich im Krankenhaus nicht nur als zentrale Nachrichtenvermittlungsstelle eines KKS einsetzen. Neben einer Vernetzung mehrerer Krankenhäuser über einen Kommunikationsserver ist auch in großen Abteilungen eines Krankenhauses ist ein Einsatz von Kommunikationsservern häufig sinnvoll. Diese „lokalen“ Kommunikationsserver steuern dann den Nachrichtenaustausch zwischen den Systemen in einer Abteilung. Über eine Kommunikationsverbindung zum zentralen Kommunikationsserver des Krankenhauses besteht ein Anschluß an das KKS. So können zum Beispiel in einer Intensivstation eines Krankenhauses die verschiedenen Überwachungs- und Behandlungsgeräte über den Kommunikationsserver mit einem zentralen Arbeitsplatz verbunden werden, von dem aus alle Patienten überwacht werden können.

Auch innerhalb einzelner Geräte läßt sich ein – nur auf die notwendigen Komponenten reduzierter – Kommunikationsserver einsetzen. Auf diese Weise wird es möglich, die Schnittstelle des Gerätes ausgesprochen flexibel zu halten.

Im folgenden sollen noch einige weitere Ansätze zur Realisierung von Kommunikationsverbindungen in einem KKS mit der Hilfe von Kommunikationsservern diskutiert werden.

Transaktionskonzepte für Kommunikationsserver

Transaktionen sind ein fundamentales Konzept in Datenbanksystemen. Operationen auf den Daten in einer Datenbank werden dann als *Transaktionen* bezeichnet, wenn die folgenden vier Eigenschaften gewährleistet sind:

1. **Atomarität**
   Aus der Sicht des Anwenders stellt eine Transaktion eine atomare Einheit dar. Dies bedeutet, daß die zur Transaktion gehörenden Datenbankoperationen – auch bei einem Systemabsturz – entweder vollständig oder gar nicht durchgeführt werden.

2. **Konsistenzerhaltung**
   Durch die Datenbankoperationen muß die logische Korrektheit der Daten erhalten bleiben. Das DBMS kann eine Transaktion zurückweisen, wenn durch sie Integritätsbedingungen verletzt werden. Ein Beispiel für eine Integritätsbedingung wäre, daß in einem Datumsfeld nur das aktuelle oder ein zukünftiges Datum eingetragen werden darf.

3. **Isolation**
   Das DBMS muß sicherstellen, daß es zwischen einer Transaktion und parallelen Transaktionen anderer Anwender keine unerwünschten Wechselwirkungen geben kann. Transaktionen müssen also isoliert von anderen Transaktionen durchgeführt werden können.

4. **Persistenz**
   Wurde eine Transaktion erfolgreich beendet, so müssen die durch die Transaktion durchgeführten Änderungen an dem Datenbestand der Datenbank dauerhaft sein.

In der Literatur werden diese Eigenschaften auch unter dem Begriff „ACID-Prinzip“ (Atomicity, Consistency, Isolation, Durability) zusammengefaßt [Hac85].

Es gibt Bestrebungen, den Transaktionsbegriff auch auf die Nachrichtenübertragung in Kommunikationsservern anzuwenden. So werden zum Beispiel im HL7-CONNECTION-SERVER^{TM} *Transaktionen* definiert, die sich aus einer oder mehreren Nachrichten zusammensetzen. Auch die im Abschnitt 4.2.2 vorgestellten komplexen Aktionen gehen in diese Richtung. Hier werden mehrere Nachrichten zu einer Einheit zusammengefaßt, die – möglichst – vollständig übertragen werden sollen.

Im folgenden soll diskutiert werden, inwieweit ein Kommunikationsserver die vier Eigenschaften von Transaktionen gewährleisten kann.

Atomarität

Eine einzelne Nachricht läßt sich durchaus als eine atomare Einheit auffassen. Ist sie korrekt aufgebaut, so garantiert der Kommunikationsserver über die Maßnahmen zur Persistenzsicherung, daß die Nachricht an den oder die Empfänger ausgeliefert wird. Über eine Synchronisierung der Übertragung kann dem Sender der Nachricht die erfolgreiche Vermittlung an die Empfänger angezeigt werden.

Atomarität bedeutet aber nicht nur, daß eine Nachricht erfolgreich übermittelt wird. Vielmehr müssen die mit den Nachrichten verbundenen Änderungen in den Datenbeständen der

Empfänger entweder bei allen Empfängern vollständig oder aber bei keinem durchgeführt werden. Dies beinhaltet, daß gegebenenfalls in einem Subsystem eine Datenänderung wieder rückgängig gemacht werden muß, wenn ein weiterer Empfänger der Nachricht nicht erreichbar ist bzw. die Durchführung der Änderung ablehnt. So zum Beispiel bei einer Stammdatenänderungsanzeige, die vom PDV-System an alle Abteilungssysteme gesandt wird, die die Stammdaten zu diesem Patienten gespeichert haben. Ist eines der Abteilungssysteme nicht erreichbar oder kann die Nachricht an ein Abteilungssystem nicht übermittelt werden, da bei der Transformation der Nachricht in das entsprechende Kommunikationsprotokoll ein Fehler aufgetreten ist, so darf die Änderung der Stammdaten in keinem der Abteilungssysteme durchgeführt werden. Wurde sie in einigen Systemen bereits durchgeführt, so müßte sie dort wieder rückgängig gemacht werden.

Konsistenzerhaltung
Eine Verletzung von Konsistenzbestimmungen (Integritätsbedingungen) in einer Datenbank treten nur bei schreibenden Zugriffen auf. In einem KKS entspricht dies einer Nachricht, die neue Daten (z.B. Befunde zu einem Patienten) oder Änderungsanzeigen (z.B. Änderungen in den Stammdaten eines Patienten) enthalten. Da der Nachrichtenaustausch zwischen den Subsystemen über definierte Kommunikationsprotokolle erfolgt, lassen sich die Verletzungen der Konsistenzbestimmungen eines Subsystems durch eine Nachricht bereits bei der Transformation der Nachricht im Kommunikationsserver feststellen. Nachrichten, die konsistenzverletzende Daten enthalten, würden so das Subsystem erst gar nicht erreichen. Hier könnten allerdings nur grobe Verletzungen der Konsitenzbestimmungen abgefangen werden. Eine Kontrolle aller Integritätsbedingungen einer Datenbank würde die Transformationsregeln sehr komplex werden lassen. Außerdem sind meistens nicht alle Integritätsbedingungen – außer dem Administrator der Datenbank – bekannt. Bei einer synchronen Übertragung können die konsistenzverletzenden Nachrichten dem Sender angezeigt werden.

Isolation
Wechselwirkungen zwischen parallelen Nachrichten an ein Subsystem können nur dann auftreten, wenn ein Subsystem über mehrere Kommunikationsverbindungen zum Kommunikationsserver verfügt. Zusätzlich müssen Nachrichten, die sich gegenseitig beeinflussen können, über verschiedene Kommunikationsverbindungen an das Subsystem übermittelt werden. Nachrichten, die sich gegenseitig beeinflussen können, treten in einem Krankenhaus allerdings äußerst selten auf. Normalerweise ist zum Beispiel für die Verwaltung der Stammdaten der Patienten nur ein System im Krankenhaus, das PDV-System, zuständig. Nur von diesem werden zum Beispiel Änderungsanzeigen oder Neuaufnahmen an andere Abteilungssysteme versandt. Da für diese Nachrichten in der Regel nur eine Kommunikationsverbindung zum Kommunikationsserver besteht, können hier parallele Nachrichten nicht vorkommen. Die Isolation der Nachrichten ist somit durch den Kommunikationsserver in der Regel gewährleistet.

Persistenz der Datenänderungen
Die Persistenz einer Datenänderung in einem der Subsysteme des KKS kann ein Kommunikationsserver nicht beeinflussen. Sie obligt ausschließlich den Subsystemen bzw. den in diesen Systemen eingesetzten DBMS. Um die Persistenz der Datenänderung aber für den Sender einer Nachricht zu gewährleisten, darf ein Kommunikationsserver erst dann eine Nachrichtenübertragung als erfolgreich abgeschlossen betrachten, wenn der Empfänger den Empfang der Nachricht **und** die Durchführung der mit der Nachricht verbundenen Datenbankoperationen bestätigt hat.

Betrachtet man die in dieser Arbeit analysierten Kommunikationsserver, so lassen sich mit diesen nur drei der vier Transaktionsbedingungen erfüllen:

1. Die Erhaltung der Konsistenz der Datenbestände durch entsprechende Transformationsregeln,
2. die Isolation der Nachrichten durch einfache Kommunikationsverbindungen sowie
3. die Persistenz der Datenänderungen in den Subsystemen durch eine entsprechende Synchronisierung des Nachrichtenaustausches.

Besteht eine Transaktion aus mehreren Nachrichten, so läßt sich die Atomarität mit den meisten Kommunikationsservern nicht gewährleisten. Dies gilt schon dann, wenn eine Nachricht mehrere Empfänger hat. Nach dem Routing stellen die einzelnen Nachrichtenkopien für die verschiedenen Empfänger selbständige Nachrichten dar. Die meisten Kommunikationsserver verwenden für die Übermittlung der Nachrichten zu den Empfängern getrennte Subsystem-Agenten. Die Überwachung der Atomarität der Transaktion müßte bei diesen Kommunikationsservern also durch die Routing-Komponente erfolgen, wozu keiner der Kommunikationsserver bisher in der Lage wäre.

Eine Ausnahme bietet der HL7-CONNECTION-SERVER^{TM}. Bei diesem Kommunikationsserver ist nur ein Subsystem-Agent für die Verteilung einer Nachricht an alle Empfänger zuständig (vgl. Abbildung 6.14 auf Seite 72). Dieser wäre prinzipiell in der Lage, die Atomarität einer Transaktion zu gewährleisten, wenn die DV-Systeme der Empfänger eine Rücknahme einer Nachricht mit den mit ihr verbundenen Datenänderungen ermöglicht. Da der HL7-CONNECTION-SERVER^{TM} aber für jede Nachricht einen solchen Subsystem-Agenten generiert, ist eine Isolation der Nachrichten nicht mehr gewährleistet.

Somit können mit keinem der analysierten Kommunikationsserver die vier Bedingungen an Transaktionen erfüllt werden. Sollen dennoch Transaktionskonzepte für die Nachrichtenübertragung zwischen den Subsystemen des KKS angewendet werden, müßte ein externes Modul die Aufgabe der Transaktionsüberwachung übernehmen. Ein solcher *Transaktionsmanager* würde den Kommunikationsserver dann weiterhin für das einsetzen, für das er ursprünglich konzipiert wurde: als Nachrichtenvermittlungsstelle.

Kommunikationsserver und föderierte DBMS

Autonome Datenbankmanagementsysteme lassen sich zu föderierten DBMS (FDBMS) zusammenfassen [SL90]. Innerhalb dieser Föderation kooperieren die einzelnen DBMS miteinander, bewahren aber ihre Autonomität.

In einem KKS bzw. KIS stellen die einzelnen EDV-Systeme der Abteilungen des Krankenhauses autonome Systeme dar. Viele von ihnen speichern Daten in Datenbanken, wie zum Beispiel die Befunde der untersuchten Patienten. Diese datenspeichernden bzw. datenerzeugenden Subsysteme des KKS lassen sich nun als autonome Datenbanksysteme auffassen. Somit ist auch der Einsatz eines FDBMS in einem Krankenhaus denkbar [Has96]. Die Datenbanksysteme der Föderation werden durch die datenspeichernden Subsysteme gebildet. Die Subsysteme, die selbst keine Daten speichern, die von anderen Systemen genutzt werden können, aber auf die Datenbestände der anderen Systeme zugreifen (z.B. die Rechnungsabteilung des Krankenhauses) können die externen Benutzer der Föderation darstellen.

Wie kann ein Kommunikationsserver nun ein FDBMS unterstützen? In einem föderierten Datenbanksystem bietet jedes einzelne DBMS über sogenannte **lokale Schemata** seine Daten der Föderation an. Aus diesen Schemata bildet das FDBMS **Komponentenschemata**, die für alle Datenbanksysteme der Föderation in einer einheitlichen Struktur vorliegen. Aus diesen werden wiederum durch Filterprozesse die **Exportschemata** der Datenbanken erzeugt. Das FDBMS bildet aus diesen Exportschemata ein oder mehrere **föderierte Schemata**, aus denen wiederum über Filterprozesse für die externen Benutzer der Föderation sogenannte **externe Schemata** zur Verfügung gestellt werden (vgl. auch Abbildung 8.1).

In einem KKS lassen sich die Kommunikationsprotokolle der einzelnen Subsysteme mit den lokalen Schemata der DBMS vergleichen. Ein Subsystem stellt anderen Systemen nur die Daten zur Verfügung, für deren Austausch im Kommunikationsprotokoll Nachrichten definiert sind. Die Abbildung der lokalen Schemata in die Komponenten- und Exportschemata würde der Abbildung einer Nachricht in ein für alle Subsysteme des KKS einheitliches Kommunikationsprotokoll entsprechen. Die hierfür notwendigen Transformationen können durch den Kommunikationsserver erfolgen. Die Generierung der lokalen Schemata aus den Datenbanken würde der Generierung der Nachrichten durch die Kommunikationsklienten der Subsysteme entsprechen. Diese Transformation würden somit die Kommunikationsklienten vornehmen. In Abbildung 8.1 wird eine mögliche Systemarchitektur eines FDBMS dargestellt.

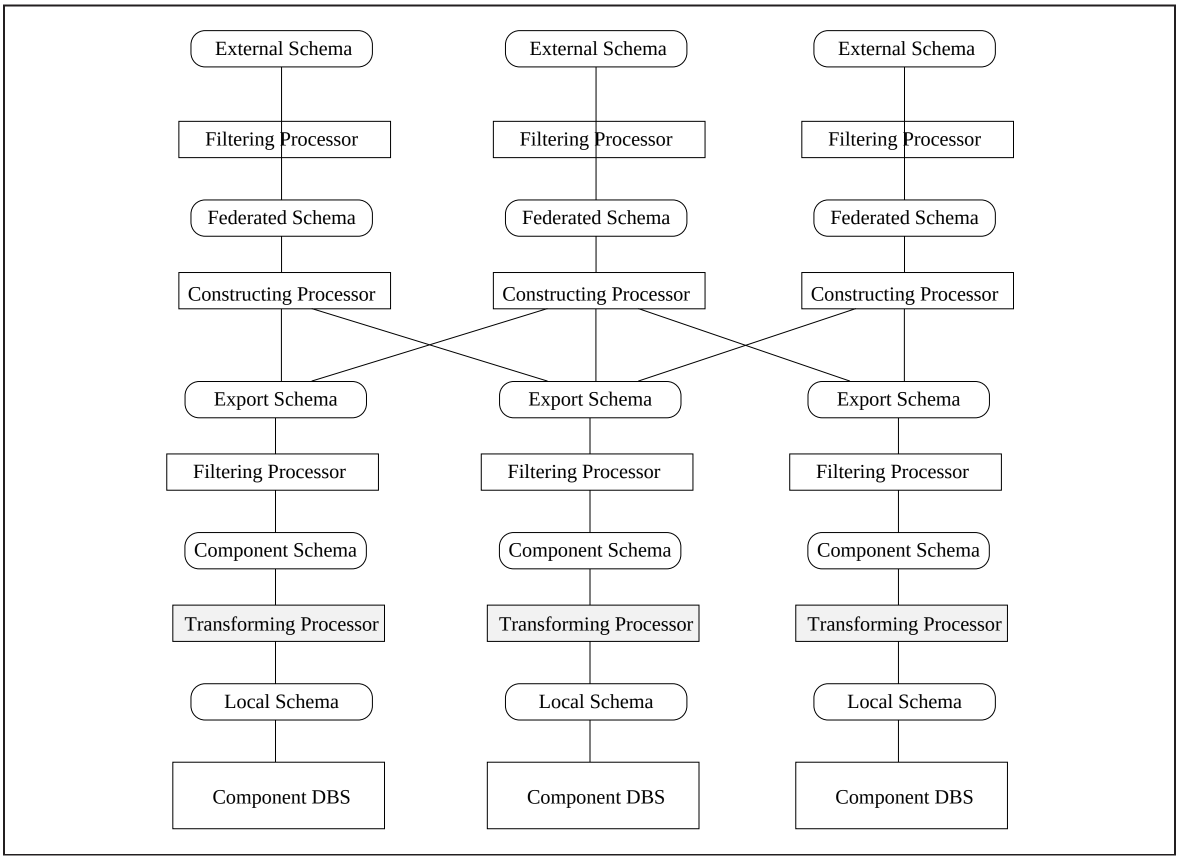


Abbildung 8.1: Darstellung der Systemarchitektur eines föderierten Datenbankmanagementsystems (entnommen aus [SL90]). Die schattiert dargestellten Prozesse könnten von einem Kommunikationsserver übernommen werden.

Sie ist [SL90] entnommen. Die in der Abbildung schattiert dargestellten Prozesse können, neben der eigentlichen Nachrichtenübertragung, von einem Kommunikationsserver übernommen werden.

Anhang A

Verwendete Notationen

A.1 OMT

OMT (*Object Modeling Technique*) ist eine Methode zur objektorientierten Analyse und Design von Software-Produkten. Sie wurde von Rumbaugh et al. entwickelt und wird ausführlich in [RBP+91] dargestellt. Für eine kurze Einführung sei auch auf [Krö96] verwiesen. Die in dieser Arbeit verwendeten Teile der Notation werden in den Abbildungen A.2 und A.1 wiedergegeben.

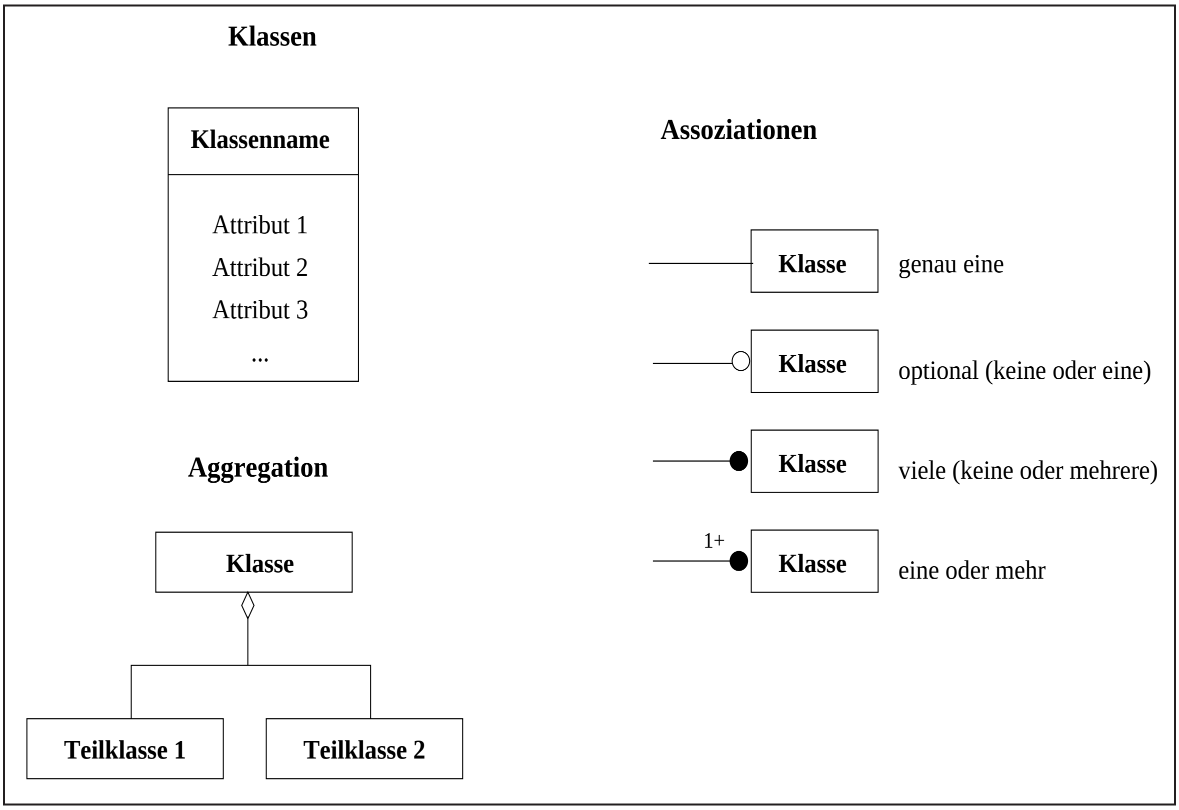


Abbildung A.1: OMT-Notation für die Objektmodelle (entnommen aus [RBP+91]).

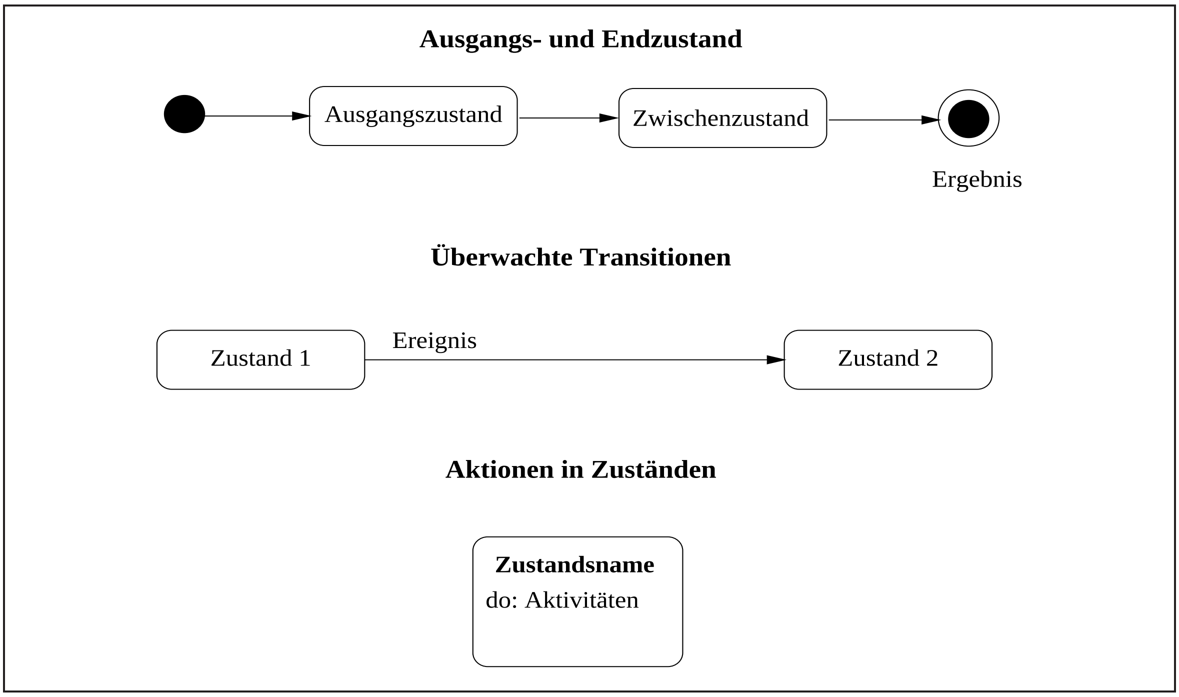


Abbildung A.2: OMT-Notation für die dynamische Modellierung (entnommen aus [RBP+91]).

A.2 Datenflußdiagramme

Die in dieser Arbeit dargestellten Datenflußdiagramme verwenden eine erweiterte Notation. Sie ist in der Tabelle A.1 dargestellt.

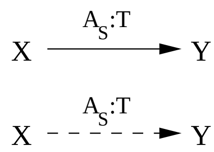
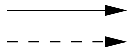
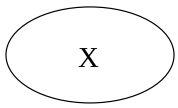
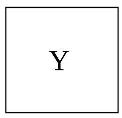
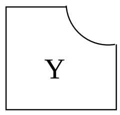
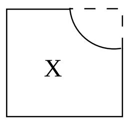
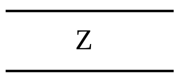
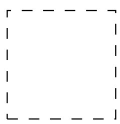
Komponente	**graphische Darstellung**	**Bedeutung**
Nachrichtenfluß		Die Nachricht `A` vom Typ `T` in der Struktur `S` wird von der Komponente X zur Komponente Y übertragen. Datenträger:  elektronische Verbindung Papier
Kontrollfluß		Kontrollfluß zwischen der Komponente X und Y.
Prozeß		Eine datenverarbeitende Komponente X. (Prozeß)
Quelle		Eine externe Komponente Y, aus der Daten entspringen.
Senke		Eine externe Komponente Y, zu der Daten fließen.
Quelle u. Senke		Eine externe Komponente, zu der Daten fließen und aus der Daten entspringen.
Speicher		Komponente Z, in der Daten gespeichert werden können.
Rechner		Der gestrichelte Kasten gibt die Grenzen eines Rechners an.

Tabelle A.1: Erweiterte Notation der Datenflußdiagramme.

Anhang B

Herstellernachweise zu den Kommunikationsservern

B.1 Cloverleaf™

Produktname: Cloverleaf™

Hersteller: Healthcare Communications, Inc., Dallas

Vertrieb: HealthComm Essen

Literatur: Handbuch zur Version 3.1 [Hea96]

B.2 DataGate™

Produktname: DataGate™

Hersteller: Software Technologie Corporation (STC), Arcadia, Kalifornien

Vertrieb: Gesellschaft für offene Systeme in der Medizin, Essen; Hinz Fabrik GmbH, Berlin

Literatur: Handbuch zur Version 2.6 und Version 3.0 [Sof95a, Sof95b]

B.3 HCS™

Produktname: Hospital-Communication-System (HCS™)

Hersteller: Gesellschaft für Softwareentwicklung mbH (SoftCon); Anstalt für kommunale Datenverarbeitung in Bayern (AKDB)

Vertrieb: Gesellschaft für Softwareentwicklung mbH (SoftCon), Oberhachingen

Literatur: Benutzerhandbuch zur Version 1.1 [Sof96]

B.4 HL7-Connection-Server™

Produktname: HL7-Connection-Server™

Hersteller: `prompt!` Medizinische Informationssysteme, Hamburg

Vertrieb:

Literatur: Informationsmaterial vom Hersteller [pro95]

B.5 pro7-System™

Produktname: pro7-System™

Hersteller: Abteilung Softwaretechnik des ICD e.V.; PRO DV Software GmbH; ROKD GmbH

Vertrieb: ICD e.V. Dortmund

Literatur: [Jun95, GS97]

Anhang C

Vergleich der Kommunikationsserver anhand des Anforderungskataloges

In diesem Abschnitt werden die fünf verschiedenen Kommunikationsserver anhand des Anforderungskataloges aus dem Kapitel 4 vergleichend gegenübergestellt. Für diesen Vergleich wurde der im Kapitel 4 erwähnte Fragenkatalog verwendet, der aus den Anforderungen und zu den Anforderungen gehörenden Fragen besteht, die die Umsetzung der Anforderungen in einem Kommunikationsserver klären sollen. Für alle betrachteten Kommunikationsserver wurde dieser Fragenkatalog bearbeitet. Die Beantwortung der Fragen wurde bei CLOVERLEAF^{TM} und DATAGATE^{TM} anhand der an den Uni-Kliniken in Münster installierten Versionen und bei den übrigen Produkten aus den von bzw. bei den Herstellern ausgefüllten Fragebögen ermittelt. Bei dem Kommunikationsserver der Firma `prompt!` (HL7-CONNECTION-SERVER^{TM}) konnten die Fragen nur über das vorliegende Informationsmaterial bearbeitet werden.

Die Auswertung der Fragebögen wird im folgenden tabellarisch dargestellt. Dabei werden jeweils die Anforderungen mit den zugehörigen Fragen zu einer Tabelle zusammengefaßt. Im Fragenkatalog waren die meisten Fragen sehr offen gehalten, um die Beantworter nicht durch Vorgaben einzuschränken. Um die Übersichtlichkeit des Vergleichs zu erhöhen, wurden diese offenen Fragen überwiegend in geschlossene Fragen mit Auswahllisten umgewandelt, die nur eine Beantwortung mit „Ja" (in den Tabellen durch ✔dargestellt), „Nein" (–) oder „Unbekannt" (?) erlauben. Die Vorgaben für diese Fragen, also die Einträge der Auswahllisten, wurden dabei jeweils aus den Antworten zu den Fragen gewonnen. Nur dort, wo eine Auswahlliste einen zu großen Informationsverlust bedeutet hätte, werden die Fragen textuell beantwortet. Bei diesen Fragen ist in den Spalten für die einzelnen Kommunikationsserver ein Buchstabe eingetragen (z.B.: „a"). Dieser Buchstabe verweist auf die textuelle Beantwortung der Frage, die jeweils am Ende einer Tabelle aufgeführt wird. Anmerkungen zu den einzelnen Fragen werden über Fußnoten vorgenommen.

Die Namen der analysierten Kommunikationsserver werden in den Tabellen wie folgt abgekürzt:

CL	CLOVERLEAF^{TM} von HCI
CS	HL7-CONNECTION-SERVER^{TM} von `prompt!`
DG3.0	DATAGATE^{TM} Version 3.0 von STC
DG3.1	DATAGATE^{TM} Version 3.1 von STC
HCS	*Hospital Communication System* (HCS^{TM}) von SOFTCON
Pro7	PRO7-SYSTEM^{TM} von ICD, PRO DV und ROKD

C.1 Anforderungen an die Serverkomponente

C.1.1 Transportprotokolle

	CL	CS	DG3.0	DG3.1	HCS	Pro7
Anf. 1.1 Unterstützung standardisierter Transportprotokolle.						
1 Welche dateilosen Transportprotokolle für die Übertragung der Nachrichten zu den Subsystemen werden standardmäßig angeboten?						
DECnet	✔	–	–[1]	–[1]	–	–[1]
LU-Gruppe	✔[2]	–	–	–	–	–
Async.	✔	–	–	–	–	–
TCP-Sockets	✔	✔	–	–	✔	–
Pipes	–	–	–	–	✔	–
RPC	–	–	–	–	–	–
weitere	–	–	–	–	–	–
2 Können Nachrichten aus Dateien eingelesen oder in Dateien geschrieben werden?	✔	✔	✔	✔	✔	✔
3 Werden FTP oder ähnliche Transportprotokolle standardmäßig angeboten?	✔	–	–	–	–	–
4 Welche Transportprotokolle werden für den Nachrichtenaustausch zwischen den Modulen des Kommunikationsservers verwendet?						
TCP-Sockets	–	?	✔	✔	✔	–
Pipes	–	?	–	–	✔	–
RPC	?	?	–	–	✔	✔
weitere	–	?	–	–	–	–

	CL	CS	DG3.0	DG3.1	HCS	Pro7
Anf. 1.2 Unterstützung verschiedener Nachrichtendarstellungsformen und Sicherungsmaßnahmen.						
1 Welche Darstellungsformen der Nachrichten werden standardmäßig angeboten?						
Längenkodierung	✔	?	–[3]	–[3]	–	–[3]
Envelope	–	?	–	–	✔	–
fixe Länge	✔	?	–	–	✔	–

...

[1] In den Subsystem-Agenten können vom Anwender beliebige Transportprotokolle implementiert werden.
[2] LU2, LU3, LU6.2 APPC
[3] In den Subsystem-Agenten können vom Anwender beliebige Darstellungsformen implementiert werden.

…	CL	CS	DG3.0	DG3.1	HCS	Pro7
weitere	–	?	–	–	–	–
2 Welche Sicherungsmechanismen werden standardmäßig angeboten?	a	a	–[4]	–[4]	a	–[4]

a Es sind Acknowledgements auf Nachrichtenebene möglich.

	CL	CS	DG3.0	DG3.1	HCS	Pro7
Anf. 1.3 Konfigurierung der Transportprotokolle durch den Anwender.						
1 In welchem Format werden die Konfigurationstabellen für die Transportprotokolle gespeichert?						
ASCII-Datei	✔	–	✔	✔	–	–[5]
Datenbank	–	✔	–	–	✔	–
andere Speicherart	–	–	–	–	–	–
2 Existiert ein spezieller Editor für die Auswahl und Konfigurierung der Protokolle?	✔	?	–	–	✔	–

	CL	CS	DG3.0	DG3.1	HCS	Pro7
Anf. 1.4 Unterstützung weiterer Transportprotokolle.						
1 Können vom Anwender weitere Sicherungsmechanismen und Nachrichtendarstellungsformen definiert werden?	✔	–	✔	✔	–	✔
Wenn ja, wie werden sie implementiert?	b		a	a		a
2 Können eigene Transportprotokolle im Kommunikationsserver verwendet werden?	✔	–	✔	✔	–	✔
Wenn ja, wie werden sie implementiert?	b		a	a		a

a Implementierung spezieller Subsystem-Agenten in „C".

b Implementierung spezieller Protokolle in TCL.

[4] In den Subsystem-Agenten können vom Anwender beliebige Sicherungsmechanismen implementiert werden.
[5] Die Speicherform der Konfigurationen sind von der Implementierung der Subsystem-Agenten abhängig.

C.1.2 Nachrichtenidentifikation

		CL	CS	DG3.0	DG3.1	HCS	Pro7
Anf. 1.5	Konfigurierbare Nachrichtenidentifikation zur Ermittlung der Struktur einer Nachricht.						
1	Wie werden die Nachrichtentypen identifiziert?						
	Bei proprietären Protokollen	b	–[6]	a	a	c	c
	Bei standardisierten Protokollen	d	d	a	?	d	d
2	Welche Funktionen zur Implementierung der Nachrichtenidentifizierung werden angeboten?						
	if-Bedingungen (if-then-else, case)	–[7]	–[8]	✔	✔	–[8]	–[8]
	Schleifen (loop)	–	–	✔	✔	–	–
	Lookup-Tabellen	–	–	✔	✔	–	–
	Auswerten von Datenfeldern	–	–	✔	✔	–	–
	weitere Funktionen	–	–	e	?	–	–
3	Werden bei der Identifikation auch extern implementierte Funktionen unterstützt?	✔	–	–	?	–	–

a Schlüsselwortmethode: Eine Identifizierungsfunktion wertet bestimmte Datenfelder einer Nachricht aus und liefert ein Schlüsselwort zurück, das der Nachricht zugeordnet wird.

b Angabe von Offset und Länge des Datenfeldes der Nachricht, in dem der Nachrichtentyp codiert ist oder mit der Schlüsselwortmethode über TCL-Skripte.

c Die Nachrichten werden in entsprechende HL7-Nachrichten umgewandelt.

d Über den im Standard vorgesehenen Nachrichtentyp.

e Funktionen zur Überprüfung der HL7-Konformität der Struktur einer Nachricht.

		CL	CS	DG3.0	DG3.1	HCS	Pro7
Anf. 1.6	Unterstützung einer automatisierten Nachrichtenidentifikation bei standardisierten Kommunikationsprotokollen.						
1	Bei welchen standardisierten Kommunikationsprotokollen wird eine automatisierte Nachrichtenidentifikation unterstützt?						
	HL7	✔	✔	–	?	✔	✔

…

[6] Es werden ausschließlich HL7-Nachrichten unterstützt.

[7] Implementierung der Identifikationen über TCL-Skripte möglich

[8] Da alle Nachrichten in ein internes Kommunikationsprotokoll transformiert werden, erübrigt sich die Implementierung von Identifizierungsregeln. Die Nachrichten des internen Kommunikationsprotokolls werden vom Kommunikationsserver automatisch identifiziert.

...	CL	CS	DG3.0	DG3.1	HCS	Pro7
EDIFACT	✔	–	–	?	–	–
weitere Standards	X12	–	–	?	–	–

	CL	CS	DG3.0	DG3.1	HCS	Pro7
Anf. 1.7 Unterstützung einheitlicher Bezeichner für gleiche Nachrichtentypen in unterschiedlichen Kommunikationsprotokollen.						
1 Sind einheitliche Bezeichner für Nachrichten mit dem gleichen semantischen Inhalt in unterschiedlichen Kommunikationsprotokollen möglich?	✔	–[9]	–	–	–[10]	–[10]

	CL	CS	DG3.0	DG3.1	HCS	Pro7
Anf. 1.8 Unterstützung von Hierarchien bei Nachrichtentypen.						
1 Werden Hierarchien bei Nachrichtentypen unterstützt?	–	–	✔	✔	–	–
Wenn ja, wie lassen sich die Hierarchien bilden?			a	a		

a Jeder Nachricht können beliebig viele Nachrichtentypen zugeordnet werden.

C.1.3 Bestimmung der Empfänger der Nachrichten

	CL	CS	DG3.0	DG3.1	HCS	Pro7
Anf. 1.9 Unterstützung einer statischen und dynamischen Ermittlung der Empfänger einer Nachricht.						
1 Welche Routing-Methoden werden angeboten?						
Statisches Routing	✔[11]	–	–	–	–	–
Routing über den Nachrichtentyp	✔	✔	–	?	✔	✔
Routing über Auswertung einzelner Datenfelder	✔[12]	–	–	–	✔[13]	✔[13]
Routing über Schlüsselwörter	✔	–	✔	✔	–	–

[9] Es werden nur HL7-Nachrichten akzeptiert.
[10] Alle Nachrichten werden auf HL7-Nachrichten abgebildet.
[11] Das Routing kann vollständig durch TCL-Skripte erfolgen.
[12] Angabe von Offset und Länge des Datenfeldes in der Nachricht.
[13] Es wird das Feld im `MSH`-Segment der HL7-Nachricht ausgewertet, in dem der Empfänger der Nachricht eingetragen ist.

		CL	CS	DG3.0	DG3.1	HCS	Pro7
Anf. 1.10	Leichte Definition der Routing-Tabellen.						
1	Existiert ein spezieller Editor für die Definition der Routing-Tabellen?	✔	✔	✔	✔	✔	✔
2	In welchem Format werden die Routing-Tabellen abgelegt?						
	ASCII-Datei	✔	–	✔	✔	–	–
	Datenbank	–	✔	–	–	✔	✔
	andere Speicherformen	–	–	–	–	–	–

		CL	CS	DG3.0	DG3.1	HCS	Pro7
Anf. 1.11	Konfigurierbare, graphische Darstellung der Routing-Informationen						
1	In welcher Form werden die Routing-Informationen dargestellt?						
	als Tabelle	–	–[14]	–	–	✔	✔
	als Netzwerkdarstellung	✔	–	–	–	–	–
	weitere Darstellungsformen	–	–	a	a	–	–
2	Läßt sich die Darstellung der Routing-Informationen konfigurieren?	✔	–	–	–	–	–

a In Form von Listen, in denen die Subsysteme durch Icons dargestellt werden. In einer Liste werden alle sendenden Subsysteme dargestellt. Zu dem in dieser Liste markierten Subsystem werden in einer weiteren Liste alle Subsysteme angezeigt, die von dem markierten Subsystem Nachrichten erhalten. Auch in dieser Liste läßt sich ein Subsystem markieren. Zu der so ausgewählten Kommunikationsverbindungen werden in einer dritten Liste die ausgetauschten Nachrichtentypen angezeigt.

C.1.4 Transformation der Nachrichtenstrukturen

		CL	CS	DG3.0	DG3.1	HCS	Pro7
Anf. 1.12	Leichte Implementierung der Funktionen zur Transformation der Nachrichten in andere Nachrichtenstrukturen.						
1	Wird ein spezieller Editor für die Implementierung der Transformationen angeboten?	✔	–[15]	–	✔	✔	–

...

[14] Die Routing-Informationen lassen sich nicht graphisch darstellen.
[15] Da nur HL7-Nachrichten unterstützt werden, ist eine Definition von Transformationsregeln nicht erforderlich.

...

	CL	CS	DG3.0	DG3.1	HCS	Pro7
2 In welchem Format werden die Transformationen gespeichert?						
ASCII-Datei	✔	–	✔	✔	–	–
Datenbank	–	–	–	–	✔	–
andere Speicherform	–	–	–	–	–	a

a Die Transformationen werden in einer sogenannten PDL (angelehnt an C) implementiert. Aus dem Quellcode werden die Konverter (ein Prozeß des Verarbeitungsmoduls) generiert.

	CL	CS	DG3.0	DG3.1	HCS	Pro7
Anf. 1.13 Unterstützung einer automatisierten Transformation bei standardisierten Kommunikationsprotokollen.						
1 Wird eine automatisierte Transformation bei standardisierten Kommunikationsprotokollen angeboten?	–[16]	✔	–[16]	–[16]	–[16]	–[17]
2 Wenn ja, zwischen welchen Protokollen?		a				

a HL7 2.1 ↔ HL7 2.2

C.1.5 Wertkonvertierungen

	CL	CS	DG3.0	DG3.1	HCS	Pro7
Anf. 1.14 Bereitstellung von allgemeinen und speziellen Konvertierungsfunktionen.						
1 Welche Konvertierungsfunktionen werden angeboten?						
Datumkonvertierung	✔	–	✔	✔	–	✔
Mathematische Funktionen	✔	–	✔	✔	–	✔
String-Funktionen (Concat, Insert, etc.)	✔	–	✔	✔	–	✔
Einfügen von konstanten Werten	✔	–	✔	✔	–	✔
Kopieren von Datenfeldern	✔	–	✔	✔	✔	✔
Kopieren von ganzen Segmenten	✔	–	✔	✔	–	✔
weitere Funktionen	✔[18]	–	–	?	–	✔[19]

...

[16] Läßt sich über vorgefertigte Transformationstabellen realisieren.
[17] Läßt sich über spezielle Konverter realisieren.
[18] Die Transformationen können komplett durch TCL-Skripte vorgenommen werden.
[19] Die Wertkonvertierungen werden in C programmiert.

...	CL	CS	DG3.0	DG3.1	HCS	Pro7
2 Werden spezielle Funktionen zur Konvertierung multimedialer Daten angeboten?	–	–	–	–	–	–
3 Wird der Aufruf extern implementierter Konvertierungsfunktionen unterstützt?	✔	–	✔	✔	✔	✔

	CL	CS	DG3.0	DG3.1	HCS	Pro7
Anf. 1.15 Unterstützung von Lookup-Tabellen.						
1 Können Lookup-Tabellen zur Konvertierung einzelner Werte verwendet werden?	✔	✔[20]	✔	✔	✔	✔[20]
Wenn ja						
2 Wird ein spezieller Editor für die Definition der Lookup-Tabellen angeboten?	✔	✔	–	✔	✔	✔
3 Können die Lookup-Tabellen auch bidirektional angelegt werden?	✔	✔	–	–	✔	✔

	CL	CS	DG3.0	DG3.1	HCS	Pro7
Anf. 1.16 Bereitstellung von Kontrollstrukturen für die Implementierung der Transformationen.						
1 Welche Kontrollstrukturen werden bei der Implementierung der Transformationen angeboten?						
if-Konstrukte (if-then-else, case, etc.)	–	–	✔	✔	–	–
Schleifen (for, while, etc.)	✔	✔	✔	✔	–	–
weitere Kontrollstrukturen	✔[21]	–	–	?	–	✔[22]

[20] Lookup-Tabellen werden für die Schlüsselkonvertierungen einzelner HL7-Felder verwendet.

[21] Die Transformationen können auch über TCL-Skripte vorgenommen werden. Dort können dann alle in TCL verfügbaren Kontrollstrukturen verwendet werden.

[22] Die Transformationen werden in C programmiert.

C.2 Anforderungen an die Steuerungskomponente

C.2.1 Übertragung einzelner Nachrichten

	CL	CS	DG3.0	DG3.1	HCS	Pro7
Anf. 2.1 Unterstützung verschiedener Nachrichtenlängen.						
1 Wieviele Gigabyte kann eine Nachricht maximal umfassen?	4	?	_[23]	_[23]	_[23]	_[23]

	CL	CS	DG3.0	DG3.1	HCS	Pro7
Anf. 2.2 Unterstützung einer synchronisierten Übertragung von Nachrichten zwischen zwei Subsystemen.						
1 Kann eine synchronisierte Übertragung einer Nachricht zwischen zwei oder mehreren Subsystemen erfolgen?	✔	–	✔	✔	–	✔
Wenn ja, welche Maßnahmen zur Synchronisation werden angeboten?	b		a	a		c

a Es werden keine besonderen Maßnahmen angeboten. Eine Synchronisation zweier Subsysteme läßt sich mit bidirektionalen *Communication Clients* (Subsystem-Agenten) mit Hilfe von Acknowledgements auf Nachrichtenebene realisieren.

b Verwaltung von Replies auf Nachrichtenebene durch den Kommunikationsserver.

c Eine Synchronisation ist auf Nachrichtenebene zwischen den Subsystem-Agenten möglich.

	CL	CS	DG3.0	DG3.1	HCS	Pro7
Anf. 2.3 Vergabe von Prioritäten an die Nachrichten.						
1 Lassen sich Prioritäten für einzelne Nachrichten vergeben?	✔	–	–	–	–	–
2 Wenn ja, wird die Übermittlung einer Nachricht mit sehr geringer Priorität durch spezielle Maßnahmen gewährleistet?	–	–	–	–	–	–

[23] Durch den Hauptspeicherplatz beschränkt.

C.2.2 Verwaltung komplexer Aktionen

	CL	CS	DG3.0	DG3.1	HCS	Pro7
Anf. 2.4 Eigenständige Generierung von Nachrichten.						
1 An welchen Stellen des Verarbeitungsprozesses können Nachrichten durch den Kommunikationsserver generiert werden?						
Empfang der Nachrichten	✔	✔	✔	✔	–	✔
Nachrichtenidentifikation	✔	–	–	–	–	–
Routing	✔	–	–	–	–	–
Nachrichtentransformation	✔	–	–	–	–	–
Senden der Nachrichten	✔	–	✔	✔	–	✔
weitere Stellen	a	–	–	–	b	–

a Über TCL-Skripte an beliebigen Stellen im Verarbeitungsprozeß.

b Es können HL7-Acknowledgements beim Auftreten eines Fehlers in der Nachrichtenverarbeitung erzeugt werden.

	CL	CS	DG3.0	DG3.1	HCS	Pro7
Anf. 2.5 Zusammenfügen einzelner Nachrichten von unterschiedlichen Sendern zu einer Nachricht.						
1 Lassen sich mehrere Nachrichten von möglicherweise verschiedenen Sendern zu einer Nachricht zusammensetzen?	✔[24]	–	✔	✔	–	✔[25]

	CL	CS	DG3.0	DG3.1	HCS	Pro7
Anf. 2.6 Unterstützung von komplexen Aktionen durch den Kommunikationsserver.						
1 Werden komplexe Aktionen vom Kommunikationsserver besonders unterstützt?	–	✔	–	–	–	–
Wenn ja, wie sieht diese Unterstützung aus.		a				

a Es können beliebige komplexe Aktionen (als „Transaktionen" bezeichnet) definiert werden. Die Implementierung kann allerdings nur vom Hersteller vorgenommen werden.

[24] Muß über TCL-Skripte implementiert werden.
[25] Muß in den Subsystem-Agenten in C implementiert werden.

C.2.3 Anforderungen an das Fehlermanagement und die Persistenzsicherung

Anf. 3.1 Effektive Behandlung von fehlerhaften Nachrichten.	CL	CS	DG3.0	DG3.1	HCS	Pro7
1 Lassen sich in der Verarbeitungskomponente Nachrichten auf Vollständigkeit gemäß der Definition der Nachrichtenstruktur im Kommunikationsprotokoll überprüfen?	✔	–	✔	✔	✔	✔
2 Wird eine automatisierte Kontrolle zwingender Segmente in standardisierten Kommunikationsprotokollen unterstützt?						
bei HL7	✔	–	–	?	✔	✔
bei EDIFACT	✔	–	–	?	–	–
bei weiteren Protokollen	X12	–	–	?	–	–
3 Kann in der Verarbeitungskomponente eine Überprüfung der Datentypen von Datenfeldern einer Nachricht durchgeführt werden?	✔	–	✔	✔	–	✔
4 Kann der Sender vom Kommunikationsserver über Fehler in einer Nachricht vom Kommunikationsserver informiert werden?	✔	–	–	✔	✔	✔

Anf. 3.2 Speicherung von fehlerhaften Nachrichten in einer Fehler-Datenbank.	CL	CS	DG3.0	DG3.1	HCS	Pro7
1 Lassen sich fehlerhafte Nachrichten im Kommunikationsserver speichern?	✔	✔[26]	✔	✔	✔	✔[26]
Wenn ja						
2 In welcher Form werden sie gespeichert?						
ASCII-Datei	–	–	✔	✔	–	–
Datenbank	✔	✔	–	–	✔	✔
andere Speicherform	–	–	–	–	–	–
3 Wird ein Modul für die Verwaltung der Fehler-Datenbank bereitgestellt?	✔	–	–	–	✔	✔
4 Wenn ja, welche Funktionen bietet es?						
Show	✔	–	–	–	✔	✔
Edit	–	–	–	–	✔	✔
Resend	–	–	–	–	✔	✔

...

[26] Es werden grundsätzlich alle Nachrichten in der Datenbank gespeichert.

...	CL	CS	DG3.0	DG3.1	HCS	Pro7
Delete	✔	–	–	–	✔	✔
weitere	a	–	–	–	b	–

a Speichern der Nachrichten in ASCII-Dateien zur Verarbeitung mit einem weiteren Tool, mit dem sie erneut versandt werden können.

b Erzeugung und Senden von HL7-Acknowledgements an den Sender der fehlerhaften Nachricht.

	CL	CS	DG3.0	DG3.1	HCS	Pro7
Anf. 3.3 Bereitstellung von Queues zur persistenten Haltung von Nachrichten.						
1 An welcher Stelle des Verarbeitungsprozesses einer Nachricht werden Queues unterhalten?						
Empfangen der Nachricht	✔	?	–[27]	–[27]	–	✔
Nachrichtenidentifikation	–[28]	?	–	–	–	–
Routing	✔	?	–	–	–	–
Nachrichtentransformation	✔	?	–	–	–	–
Versenden der Nachricht	✔	?	✔	✔	–	✔
weitere Stellen	–	?	–	–	a	–
2 Existieren getrennte Queues für die einzelnen Kommunikationsverbindungen?	✔	?	✔	✔	✔	✔
3 Auf welchem Speichermedium lassen sich die Queues halten?						
Datei, Pipe, etc.	✔	?	✔	✔	✔	–
Datenbank	–	?	–	–	–	✔
RAM	–	?	✔	✔	–	–
4 Werden Verschlüsselungsmethoden für das Queuing der Nachrichten unterstützt?	–	?	✔	✔	–	–
Falls nein, wie werden die Queues dann vor einem unbefugten Zugriff geschützt?	b	?			b	c
5 Läßt sich für einzelne Nachrichten eine maximale Wartezeit in der Queue festlegen?	–	?	–	–[29]	–	–
6 Wann werden die Nachrichten in der Queue wieder gelöscht?	d	?	d	d	d	e

[27] Kann im entsprechenden Subsystem-Agenten implementiert werden.
[28] Identifikation und Routing erfolgen in einem Schritt.
[29] Es existiert ein Modul zur Verwaltung der Inhalte der Queues durch den Administrator. Dort können Nachrichten gelöscht, innerhalb der Queue mit Prioritäten versehen oder die Reihenfolge der Nachrichten verändert werden.

a In den Verarbeitungsmodulen wird jeweils eine Queue für alle angeschlossenen Subsystem-Agenten unterhalten.

b Durch die Sicherungsmaßnahmen, die das Betriebsystem bietet.

c Durch das DBMS.

d Sobald der Empfänger den korrekten Empfang der Nachricht bestätigt hat.

e Die Nachrichten müssen in der Datenbank explizit vom Administrator gelöscht werden. Ist der Speicherplatz der Datenbank verbraucht, werden keine weiteren Nachrichten angenommen.

Anf. 3.4 Unterstützung von alternativen Kommunikationsverbindungen zu Subsystemen.	CL	CS	DG3.0	DG3.1	HCS	Pro7
1 Werden alternative Verbindungen zu einem Subsystem unterstützt, falls eine Verbindung ausgefallen ist?	✔	–	–[30]	–[30]	–	–[30]

Anf. 3.5 Bereitstellung von Recover-Mechanismen für den Fall eines Systemabsturzes.	CL	CS	DG3.0	DG3.1	HCS	Pro7
1 Wie wird die Persistenz der Nachrichten im Kommunikationsserver bei einem Systemabsturz gewährleistet?	b	d	a	a	c	e

a Erst wenn eine Nachricht in die Queues der Subsystem-Agenten der Empfänger geschrieben wurde, erhält der Sender über seinen Subsystem-Agenten ein positives Acknowledgement. Eine Nachricht wird erst dann aus einer Queue gelöscht, wenn der Subsystem-Agent ein positives Acknowledgement vom Empfänger erhalten hat.

b Die Nachrichten werden kontinuierlich nach jedem Verarbeitungsschritt in einer Datenbank gespeichert.

c Handshaking zwischen den Modulen; Abspeichern der Nachrichten in Sicherungsdateien und Nachsenden durch den Administrator.

d Alle Nachrichten werden über definierte komplexe Aktionen ausgetauscht. Diese werden hier als „Transaktionen“ bezeichnet. Der Ausführungsstand der komplexen Aktionen wird an definierten Stellen in der Datenbank gesichert.

e Jede Nachricht wird direkt nach dem Empfang in der Datenbank gespeichert.

[30] Kann durch spezielle Subsystem-Agenten realisiert werden.

	CL	CS	DG3.0	DG3.1	HCS	Pro7
Anf. 3.6 Unterstützung von Maßnahmen zur Ausfallsicherheit des Kommunikationsservers.						
1 Welche Maßnahmen zur Sicherung der Kommunikationsverbindungen bei Ausfall des Kommunikationsservers werden unterstützt?	a	a	a	a	b	a

a Alle Sicherungskonzepte des Hardware-Herstellers.

b Alle Sicherungskonzepte des Hardware-Herstellers, sowie Verteilung der Module auf unterschiedliche Rechner.

C.3 Anforderungen an die Überwachungskomponente

C.3.1 Logging und Nachrichtenarchivierung

	CL	CS	DG3.0	DG3.1	HCS	Pro7
Anf. 4.1 Konfigurierbares Logging für alle Schritte des Bearbeitungsprozesses einer Nachricht.						
1 Läßt sich der Umfang der Log-Informationen konfigurieren?	✔	?	✔	✔	✔	–[31]
2 Lassen sich für die Kommunikationsverbindungen getrennte Log-Dateien erstellen?	–	?	✔	✔	✔	–
3 Läßt sich der Umfang der Log-Informationen für jede Verbindung getrennt konfigurieren?	✔	?	✔	✔	✔	–
4 In welcher Form werden die Log-Dateien gespeichert?						
ASCII-Datei	✔	–	✔	✔	✔	–
Datenbank	–	✔	–	–	–	✔
andere Speicherform	–	–	–	–	–	–

[31] Es werden keine eigenständige Log-Dateien erzeugt. Die Informationen über den Verarbeitungsprozeß werden in speziellen Feldern der internen Darstellung der Nachrichten mit diesen in der Datenbank abgelegt.

	CL	CS	DG3.0	DG3.1	HCS	Pro7
Anf. 4.2 Automatische Verwaltung der Log-Dateien durch den Kommunikationsserver.						
1 Werden die Log-Dateien vom Kommunikationsserver verwaltet?	–	–	–	–	–	–
2 Wenn ja, wie werden sie verwaltet?	–	–	–	–	–	–

	CL	CS	DG3.0	DG3.1	HCS	Pro7
Anf. 4.3 Bereitstellung eines Moduls für die Auswertung der Log-Dateien.						
1 Wird der Anwender bei der Auswertung der Log-Dateien vom Kommunikationsserver unterstützt?	–	✔	✔	✔	–	✔
Wenn ja, wie sieht diese Unterstützung aus?		b	a	a		c

a Farbliche Hervorhebung der unterschiedlichen Eintragstypen (Error, Warning, Info, etc.); Such- und Filterfunktionen für Fehlermeldungen.

b Die Log-Informationen können für Nachrichtenklassen (Patientendaten, Aufträge, Befunde, etc.) getrennt angezeigt werden.

c Es werden Filterfunktionen angeboten, die nur die Log-Informationen zu Nachrichten bestimmter Subsysteme anzeigen.

	CL	CS	DG3.0	DG3.1	HCS	Pro7
Anf. 4.4 Möglichkeit zur Speicherung von Nachrichteninhalten.						
1 Lassen sich Nachrichteninhalte speichern?	✔	✔	✔	✔	✔	✔
Wenn ja						
2 Können die gespeicherten Nachrichten eindeutig einem Sender zugeordnet werden?	✔	✔	✔	✔	✔	✔
3 In welchem Format werden die Nachrichten gespeichert?						
ASCII-Datei	✔	–	✔	✔	✔	–
Datenbank	✔	✔	–	–	–	✔
andere Speicherform	–	–	–	–	–	–
4 Werden die Nachrichteninhalte auch in die Log-Dateien aufgenommen?	✔	–	–	–	✔	–[32]
Wenn ja, kann dies auch unterbunden werden?	✔				–	

[32] Die Log-Informationen sind untrennbar mit den Nachrichten verbunden.

	CL	CS	DG3.0	DG3.1	HCS	Pro7
Anf. 4.5 Bereitstellung eines Moduls für die Verwaltung der gespeicherten Nachrichten.						
1 Existiert ein Modul zur Verwaltung der gespeicherten Nachrichten?	✔	✔	✔	✔	✔	✔
Wenn ja						
2 Welche Funktionen bietet es an?						
Show	✔	✔	✔	✔	✔[33]	✔
Resend	✔	–	✔	✔	✔	✔
Edit	–	–	✔	✔	–	✔
Delete	✔	–	✔	✔	✔	✔
weitere Funktionen	–	–	a	?	–	–
3 Besitzt es eine graphische Oberfläche?	✔	✔	✔	✔	✔	✔

a Aktivieren/Deakivieren der Nachrichtenarchivierung getrennt für jedes Subsystem.

C.3.2 System-Monitor

	CL	CS	DG3.0	DG3.1	HCS	Pro7
Anf. 4.6 Darstellung der Kommunikationsverbindungen im System-Monitor.						
1 Wie lassen sich die Kommunikationsverbindungen im System-Monitor darstellen?						
Tabelle	–	–	✔	✔	–	✔
Netzwerk	✔	–	–	–	–	–
weitere Darstellungsformen	–	–	–	–	a	–
2 Läßt sich die Darstellung konfigurieren?	✔	–	–	–	–	–
3 Wie werden die Betriebszustände ('Verbindung hergestellt', 'keine Verbindung' etc.) der Kommunikationsverbindungen dargestellt?						
textuell	✔	–	✔	✔	✔	✔
graphisch (z.B. farblich hervorgehoben, blinkend)	✔	–	✔	✔	–	–
akustisch	–	–	✔	✔	–	–
weitere Darstellungsformen	–	–	–	–	–	–

[33] Eine Einsicht der Nachrichten wird nur vom System-Monitor bei der temporären Speicherung der Nachrichten (Mithören) in der Datenbank angeboten.

a In Form von Listen: Bei Auswahl eines Verarbeitungsmoduls in der ersten Liste werden alle an dieses Modul angeschlossene Subsystem-Agenten in einer zweiten Liste angezeigt.

	CL	CS	DG3.0	DG3.1	HCS	Pro7
Anf. 4.7 Bereitstellung von Funktionen zur getrennten Steuerung der Module und Verbindungen des Kommunikationsservers.						
1 Welche Funktionen zur Steuerung der Verbindungen stehen zur Verfügung?						
Verbindungsaufbau	✔	–	✔	✔	✔	✔
Verbindungsabbau	✔	–	✔	✔	✔	✔
Anhalten der Nachrichtenübertragung	✔	–	✔	✔	–	✔
Reload der Konfigurationen	✔	–	✔	✔	✔	–
weitere Funktionen	a	–	–	?	b	–
2 Lassen sich die Verbindungen getrennt steuern?	✔	–	✔	✔	✔	✔
3 Lassen sich die Log-Dateien vom System-Monitor aus einsehen?	✔	✔	✔	✔	–	–
4 Lassen sich die gesendeten Nachrichten vom System-Monitor aus einsehen?	–[34]	✔	✔	✔	✔	–[34]
5 Existiert eine 'Kommandozeilen'-Version des System-Monitors, die ohne graphische Oberfläche auskommt?	✔	?	✔	✔	✔	–
Wenn ja, welche Funktionalität bietet sie?	c		c	c	d	

a Aktivieren und Deaktivieren der Nachrichtenspeicherung. Resend von gespeicherten Nachrichten.

b Ein-/Ausschalten der Nachrichtenarchivierung; Ping an beliebige Subsystem-Agenten; Aktivierung von Statistiken; Nachversenden von Nachrichten.

c Alle Funktionen der GUI-Version.

d Starten/Stoppen der Module; Anfordern von Statusmeldungen von den Modulen.

	CL	CS	DG3.0	DG3.1	HCS	Pro7
Anf. 4.8 Aufstellung und Verwaltung von Statistiken über verschiedene Parameter.						
1 Über welche Parameter werden vom Kommunikationsserver Statistiken aufgestellt?						
Nachrichtendurchsatz gesamt	–	–	✔	✔	–	–
Durchsatz spezieller Nachrichtentypen	–	–	–	–	✔	–

...

[34] Für die Verwaltung der gespeicherten Nachrichten existiert ein eigenes Tool.

...

		CL	CS	DG3.0	DG3.1	HCS	Pro7
	Länge der Queues	–	–	✔	✔	–	–
	Festplatten-Speicherplatz	–	–	–	–	–	–
	CPU-Auslastung	✔	–	–	–	–	–
	Netzauslastung	–	–	–	–	–	–
	weitere Statistiken	a	–	–	–	–	–
2	Wie werden die Meßwerte dargestellt?						
	aktuelle Werte	✔	–	✔	✔	✔	–
	zeitkontinuierlich	✔	–	✔[35]	✔[35]	–	–
	weitere Darstellungsformen	–	–	–	–	–	–
3	Läßt sich die Darstellung der Statistiken konfigurieren?	✔	–	–	–	–	–
4	Können für die Verbindungen getrennte Statistiken erstellt werden?	–	–	✔	✔	✔[36]	–
5	Lassen sich die Meßwerte für die Statistiken kontinuierlich speichern?	–	–	–	–	✔	–
6	Existiert ein spezielles Modul für die Auswertung der Statistiken?	–	–	–	–	–	–

a Disk I/O

C.3.3 Alarmfunktionen

		CL	CS	DG3.0	DG3.1	HCS	Pro7
Anf. 4.9	Bereitstellung von konfigurierbaren Alarmfunktionen.						
1	Welche Situationen können mit Alarmen versehen werden?						
	Verbindungsstatus	✔	–	✔	✔	✔	✔
	Nachrichtendurchsatz	✔	–	–	–	–	–
	HDD-Speicherplatz	✔	–	–	–	–	–
	CPU-Auslastung	✔	–	–	–	–	–
	Queue-Auslastungen	✔	–	–	–	–	–
	weitere Situationen	a	–	–	–	b	c

...

[35] Nur für den Nachrichtendurchsatz einzelner Subsystem-Agenten.

[36] Statistiken werden jeweils in den Verarbeitungsmodulen erstellt und gelten somit für alle über diese Module geführten Verbindungen.

...

		CL	CS	DG3.0	DG3.1	HCS	Pro7
2	Wie macht sich ein Alarm im System-Monitor bemerkbar?						
	graphisch	✔	–	✔	✔	–	✔
	textuell	✔	–	✔	✔	✔	✔
	akustisch	–	–	✔	✔	–	–
	weitere Möglickeiten	–	–	–	–	–	–
3	Kann der Administrator beim Auftreten eines Alarms vom Kommunikationsserver z.B. über eMail, Beeper etc. benachrichtigt werden?	✔	–	–	–	✔	–
4	Lassen sich die Alarmfunktionen für die Kommunikationsverbindungen getrennt konfigurieren?	✔	–	–	–	–	✔
5	Werden auch extern implementierte Alarmierungsfunktionen unterstützt?	✔	–	✔	✔	✔	✔

a Viele weitere, sehr differenzierte Möglichkeiten.

b Fehlerhafter Aufbau einer Nachricht.

c Fehler bei der Transformation einer Nachricht.

C.4 Weitere Anforderungen an die Konfigurationskomponente

C.4.1 Definition der Nachrichtenstrukturen

		CL	CS	DG3.0	DG3.1	HCS	Pro7
Anf. 5.1	Definition der Nachrichtenstrukturen proprietärer Kommunikationsprotokolle im Kommunikationsserver.						
1	Lassen sich die Nachrichtenstrukturen aus proprietären Kommunikationsprotokollen im Kommunikationsserver definieren?	✔	–	–	✔	✔	✔
Wenn ja							
2	Exisitert ein spezieller Editor für die Definition der Kommunikationsprotokolle?	✔			✔	✔	–
3	In welcher Form werden die Definitionen gespeichert?						
	ASCII-Datei	✔			✔	–	–
	Datenbank	–			–	✔	–

...

…	CL	CS	DG3.0	DG3.1	HCS	Pro7
andere Speicherform	–			–	–	a
4 Werden Datenfelder mit variabler Feldlänge unterstützt?	✔			✔	✔	✔

a Im Quellcode der Prozesse des Verarbeitungsmoduls, die die Transformationen vornehmen (Konverter).

	CL	CS	DG3.0	DG3.1	HCS	Pro7
Anf. 5.2 Hinterlegung der Nachrichtenstrukturen standardisierter Kommunikationsprotokolle im Kommunikationsserver.						
1 Lassen sich die Nachrichtenstrukturen aus standardisierten Kommunikationsprotokollen im Kommunikationsserver definieren?	✔	✔	–	✔	✔	✔
Wenn ja						
2 Welche standardisierten Kommunikationsprotokolle sind hinterlegt?						
HL7	✔	✔	–	?	✔	✔
EDIFACT	✔	–	–	?	–	–
weitere	X12	–	–	?	–	–
3 In welcher Form werden die Nachrichtenstrukturen hinterlegt?						
ASCII-Datei	✔	–	–	✔	–	–
Datenbank	–	✔	–	–	✔	–
andere Speicherform	–	–	–	–	–	a
4 Lassen sich die standardisierten Kommunikationsprotokolle durch den Anwender anpassen?	✔	✔	–	✔	✔	✔
5 Existiert ein spezieller Editor für die Erweiterung und Änderung der standardisierten Kommunikationsprotokolle?	✔	✔	–	?	✔	–

a Im Quellcode der Prozesse des Verarbeitungsmoduls, die die Transformationen vornehmen (Konverter).

C.4.2 Änderung der Konfigurationen

		CL	CS	DG3.0	DG3.1	HCS	Pro7
Anf. 5.3	Änderungen in der Konfiguration der Kommunikationsverbindungen müssen im laufenden Betrieb möglich sein.						
1	Ist eine Änderung der Konfigurationen im laufenden Betrieb möglich?	✔	✔	✔	✔	✔	✔
2	Wie werden die geänderten Konfigurationen wirksam gemacht?	a	b	a	a	a	b

a *Reload* vom System-Monitor aus.

b Änderungen werden sofort wirksam.

C.4.3 Dokumentation der Kommunikationsverbindungen

		CL	CS	DG3.0	DG3.1	HCS	Pro7
Anf. 5.4	Automatisierte, konfigurierbare Dokumentation der Kommunikationsverbindungen durch den Kommunikationsserver.						
1	Welche Konfigurationen der Kommunikationsverbindungen lassen sich zum Beipiel durch einen Ausdruck extern dokumentieren?						
	Transportprotokolle	✔[37]	?	–[38]	–[38]	?	–[38]
	Kommunikationsprotokolle	✔	?	–	✔	–	–
	Identifizierungsregeln	✔	–[39]	✔	✔	–[39]	–[39]
	Routing-Tabellen	✔	?	✔	✔	✔	–
	Transformationsregeln	✔	–[40]	✔	✔	✔	–[41]
2	Bietet der Kommunikationsserver eine automatische Generierung einer Dokumentation an?	–	–	–	–	–	–

[37] Es lassen sich allerdings nur die Konfigurationen zu den standardmäßig angebotenen Transportprotokollen ausdrucken. Werden in TCL implementierte Subsystem-Agenten eingesetzt, kann nur der Quellcode ausgedruckt werden.
[38] Nur über einen Ausdruck des Quellcodes der Subsystem-Agenten.
[39] Die Nachrichten liegen in einem internen Kommunikationsprotokoll vor, bei dem die Identifizierung der Nachrichten automatisch durch den Kommunikationsserver durchgeführt wird.
[40] Es werden ausschließlich Nachrichten im HL7-Format unterstützt.
[41] Es kann lediglich der Quellcode der Konverter ausgedruckt werden, die für die Transformation der Nachrichten in das interne Kommunikationsprotokoll zuständig sind.

C.5 Anforderungen an die Testkomponente

	CL	CS	DG3.0	DG3.1	HCS	Pro7
Anf. 6.1 Bereitstellung eines Moduls zum Testen der Konfigurationen des Kommunikationsservers.						
1 Für welche Module bzw. Schritte im Verarbeitungsprozeß der Nachrichten werden Testfunktionen angeboten?						
Transportprotokolle	✔	–	–	–	–[42]	–
Definitionen der Kommunikationsprotokolle	✔	–	–	?	–	–
Nachrichtenidentifikation	✔	–	✔	✔	–	–
Nachrichten-Routing	✔	–	✔	✔	–	–
Nachrichtentransformationen	✔	–	✔	✔	–	–
weitere Testfunktionen	–	–	–	–	–	a
2 Besitzt das Testmodul eine graphische Oberfläche?	✔	–	✔	✔	–	–

a Einzelne Kommunikationsverbindungen können in einem Testmodus betrieben werden. In diesem Modus werden keine Nachrichten von den Subsystem-Agenten an die Subsysteme ausgeliefert.

	CL	CS	DG3.0	DG3.1	HCS	Pro7
Anf. 6.2 Unterstützung verschiedener Betriebsmodi für einzelne Kommunikationsverbindungen.						
1 Werden verschiedene Betriebsmodi angeboten?	✔	✔	–	–	–	✔
Wenn ja						
2 Wie unterscheiden sich die Betriebsmodi voneinander?	a	b	–	–	–	c
3 Lassen sie sich für einzelne Verbindungen getrennt einstellen?	–	✔	–	–	–	✔

a Es können drei verschiedene, voneinander unabhängige Kommunikationsserver aufgebaut werden. Die Funktionalität ist bei allen gleich.

b Es werden die im HL7-Standard angebotenen Nachrichtenverarbeitungsmodi (Test, Routine, Schulung) unterstützt.

c Einzelne Kommunikationsverbindungen können in einem Testmodus betrieben werden. In diesem Modus werden keine Nachrichten von den Subsystem-Agenten an die Subsysteme ausgeliefert.

[42] Tests können über einen dafür reservierten Verarbeitungsprozeß (HCS-Local) mit zwei Subsystem-Agenten (HCS-Communication) durchgeführt werden. Der laufende Betrieb wird dadurch nicht gestört.

C.6 Anforderungen an den Datenschutz

C.6.1 Nachrichtenverschlüsselung

Anf. 7.1 Unterstützung von Verschlüsselungsmethoden für die Nachrichtenübertragung.

		CL	CS	DG3.0	DG3.1	HCS	Pro7
1	Lassen sich für die Übertragung der Nachrichten zu den Subsystemen Verschlüsselungsmethoden einsetzen?	✔[43]	–	✔[44]	✔[44]	–	✔[44]
	Wenn ja, lassen sich für die Subsysteme unterschiedliche Verschlüsselungsmethoden einsetzen?	✔		✔	✔		✔
	Wenn nein, wie werden die Nachrichten dann vor einem unbefugten Zugriff geschützt?		a			a	
2	Lassen sich für die Übertragung der Nachrichten zwischen den Modulen des Kommunikationsservers Verschlüsselungsmethoden einsetzen?	–	–	–	–	–[45]	–[46]

a Durch die Sicherungsmaßnahmen, die das Betriebsystem bzw. Netzwerksystem bereitstellt.

Anf. 7.2 Möglichkeit zur Verschlüsselung der gespeicherten Nachrichten.

		CL	CS	DG3.0	DG3.1	HCS	Pro7
1	Lassen sich die Nachrichten in einer verschlüsselten Form speichern?	–	–[47]	–	–	–	–[47]

Anf. 7.3 Möglichkeit zur Verschlüsselung der Log-Dateien.

		CL	CS	DG3.0	DG3.1	HCS	Pro7
1	Lassen sich die Log-Dateien in einer verschlüsselten Form erstellen?	–	–[48]	–	–	–	–[48]

[43] Wird das Transportprotokoll über ein TCL-Skript implementiert, so können dort beliebige Verschlüsselungsmethoden eingesetzt werden.
[44] In den Subsystem-Agenten können beliebige Verschlüsselungsmethoden implementiert werden.
[45] Hier kann die Übertragung der Nachrichten über RPC's erfolgen.
[46] Die Übertragung der Nachrichten zwischen den Modulen erfolgt ausschließlich über RPC's.
[47] Die Nachrichten werden nur in der Datenbank gespeichert und werden dort durch das DBMS geschützt.
[48] Log-Informationen werden nur in der Datenbank abgelegt und werden dort durch das DBMS geschützt.

C.6.2 Zugriffsschutz

	CL	CS	DG3.0	DG3.1	HCS	Pro7
Anf. 7.4 Vom Betriebssystem unabhängiger Zugriffsschutz zum System-Monitor und den weiteren Modulen des Kommunikationsservers.						
1 Für welche Module des Kommunikationsservers besteht ein gesonderter Zugriffsschutz?						
System-Monitor	–	?	✔	✔	✔	–
Editoren der Konfigurationstabellen	–	?	–	–	–	–
Steuerung der Verbindungen (*shutdown* etc.)	–	?	–	–	–	–
weitere Module	–	?	–	–	–	–

	CL	CS	DG3.0	DG3.1	HCS	Pro7
Anf. 7.5 Unterstützung von unterschiedlichen Benutzerrollen zur Steuerung der Rechte der Benutzer.						
1 Lassen sich unterschiedliche Benutzerrollen vergeben?	✔	✔	✔	✔	–	✔
2 Wenn ja, lassen sich vom Anwender eigene Benutzerrollen definieren?	–	–	–	–		–

C.7 Verwaltung einer Datenbank durch den Kommunikationsserver

	CL	CS	DG3.0	DG3.1	HCS	Pro7
Anf. 8.1 Anschluß eines DBMS an den Kommunikationsserver.						
1 Wie läßt sich ein DBMS an den Kommunikationsserver anschließen?						
als eigenständiges Subsystem	✔	✔	✔	✔	✔	✔
über eine spezielle Schnittstelle	–	–	–	–	–	–
2 Welche DBMS werden besonders unterstützt?	–	–	–	–	–	–

	CL	CS	DG3.0	DG3.1	HCS	Pro7
Anf. 8.2 Generierung von Datenbankoperationen aus Nachrichten von Subsystemen.						
1 Lassen sich im Kommunikationsservers aus den Nachrichten Datenbankoperationen generieren?	✔	–	✔	✔	–	✔
2 Wenn ja, wie werden die Operationen generiert und an das DBMS übergeben?	a		a	a		a

a Die Datenbankbefehle lassen sich in Form eines Textes (z.B. SQL-String) durch die Transformationsregeln erzeugen. Dieser Befehl wird vom Subsystem-Agenten dem DBMS übergeben.

	CL	CS	DG3.0	DG3.1	HCS	Pro7
Anf. 8.3 Bereitstellung eines Moduls zur Verwaltung der externen Datenbank.						
1 Wird ein Modul zur Verwaltung der Datenbank vom Kommunikationsserver angeboten?	–	–	–	–	–	–

Anhang D

Dokumentation der Kommunikations- und Transportprotokolle

D.1 Kommunikationsprotokoll für die Abfrage der Patientenstammdaten im PDV-System

D.1.1 Nachrichtenstruktur `PatDatQuery`

Feld-Nr.	Feldname	Typ	Länge	Satzstelle	
				von	bis
1	Message-Kennung PATDATQUERY = Anfrage an IDIK nach Patientendaten	A (lb[1])	12	1	12
2	Nachrichtentyp 600 = Sende Pat.Daten zu folgender Aufnahmenummer	N	3	13	15
3	ANFORD-VON MACDOC = Nuklearmedizin TRANSMED = TrafuMed IDIK = PDV RADOS = Radiologie	A (lb)	8	16	23
4	ZURÜCK-AN MACDOC = Nuklearmedizin TRANSMED = TrafuMed	A (lb)	8	24	31

...

[1] linksbündig

...

Feld-Nr.	Feldname	Typ	Länge	Satzstelle von	Satzstelle bis
	IDIK = PDV RADOS = Radiologie usw.				
5	AUFN-NUMMER	N	8	32	39
6	PATID	A	15	40	54
7	MESSAGE-Datum Datum und Uhrzeit der Erstellung der aktuellen Message (Format: JJJJMMTTSSMMss)	A	14	55	68
8	MESSAGE-Nr Fortlaufende Nummer, die von jedem Daten anforderden System intern verwaltet wird und für jeden Monat neu initialisiert wird.	N	8	69	76

D.1.2 Nachrichtenstruktur `PatDatReply`

Die Nachrichtenstruktur für die Nachrichten der Form `PatDatReply` ist sehr umfangreich (Gesamtlänge einer Nachricht beträgt 2570 Bytes). Hier werden nur die ersten 21 Datenfelder widergegeben, um einen Eindruck von dem Aufbau der Nachricht zu geben.

Feld-Nr.	Feldname	Typ	Länge	Satzstelle von	Satzstelle bis
1	MESSAGE-KENNUNG PATDATREPLY = Antwort von IDIK auf eine Anfrage PATDATA01 = Aufnahme PATDATA02 = Verlegung PATDATA03 = Entlassung	A (lb[2])	12	1	12
2	NACHRICHTENTYP 701 = ok 702 = Patient nicht im Bestand, Datensatz endet nach Feld 8. 703 = Patient wurde gelöscht (kann nur Krankenhausverwaltung!), Datensatz endet nach Feld 8.	N	3	13	15

...

[2] linksbündig

...

Feld-Nr.	Feldname	Typ	Länge	Satzstelle von	Satzstelle bis
	704 = PatdatQuery wahr fehlerhaft, Datensatz endet nach Feld 8.				
3	ANFORD-VON MACDOC = Nuklearmedizin TRANSMED = TrafuMed IDIK = PDV RADOS = Radiologie usw. Muß aus PATDATQUERY übernommen werden.	A (lb)	8	16	23
4	ZURÜCK-AN MACDOC = Nuklearmedizin TRANSMED = TrafuMed IDIK = PDV RADOS = Radiologie usw. Muß aus PATDATQUERY übernommen werden.	A (lb)	8	24	31
5	AUFN-NUMMER	N	8	32	39
6	PATID (I-ZAHL-ALT)	A	15	40	54
7	MESSAGE-Datum Datum und Uhrzeit der Erstellung der aktuellen Nachricht (Format: JJJJMMTTSSMMss)	A	14	55	68
8	MESSAGE-Nr Fortlaufende Nummer, die von jedem Datenanforderden System intern verwaltet wird und für jeden Monat neu initialisiert wird. Muß aus PATDATQUERY übernommen werden.	N	8	69	76
9	BEHANDLUNGS-ART A = ambulant S = stationär H = halbstationär	A	1	77	77
10	BEHANDLUNGS-STATUS	A	1	78	78

...

...

Feld-Nr.	Feldname	Typ	Länge	Satzstelle	
				von	bis
	wenn Feld 9 = A dann: SPACE wenn Feld 9 = H dann: SPACE wenn Feld 9 = S dann: S = vollstationär oder V = vorstationär oder N = nachstationär (derzeit nicht in IDIK!!!)				
11	NAME	A	30	79	108
12	VORNAME	A	30	109	138
13	GEB-NAME	A	15	139	153
14	GEB-DATUM Format: JJJJMMTT	A	8	154	161
15	GESCHLECHT m = männlich w = weiblich u = unbekannt	A	1	162	162
16	FAM-STAND oa = ohne Angabe ld = ledig vh = verheiratet ge = geschieden gt = getrennt lebend vw = verwitwet	A	2	163	164
17	KONFESSION rk = römisch-kath. ev = evangelisch o.a. (freies Textfeld!)	A	2	165	166
18	STAAT	A	3	167	169
19	STRASSE	A	29	170	198
20	PLZ	A	6	199	204
21	ORT	A	25	205	229

D.2 HL7 Lower-Level-Protocol

Das *HL7 Lower-Level-Protocol* definiert eine Nachrichtendarstellungsform für die Übertragung einer Nachricht über zum Beispiel TCP-Sockets. Zusätzlich werden in diesem Protokoll Acknowledgement-Regeln und zwei Acknowledgement-Nachrichten (positives Acknowledegment ACK und negatives Acknowledgement NAK) festgelegt, die eine Synchronisierung des Nachrichtenaustausches auf der Ebene des Transportprotokolls ermöglichen.

Für die Übertragung der Nachrichten legt das *HL7 Lower-Level-Protocol* einen *Envelope* fest, der aus drei verschiedenen Steuerzeichen besteht. Diese Steuerzeichen markieren den Beginn und das Ende einer Nachricht. Gemäß diesem Envelope werden die Nachrichten folgendermaßen dargestellt:

`[STB][---DATA---][ETX][EOB]`	Datenübertragung
`[STB][ACK][ETX][EOB]`	pos. Acknowledgment
`[STB][NAK][ETX][EOB]`	neg. Acknowledgment

Die Belegung der Steuerzeichen ist variabel, standardmäßig werden die folgenden Werte verwendet:

Steuerzeichen	**Bedeutung**	**Wert**
STB	Startblockchar	0x1
ETX	Enddatachar	0x2
EOB	Endblockchar	0xd
ACK	pos. Acknowledgment	0x6
NAK	neg. Acknowledgment	0x15

Anhang E

Beschreibung der entwickelten Kommunikationsklienten

In diesem Kapitel werden die Ergebnisse der Entwicklung der Kommunikationsklienten *IdikGate* und *IdikClient* vorgestellt. Für beide Kommunikationsklienten wurde nach der Anforderungsanalyse eine Beutzeranleitung aufgestellt. Sie werden in den Abschnitten E.1 ab Seite 144 und E.2 ab Seite 154 dargestellt.

E.1 Benutzungsanleitung zu *IdikGate*

In diesem Abschnitt werden die wesentlichen Teile der Benutzeranleitung zu *IdikGate* vorgestellt.

E.1.1 Funktionalität von *IdikGate*

In diesem Abschnitt soll die Funktionalität von *IdikGate* beschrieben werden. Details können aus dem Zustandsübergangsdiagramm von *IdikGate* in der Abbildung E.1 entnommen werden.

Für die Kommunikation mit dem Kommunikationsserver und IDIK verwaltet *IdikGate* insgesamt drei verschiedene Schnittstellen:

1. einen **Inport** zum Empfang der Nachrichten „PATDATQUERY“ vom *Outbound Communication Client* (CC-Out) von DATAGATE^{TM},
2. einen **Outport** zum Senden der Nachrichten „PATDATREPLY“ zum *Inbound Communication Client* (CC-In) von DATAGATE^{TM}, sowie
3. einen **Telnet-Port** zur Abfrage der Stammdaten in IDIK.

Inport: Der Inport besteht aus einer TCP-Socket, die als *Server* konfiguriert ist. Nach der Initalisierung erwartet sie eine *Connect*-Anfrage vom entsprechenden *Communication Client* von DATAGATE^{TM}. Der Nachrichtenaustausch über diese Socket erfolgt gemäß dem *HL7 Lower-Level-Protocol* (siehe Anhang D.2).

Outport: Der Outport besteht ebenfalls aus einer Socket und ist entsprechend als *Client* konfiguriert. Nach der Initialisierung und nach jedem Verbindungsabbruch versucht *IdikGate* diese Socket mit dem entsprechenden *Communication Client* von DATAGATE^{TM} zu verbinden (*connect*). Der Nachrichtenaustausch erfolgt hier ebenfalls gemäß dem *HL7 Lower Level Protocol* (siehe Anhang D.2).

Telnet-Port: Der Telnet-Port zu IDIK besteht aus einer automatisierten Telnet-Sitzung zum Rechner von IDIK. Diese Sitzung wird komplett von *IdikGate* gesteuert. Nach der Initialisierung der Sitzung (Telnet-Aufruf und Login) ruft *IdikGate* auf dem IDIK-Rechner ein Abfrage-Skript auf. Dieses erhält einen Datensatz „PATDATQUERY“ als Eingabe und liefert als Antwort den Datensatz „PATDATREPLY“, gefüllt mit dem Stammdatensatz des Patienten. Diese Ausgabe des Abfrage-Skriptes wird von *IdikGate* eingelesen und zur Nachricht „PATDATREPLY“ umgewandelt.

Die drei Ports werden über eine Konfigurationsdatei mit den notwendigen Informationen (Port-Nummer, Host etc.) versorgt (siehe Abschnitt E.1.2).

Initialisierung

Nach dem Aufruf von *IdikGate* wird zuerst die Konfigurationsdatei eingelesen und die entsprechenden Konfigurationen vorgenommen. Konnten dabei nicht alle zwingenden Konfigurationen (Konfigurationen ohne Default-Wert) aus der Konfigurationsdatei bestimmt werden, bricht *IdikGate* mit einer entsprechenden Fehlermeldung ab. Die optionalen Parameter werden, wenn nicht anderes in der Konfigurationsdatei angegeben, mit ihrem Default-Wert initialisiert. Die Parameter, die beim Aufruf von *IdikGate* mit angegeben werden (siehe Abschnitt E.1.3), haben vor den entsprechenden Parametersetzungen in der Konfigurationsdatei Vorrang. Das Einlesen der Konfigurationsdatei ist ein einmaliger Vorgang, er kann während des Betriebes der Schnittstelle nicht wiederholt werden.

Nach der Initialisierung der Parameter erfolgt die Initalisierung und der Aufbau der drei Verbindungen zu DataGate^{TM} (In- und Outport) und IDIK (Telnet-Port). Dabei wird die folgende Reihenfolge eingehalten:

1. Initialisierung des Inports als Server,
2. Initialisierung des Outports als Client,
3. Verbindungsaufbau über den Outport zum CC-In von DataGate^{TM} (*connect*),
4. Initialisierung der Telnet-Sitzung zu IDIK und Aufruf des Abfrage-Skriptes und schließlich
5. Warten auf eine Verbindungsanfrage (*listen*) vom CC-Out von DataGate^{TM} am Inport.

Erst wenn alle Verbindungen hergestellt sind, nimmt *IdikGate* die Nachrichten über den Inport von DataGate^{TM} entgegen. Eine Ausnahme gilt für den Telnet-Port: Läßt sich bei der Initialisierung keine Verbindung zum IDIK-Rechner herstellen, so werden dennoch Nachrichten am Inport entgegengenommen. Diese werden allerdings an den Sender zurückgewiesen (siehe Abschnitt E.1.1).

Abfrage der Stammdaten eines Patienten

Ist eine Nachricht am Inport vorhanden und befindet sich *IdikGate* im Warte-Zustand (vgl. Abbildung E.1), so wird die Nachricht eingelesen und aus ihr der Datensatz „PATDATQUERY“ extrahiert. Hierbei erfolgt keine Überprüfung der Nachrichtenstruktur und der einzelnen Datenfelder der Nachricht. Er wird unverändert über den Telnet-Port an das Abfrage-Skript in IDIK übergeben. Die Ausgabe des Skriptes erfolgt in Form eines Datensatzes „PATDATREPLY“. Dieser wird von *IdikGate* aus dem Telnet-Port ausgelesen und in Form einer Nachricht „PATDATREPLY“ über den Outport an DataGate^{TM} gesandt. Erst wenn diese Übertragung erfolgreich abgeschlossen wurde, also das positive Acknowledgment (*ACK*) vom Kommunikationsserver empfangen wurde, wird der Empfang der Nachricht „PATDATQUERY“ über den Inport an den Kommunikationsserver bestätigt (versenden von ACK über den Inport). Damit wird die Notwendigkeit einer Pufferung von Nachrichten in *IdikGate* umgangen. Konnte nach dem Empfang einer Anfrage die Antwort nicht versandt werden (z.B. wegen eines Systemabsturzes) befindet sich die Anfrage noch im Puffer des Kommunikationsservers, da

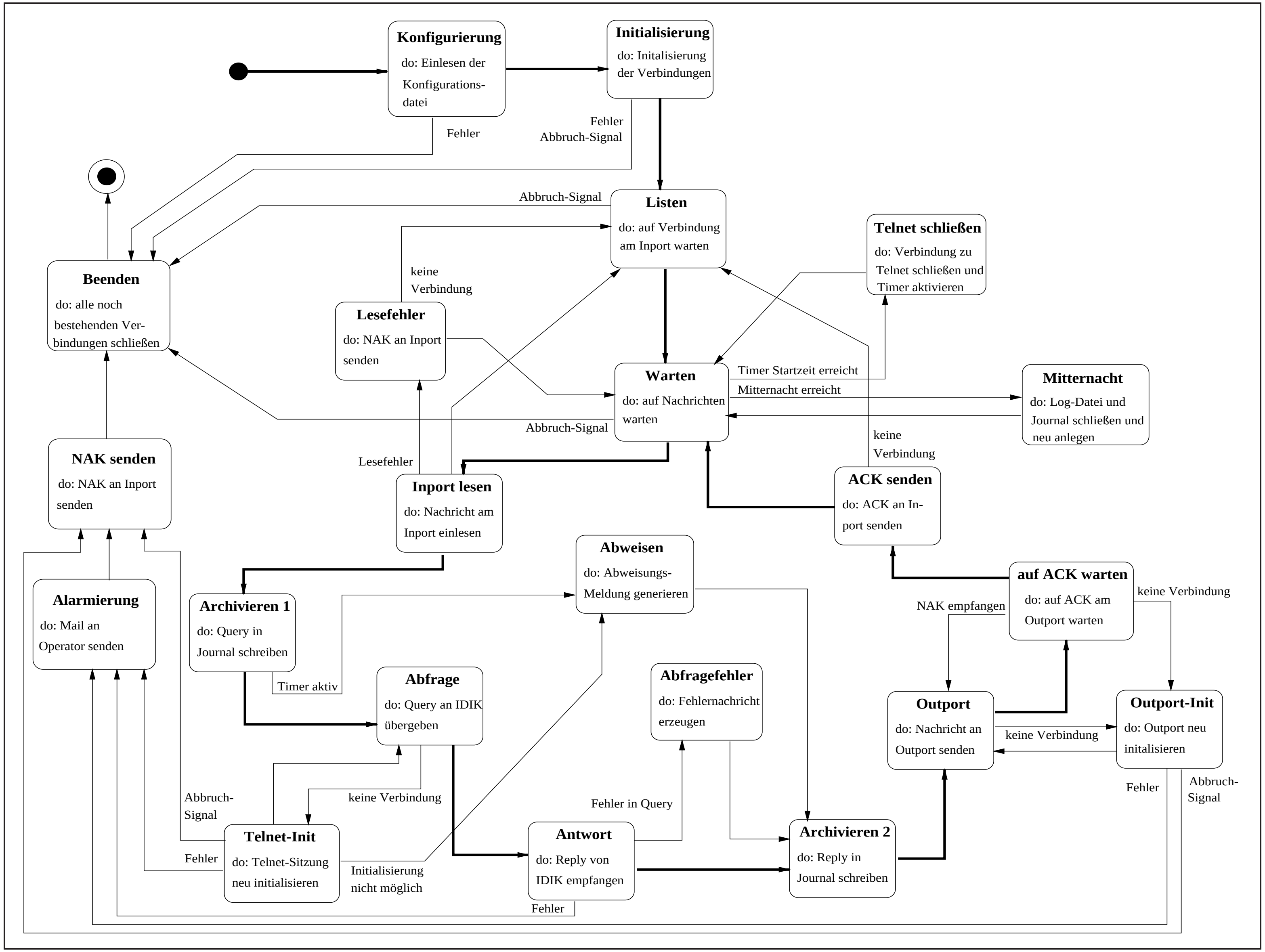


Abbildung E.1: Dynamisches Modell zu *IdikGate* dieser noch keine positive Bestätigung erhalten hat. Die Anfrage geht also nicht verloren und wird vom Kommunikationsserver erneut an *IdikGate* gesandt. Dieses Verfahren der verzögerten Bestätigung setzt natürlich eine entsprechende Verwaltung der Outbound-Queues durch den Kommunikationsserver voraus.

Die oben beschriebene Steuerung des Nachrichtenaustausches über Acknowledgments wird nur dann verwendet, wenn für die Nachrichtenkodierung das *HL7 Lower Level Protocol* verwendet wird (siehe Anhang D.2). Die Nachrichtenkodierung wird in der Konfigurationsdatei ausgewählt und konfiguriert (siehe Abschnitt E.1.2).

Zurückweisen von Nachrichten

Anfragen (PATDATQUERY) werden von *IdikGate* in zwei Fällen zurückgewiesen:

1. Die Anfrage ist fehlerhaft (z.B. eine fehlerhafte Aufnahmenummer).
2. Die Telnet-Verbindung zu IDIK läßt sich zur Zeit nicht aufbauen (z.B. innerhalb des Intervalls, daß durch den Timer bestimmt wird (siehe Abschnitt E.1.1)).

Im ersten Fall wird der Datensatz PATDATREPLY mit dem Nachrichtentyp „704“ und im zweiten Fall mit dem Nachrichtentyp „705“ an den Sender der Anfrage zurückgesandt.

Fehlerbehandlung

Sämtliche Fehler, die beim Empfangen oder Versenden von Nachrichten an den verschiedenen Ports auftreten können, werden von *IdikGate* behandelt. Zu Einzelheiten sei auf das Zustandsübergangsdiagramm (Abbildung E.1) verwiesen. Fehler, die nicht von *IdikGate* behoben werden können, wie zum Beispiel Fehler bei der Initialisierung der Telnet-Sitzung aufgrund eines falschen Paßwortes, führen zu einem Abbruch der Schnittstelle. Dabei werden alle Verbindungen, wenn möglich, vorher korrekt geschlossen. Zusätzlich kann in diesen Fällen an den Administrator eine *eMail* mit einer Fehlermeldung versandt werden. Diese Alarmierungsfunktion kann über die Konfigurationsdatei aktiviert werden (siehe Abschnitt E.1.2).

Logging und Journaling

IdikGate bieten die Möglichkeit, sämtliche Vorgänge in der Schnittstelle in Form von Log-Dateien zu protokollieren. Der Umfang des Logging läßt sich über die Konfigurationsdatei einstellen (siehe Abschnitt E.1.2).

In der Log-Datei werden drei Eintragstypen unterschieden:

Info: Einträge dieser Form beinhalten Informationen über den normalen Verarbeitungsprozeß einer Anfrage. Protokolliert wird zum Beispiel, wann eine Nachricht empfangen oder gesendet wurde. Der Umfang dieser Meldungen kann über die Konfigurationsdatei eingestellt werden.

Warnung: Eine Warnung wird immer dann ausgegeben, wenn eine Unregelmäßigkeit in der Nachrichtenverarbeitung aufgetreten ist. Dies kann zum Beispiel der Empfang einer negativen Quittung (NAK) vom Kommunikationsserver oder der Ausfall einer Verbindung über einen der drei Ports sein.

Fehler: Fehler im Verarbeitungsprozeß einer Nachricht führen zum Eintragstyp „*Fehler*“. Neben einer textuellen Beschreibung des Fehlers und einer Angabe der Lokalisation im Verarbeitungsprozeß, wird hier auch die Fehlernummer angegeben, die von den Betriebssystembefehlen ausgegeben wird. Läßt sich der Fehler nicht von *IdikGate* beheben, so hat dieser einen Abbruch der Schnittstelle zur Folge.

Die Name der Log-Dateien besteht jeweils aus dem aktuellen Tagesdatum in der Form „`TTMMJJ.log`" („`161296.log`" für den 16.12.1996). Jeweils um Mitternacht wird die laufende Log-Datei geschlossen und eine neue mit dem nun aktuellen Tagesdatum eröffnet.

Nachrichteninhalte werden in der Log-Datei nicht protokolliert. Hier erfolgt lediglich eine durchgehende Numerierung der Anfragen (PATDATQUERY) und der Antworten (PATDATREPLY). Die Nachrichteninhalte lassen sich aber in einem *Journal* protokollieren. Dabei wird die laufende Nummer der Nachricht aus der Log-Datei mit ins Journal übernommen. So lassen sich die Nachrichten mit dem protokollierten Verarbeitungsprozeß verbinden.

Standardmäßig werden die Nachrichtenzähler bein einem Neustart von *IdikGate* mit dem Wert „0" initalisiert. Die Zählerstände lassen sich bei dem Abbruch der Schnittstelle aber auch sichern. Bei einem Neustart werden dann die Zähler auf die alten Werte initialisiert (siehe Abschnitt E.1.2).

Der Name der Journal-Dateien entspricht dem der Log-Dateien. Sie tragen allerdings die Endung „`.journal`". Auch das Journal wird von *IdikGate* jeweils tagesweise angelegt. Es kann über die Konfigurationsdatei akiviert werden (siehe Abschnitt E.1.2).

Timer

Für *IdikGate* kann in der Konfigurationsdatei ein Timer aktivert werden (siehe Abschnitt E.1.2). Dieser Timer bestimmt ein Zeitintervall, in dem *IdikGate* die Telnet-Verbindung zu IDIK beendet und alle Anfragen an den Sender zurückweist (siehe Abschnitt E.1.1). Der Beginn dieses Zeitintervalls wird in der Konfigurationsdatei über den Parameter „`BEGINN`" bestimmt. Der Parameter „`ENDE`" gibt das Ende des Zeitintervalls an. Ist dieser Zeitpunkt erreicht, baut *IdikGate* die Telnet-Verbindung zu IDIK wieder auf und nimmt Anfragen vom Kommunikationsserver entgegen.

E.1.2 Konfiguration der Schnittstelle

Im folgenden Abschnitt wird die Konfigurierung von *IdikGate* durch die Einträge in der Konfigurationsdatei beschrieben.

Die Konfigurationsdatei besteht aus *Parametern* und *Werten*, die den Parametern zugewiesen werden. Die allgemeine Syntax lautet:

```
<PARAMETER> = <Wert>
```

Jeder Parameter mit seinem Wert steht dabei in einer eigenen Zeile in der Konfigurationsdatei. Kommentare können mit einem „`#`" in der ersten Spalte einer Zeile eingefügt werden. Fehlen optionale Parameter in der Konfigurationsdatei, werden die Default-Werte für diesen Parameter eingesetzt.

Ein Ausdruck einer vollständigen Konfigurationsdatei befindet sich im Anhang E.1.4.

Konfigurierung der Ports

Über die Ports empfängt und sendet *IdikGate* Nachrichten vom bzw. zum Kommunikationsserver und zu IDIK (siehe Abschnitt E.1.1).

Inport

INPORT: Portnummer, die der Inport für die Socket verwenden soll.

INPORT-CCNAME: Name des *Communication Client*s, der Nachrichten an den Inport senden wird.

INPORT-WARTEN (optional): Zeitintervall, das zwischen zwei Initialisierungsversuchen der Socket verstreichen soll.
(Default-Wert: 10 Sekunden)

Outport

OUTPORT: Portnummer des *Communication Client*s, an den der Outport Nachrichten senden soll.

OUTPORT-HOST: Name des Rechners, auf dem sich der *Communication Client*von DataGate^{TM} befindet.

OUTPORT-CCNAME: Name des *Communication Client*s, an den die Nachrichten gesandt werden.

OUTPORT-WARTEN (optional): Zeitintervall, das zwischen zwei Initialisierungsversuchen der Socket verstreichen soll.
(Default-Wert: 10 Sekunden)

Telnet-Port

TELNET-HOST: Name des Rechners, zu dem die Telnet-Verbindung aufgbaut werden soll.

TELNET-USER: Logon für die Telnet-Sitzung auf dem IDIK-Rechner.

TELNET-PROGRAMM: Name des Abfrage-Skriptes auf dem IDIK-Rechner.

TELNET-EXITCODE: Exit-Code, der das Abfrage-Skript wieder beendet. Dieser wird zum Abbruch des Skriptes anstelle einer Nachricht „PATDATQUERY“ an das Skript übergeben.

TELNET-WARTEN (optional): Wartezeit in Sekunden, die zwischen zwei Initialisierungsversuchen verstreichen soll.
(Default-Wert: 10 Sekunden)

TELNET-TIMEOUT (optional): Diese Zeitangabe wird für die Steuerung der Telnet-Sitzung benötigt. Sie bestimmt ein maximales Zeitintervall, das nach einem Schreibvorgang auf den Telnet-Port verstreichen darf, bis eine Antwort bzw. Reaktion von IDIK gekommen sein muß. Erfolgt nach einem Schreibvorgang auf den Telnet-Port innerhalb dieses Zeitintervalls keine Reaktion oder Antwort von IDIK, nimmt *IdikGate* an, daß die Verbindung unterbrochen wurde und initialisiert die Telnet-Sitzung neu. Die Angabe erfolgt in Sekunden.
(Default-Wert: 20 Sekunden)

Konfigurierung der Log-Dateien

LOGGING (optional): Über diesen Eintrag kann das Logging (siehe Abschnitt E.1.1) ein- oder ausgeschaltet werden. Nur wenn hier ein „`JA`“ eingetragen ist, wird eine Log-Datei angelegt.
(Default-Wert: JA)
Beispiel: `LOGGING = JA`

LOGGING-ZEITSTEMPEL (optional): Jeder Eintrag in der Log-Datei wird mit einem Zeitstempel versehen, dessen Format über diesen Parameter eingestellt werden kann. Die Formatangabe erfolgt mit Hilfe eines Strings, der Steuerzeichen und jedes andere druckbare Zeichen enthalten kann. Steuerzeichen werden dabei durch ein vorangesetztes Prozentzeichen von den übrigen Zeichen getrennt. Eine Übersicht über die möglichen Steuerzeichen gibt die Tabelle E.1.
(Default-Wert: % H:% M:% S)
Beispiel: `LOGGING-ZEITSTEMPEL = %H:%M:%S`

LOGGING-STATUS (optional): Über diesen Parameter kann der Detailierungsgrad der Log-Dateien gesteuert werden. Es werden zwei Detailgrade unterstützt. Im Detailgrad „1“ werden nur die wichtigsten Informationen über den Verarbeitungsprozeß protokolliert, unwichtige werden weggelassen. Zum Beispiel der Austausch der Acknowledgments wird nicht mitprotokolliert.

Steuerzeichen	Bedeutung
%a	Abkürzung des Tagesnamen
%A	voller Tagesname
%b	Abkürzung des Monatsnamen
%B	voller Monatsname
%d	Tag im Monat (01-31)
%H	Stunde (00-23)
%I	Stunde (00-12)
%m	Monat im Jahr (01-12)
%M	Minute (00-59)
%S	Sekunden (00-59)
%y	Jahr im Jahrhundert (00-99)
%Y	Jahr mit Jahrhundertangabe (z.B. 1996)

Tabelle E.1: Tabelle mit den unterstützten Steuerzeichen für die Formatierung des Zeitstempels in der Log-Datei.

Detailgrad „2“ liefert ein vollständiges Protokoll. Warnung und Fehlermeldungen werden bei beiden Detailgraden angezeigt.
(Default-Wert: 2)
Beispiel: `LOGGING-STATUS = 2`

LOGGING-PFAD (optional): Standardmäßig werden die Log-Dateien im selben Verzeichnis abgelegt, aus dem *IdikGate* heraus aufgerufen wurde. Über diesen Parameter läßt sich ein anderes Verzeichnis für die Log-Dateien bestimmen. Dieses Verzeichnis muß vor dem Aufruf von *IdikGate* bereits bestehen. Es kann der absolute oder relative Pfadname des Verzeichnisses angegeben werden. Die relative Angabe bezieht sich dabei immer auf das Verzeichnis, aus dem *IdikGate* heraus aufgerufen wurde.
(Default-Wert: .)
Beispiel: `LOGGING-PFAD = Logging`

LOGGING-MODUS (optional): Standardmäßig wird eine bestehende Log-Datei zwischen zwei Aufrufen von *IdikGate* erweitert (Schreibmodus „`a`“). Bestehende Log-Dateien können aber auch nach jedem Aufruf von *IdikGate* überschrieben werden (Schreibmodus „`w`“). Dazu ist hier ein „`w`“ einzutragen.
(Default-Wert: a)
Beispiel: `LOGGING-MODUS = w`

Konfigurierung des Journals

JOURNAL (optional): Über diesen Eintrag kann das Journal (siehe Abschnitt E.1.1) ein- oder ausgeschaltet werden. Nur wenn hier ein „`JA`“ eingetragen ist, wird ein Journal angelegt.
(Default-Wert: NEIN)
Beispiel: `JOURNAL = JA`

JOURNAL-PFAD (optional): Standardmäßig werden die Journale im selben Verzeichnis abgelegt, aus dem *IdikGate* heraus aufgerufen wurde. Über diesen Parameter läßt sich ein anderes Verzeichnis bestimmen. Dieses Verzeichnis muß vor dem Aufruf von *IdikGate* bereits bestehen. Es kann der absolute oder relative Pfadname des Verzeichnisses angegeben werden. Die relative Angabe bezieht sich dabei immer auf das Verzeichnis, aus dem *IdikGate* heraus aufgerufen wurde.
(Default-Wert: .)
Beispiel: `JOURNAL-PFAD = Journal`

JOURNAL-MODUS (optional): Standardmäßig wird ein bestehendes Journal zwischen zwei Aufrufen von *IdikGate* erweitert (Schreibmodus „`a`"). Bestehende Journale können aber auch nach jedem Aufruf von *IdikGate* überschrieben werden (Schreibmodus „`w`"). Dazu ist hier ein „`w`" einzutragen.
(Default-Wert: a)
Beispiel: `JOURNAL-MODUS = w`

Konfigurierung des Timers

TIMER (optional): Über diesen Parameter kann der Timer (siehe Abschnitt E.1.1) aktiviert oder deaktiviert werden. Wird hier der Wert „`NEIN`" eingetragen, wird kein Timer verwendet, „`JA`" aktiviert den Timer.
(Default-Wert: `NEIN`)
Beispiel: `TIMER = JA`

TIMER-ENDE: Wenn der Timer aktiviert ist (`TIMER = JA`), so wird über diesen Parameter die Stopzeit des Timers eingestellt. Ist der Timer deaktiviert, so kann dieser Parameter entfallen.
Beispiel: `TIMER-STARTZEIT = 16:47`

TIMER-BEGINN: Wenn der Timer aktiviert ist (`TIMER = JA`), so wird über diesen Parameter die Startzeit des Timers eingestellt. Ist der Timer deaktiviert, so kann dieser Parameter entfallen.
Beispiel: `TIMER-STOPZEIT  = 16:46`

Konfigurierung der Alarmierungsfunktion

MAILING (optional): Aktiviert oder deaktivert die Alarmierungsfunktion (siehe Abschnitt E.1.1) von *IdikGate*. Wird hier ein „`NEIN`" eingetragen, werden keine *eMails* im Fehlerfall versandt. Steht hier ein „`JA`" sendet *IdikGate* im Fehlerfall eine *eMail* an die angegebene Adresse.
(Default-Wert: `NEIN`)
Beispiel: `MAILING = JA`

MAIL-USER: Ist die Alarmierungsfunktion aktiviert (`MAILING = JA`), so wird hier die *eMail*-Adresse angegeben, an die im Fehlerfall eine Nachricht versandt werden soll. Dieser Parameter kann entfallen, wenn die Aktivierungsfunktion deaktiviert ist.
Beispiel: `MAIL-USER = lange@komed01.uni-muenster.de`

MAIL-SUBJECT: Ist die Alarmierungsfunktion aktiviert (`MAILING = JA`), so kann hier ein *subject* der *eMail* angegeben werden. Dieser Parameter kann entfallen, wenn die Aktivierungsfunktion deaktiviert ist.
Beispiel: `MAIL-SUBJECT = IdikGate`

Konfigurierung des Nachrichtenzählers

ZAEHLER-SICHERN (optional): Über diesen Parameter kann angegeben werden, ob der Nachrichtenzähler (siehe Abschnitt E.1.1) beim Abbruch von *IdikGate* gesichert werden soll (Wert „`JA`"). Wurde der Nachrichtenzähler gesichert, so werden die Zähler beim erneuten Start von

IdikGate auf die alten Wert initalisiert. Ansonsten werden sie mit dem Wert „`0`“ initalisiert. (Default-Wert: `NEIN`)
Beispiel: `ZAEHLER-SICHERN = JA`

Konfigurierung des Transportprotokolls für In- und Outport

TCP-ENCODING: Dieser Parameter bestimmt den Envelope-Typ für das Versenden der Nachrichten über In- und Outport. Unterstützt werden drei verschiedene Typen:

Wert	Envelope-Typ
`HL7_LOWLEVEL`	Envelope gemäß dem *HL7 Lower Level Protocol* (siehe Anhang D.2 auf Seite 143).
`LENGTH_ENCODED_2BYTE`	Es wird eine Längenkodierung verwendet. Die ersten beiden Bytes einer Nachricht geben die Länge der Nachricht an (exklusiv der Bytes für die Längenangabe).
`LENGTH_ENCODED_4BYTE`	Entspricht `LENGTH_ENCODED_2BYTE`, nur daß für die Längenkodierung vier Bytes verwendet werden.

Beispiel: `TCP-ENCODING = HL7_LOWLEVEL`

ACK: Zeichen, daß für ein positives Acknowledgment verwendet wird. Es muß der ASCII-Wert des Zeichens als Dezimalzahl angegeben werden. Wird nur bei `TCP-ENCODING = HL7_LOWLEVEL` verwendet.
Beispiel: `ACK = 06`

NAK: Zeichen, daß für ein negatives Acknowledgment verwendet wird. Es muß der ASCII-Wert des Zeichens als Dezimalzahl angegeben werden. Wird nur bei `TCP-ENCODING = HL7_LOWLEVEL` verwendet.
Beispiel: `NAK = 21`

STB: Zeichen, daß den Beginn einer Nachricht angibt. Es muß der ASCII-Wert des Zeichens als Dezimalzahl angegeben werden. Wird nur bei `TCP-ENCODING = HL7_LOWLEVEL` verwendet.
Beispiel: `STB = 01`

EOB: Zeichen, daß das Ende einer Nachricht angibt. Es muß der ASCII-Wert des Zeichens als Dezimalzahl angegeben werden. Wird nur bei `TCP-ENCODING = HL7_LOWLEVEL` verwendet.
Beispiel: `EOB = 02`

ETX: Zeichen, daß das Ende der Daten anzeigt. Es muß der ASCII-Wert des Zeichens als Dezimalzahl angegeben werden. Wird nur bei `TCP-ENCODING = HL7_LOWLEVEL` verwendet.
Beispiel: `ETX = 13`

E.1.3 Betrieb von *IdikGate*

Aufruf von *IdikGate*

IdikGate läßt sich in zwei verschiedenen Modi starten:

1. Im **Betriebsmodus** läuft *IdikGate* stumm ab, d.h. es erscheinen keine Meldungen am Bildschirm. In diesem Modus können die Vorgänge in der Schnittstelle nur über die Log-Dateien verfolgt werden. Dieser Modus wird mit dem Befehl „`idikgate`“ gestartet.

2. Im **Debug-Modus** werden alle Informationen, die in die Log-Datei geschrieben werden, auch auf dem Bildschirm ausgegeben. Zusätzlich werden alle Ein- und Ausgaben der Telnet-Sitzung auf dem Bildschirm angezeigt. *IdikGate* wird mit dem Befehl „`idikgate -debug`“ in diesem Modus gestartet.

Abbruch von *IdikGate*

IdikGate läßt sich zu jedem Zeitpunkt beenden. Dabei gehen aufgrund des verzögerten Acknowledgements (siehe Abschnitt E.1.1) keine Anfragen oder Antworten verloren (siehe auch Zustandübergangsdiagramm in Abbildung E.1).

Zum Beenden von *IdikGate* ist `Control C` oder der Unix-Befehl `kill` zu verwenden.

E.1.4 Vollständige Konfigurationsdatei

```
######################################################
#
# Konfigurationsdatei fuer IDIKGATE
#
######################################################
#
# Inport
#
INPORT        = 20110
INPORT-WARTEN = 10
INPORT-CCNAME = CC-Out
#
# Outport
#
OUTPORT        = 20100
OUTPORT-HOST   = komed01
OUTPORT-WARTEN = 10
OUTPORT-CCNAME = CC-In
#
# Telnet
#
TELNET-HOST      = idik
TELNET-USER      = ###########
TELNET-PROGRAMM  = do.pat1257
TELNET-EXITCODE  = 699
TELNET-WARTEN    = 10
TELNET-TIMEOUT   = 20
#
# Log-Datei
#
LOGGING            = JA
LOGGING-ZEITSTEMPEL = %H:%M:%S
LOGGING-STATUS     = 2
LOGGING-PFAD       = .
LOGGING-MODUS      = w
#
# Journal
#
JOURNAL       = JA
JOURNAL-PFAD  = .
JOURNAL-MODUS = w
#
# Timer
#
TIMER        = JA
TIMER-BEGINN = 15:00
TIMER-ENDE   = 15:07
#
# Mailing
#
MAILING      = JA
MAIL-USER    = lange@komed01.uni-muenster.de
MAIL-SUBJECT = IdikGate
#
# Nachrichten Zaehler
#
ZAEHLER-SICHERN = NEIN
#
# Envelope
```

```
#
TCP-ENCODING = HL7_LOWLEVEL
# HL7_LOWLEVEL | LENGTH_ENCODED_(2|4)BYTE
ACK = 06
NAK = 21
STB = 01
EOB = 02
ETX = 13
```

E.2 Benutzungsanleitung zu *IdikClient*

IdikClient ist eine interaktive Schnittstelle zur Abfrage von Patientenstammdaten aus dem Patientendatenverwaltungssystem IDIK an den Medizinischen Einrichtungen der Universität Münster. Die Schnittstelle nimmt die Aufnahmenummer eines Patienten entgegen, baut einen Datensatz nach der Datensatzdefinition *PATDATQUERY* auf, sendet diesen an den Kommunikationsserver (KS) und empfängt anschließend den zur Aufnahmenummer gehörenden Stammdatensatz im Datensatzformat *PATDATREPLY* vom KS. Die Datensatzbeschreibung von *PATDATQUERY* und *PATDATREPLY* befinden sich im Anhang D.

E.2.1 Anleitung zur Benutzung der Schnittstelle

IdikClient wird mit dem folgenden Befehl gestartet:

Aufruf: `idikclient`

Über Parameter lassen sich Eigenschaften der Schnittstelle verändern. Sie werden in der Tabelle E.2 beschrieben. Eine weitergehende Konfiguration wird über eine Konfigurationsdatei vorgenommen (siehe Abschnitt E.2.2).

Syntax: `idikclient [-in <Dateiname> [-status][-exit]][-nodisplay][-debug]`

Parameter	**Bedeutung**
`-in <Dateiname>`	Name der Datei, aus der die Aufnahmenummern gelesen werden sollen.
`-status`	Nur in Verbindung mit dem Parameter `-in`. Es werden Statusmeldungen von der Schnittstelle angezeigt.
`-exit`	Nur in Verbindung mit dem Parameter `-in`. Die Schnittstelle wird nach dem Einlesen der letzten Aufnahmenummer und dem Ablauf eines Zeitintervalls zum Empfang der Stammdatensätze automatisch beendet.
`-nodisplay`	Es werden keine Stammdatensätze am Bildschirm angezeigt. Dieser Parameter wird nur im interaktiven Modus berücksichtigt.
`-debug`	Es werden alle Protokoll-Einträge auch am Bildschirm angezeigt. Ist in beiden Modi erlaubt.

Tabelle E.2: Übersicht über die Parameter für den Aufruf von *IdikClient*.

Nach dem Aufruf von *IdikClient* wird die Verbindung zum KS aufgebaut. *IdikClient* informiert den Benutzer mit der Ausgabe einer Statusmeldung, sobald die Verbindung hergestellt ist. Der Verbindungsaufbau kann unter bestimmten Umständen etwa 30 Sekunden dauern. Ist nach Ablauf dieser Zeit keine Verbindung hergestellt, sollte die Ausführung von *IdikClient* abgebrochen werden (`Ctrl C`). Aus den Einträgen im Protokoll (siehe Abschnitt E.2.3) läßt sich die Ursache für das Scheitern des Verbindungsaufbaus ablesen. Die Hauptursache wird sein, daß der KS nicht erreichbar ist.

```
$idikclient
DISPLAY : [13:51:41] Warte auf Verbindung...
DISPLAY : [13:51:47] Verbindung hergestellt.
Aufnahmenummer oder Dateiname ('exit' fuer Programmende):
>28453765
DISPLAY : [13:51:50] Warte auf Antwort von IDIK...

Stammdaten zu Aufnahmenummer >28453765<
=======================================
Nachrichten-Nummer: >1<
Name..............: >Franz Mustermann<
Geburtsdatum......: >02.02.1943<
Anschrift.........: >Hauptstrasse 4a, 48000 Muenster<
Aufnahmedatum.....: >30.08.1995<        Behandlungsart: >S<
Entlassungsdatum..: >12.09.1995<        Entlassungsart: >1<

Einweisender Arzt
-----------------
Name..............: >                              <
Anschrift.........: >                             <

Pflegesaetze: (Anzahl:  1)
--------------------------
Datum          Privat  Pflegesatz
30.08.1995     N       030

Aufnahmenummer oder Dateiname ('exit' fuer Programmende):
>
```

Abbildung E.2: Beispielsitzung mit Eingabe der Aufnahmenummer über die Tastatur. Die Aufnahmenummer und der Stammdatensatz sind frei erfunden. Das Zeichen **$** stellt den Prompt des Betriebssystems des Rechners dar und ist abhängig von den lokalen Einstellungen.

Nachdem die Verbindung zum KS hergestellt ist, erscheint die Eingabeaufforderung für die Eingabe der Aufnahmenummern.

Eingabe der Aufnahmenummern über die Tastatur

Sobald die Verbindung zum KS hergestellt und die Eingabeaufforderung erschienen ist, können die Aufnahmenummern über die Tastatur eingegeben werden. Dabei erfolgt eine Prüfung, ob die eingegebene Aufnahmenummer gültig ist (achtstellige Dezimalzahl). Ungültige Eingaben werden zurückgewiesen. *IdikClient* generiert nun aus der Aufnahmenummer eine Nachricht der Struktur PATDATQUERY und sendet sie an den KS. Jede Nachricht erhält dabei eine laufende Nummer. Diese Numerierung wird in der Regel für jede Sitzung neu initialisiert. Die Schnittstelle kann aber auch so konfiguriert werden, daß die Numerierung über mehrere Sitzungen fortgeführt wird (siehe Abschnitt E.2.2).

IdikClient wartet nun solange, bis der zugehörige Stammdatensatz in Form der Nachricht PATDATREPLY vom KS empfangen wurde. Dieser wird auszugsweise am Bildschirm dargestellt und zusätzlich als vollständiger Stammdatensatz in der Ausgabedatei (siehe Abschnitt E.2.1) gespeichert. Nachdem der Stammdatensatz angezeigt wurde, erscheint wieder die Eingabeaufforderung und eine weitere Aufnahmenummer kann eingegeben werden.

In der Abbildung E.2 ist eine Sitzung beispielhaft dargestellt.

Einlesen der Aufnahmenummern aus einer Datei

Die Aufnahmenummern können auch aus einer Datei eingelesen werden. Dazu ist anstelle einer Aufnahmenummer der Name der Datei einzugeben, in der sich die Aufnahmenummern befinden. Die Aufnahmenummern in der Eingabedatei müssen jeweils in einer eigenen Zeile stehen. *IdikClient* liest nun die Aufnahmenummern einzeln ein und zeigt jeweils den Stammdatensatz an. Ungültige Aufnahmenummern werden dabei ignoriert. Nachdem die letzte Aufnahmenummer eingelesen und der Stammdatensatz empfangen wurde, erscheint wieder die Eingabeaufforderung.

Eine Beispielsitzung mit Einlesen der Aufnahmenummern aus einer Datei `input.dat` mit zwei Aufnahmenummern ist in der Abbildung E.3 dargestellt.

```
$idikclient
DISPLAY : [14:16:28] Warte auf Verbindung...
DISPLAY : [14:16:42] Verbindung hergestellt.
Aufnahmenummer oder Dateiname ('exit' fuer Programmende):
> input.dat

DISPLAY : [14:16:42] Aufnahmenummer <42563485> eingelesen.
DISPLAY : [14:16:42] Warte auf Antwort von IDIK...

Stammdaten zu Aufnahmenummer >42563485<
=======================================
Nachrichten-Nummer: >1<
Name..............: >Theo Mustermann<
Geburtsdatum......: >02.02.1945<
Anschrift.........: >Sonennstrasse 15, 35466 Sonnenberg<
Aufnahmedatum.....: >30.08.1995<        Behandlungsart: >A<
Entlassungsdatum..: >  .  .    <        Entlassungsart: > <

Einweisender Arzt
-----------------
Name..............: >                              <
Anschrift.........: >                             <
                    >                                <
Pflegesaetze:  (Anzahl:  1)
---------------------------
Datum           Privat  Pflegesatz
07.03.1996      N       010


DISPLAY : [14:16:44] Aufnahmenummer <73852971> eingelesen.
DISPLAY : [14:16:45] Warte auf Antwort von IDIK...

Stammdaten zu Aufnahmenummer >73852971<
=======================================
Nachrichten-Nummer: >2<
Name..............: >Maria Montessori<
Geburtsdatum......: >29.04.1942<
Anschrift.........: >Schulstrasse 3, 44000 Lehrerheim<
Aufnahmedatum.....: >20.04.1995<        Behandlungsart: >A<
Entlassungsdatum..: >  .  .    <        Entlassungsart: > <

Einweisender Arzt
-----------------
Name..............: >.                             <
Anschrift.........: >.                            <
                    >.      .                        <
Pflegesaetze:  (Anzahl:  1)
---------------------------
Datum           Privat  Pflegesatz
18.05.1994      N       002

Aufnahmenummer oder Dateiname ('exit' fuer Programmende):
>
```

Abbildung E.3: Beispielsitzung mit Einlesen der Aufnahmenummer aus einer Datei. Die Aufnahmenummern und die Stammdatensätze sind frei erfunden. Das Zeichen `$` stellt den Prompt des Betriebssystems des Rechners dar und ist abhängig von den lokalen Einstellungen.

Unterdrücken der Anzeige der Stammdatensätze

IdikClient kann auch so gestartet werden, daß die Stammdatensätze nicht am Bildschirm angezeigt werden, sondern nur in die Ausgabedatei geschrieben werden. Nachdem ein Stammdatensatz empfangen wurde, erscheint dann lediglich eine Statusmeldung auf dem Bildschirm, die über den erfolgreichen

Empfang informiert. Um die Anzeige zu unterdrücken, wird *IdikClient* mit dem Parameter `-nodisplay` gestartet:

Aufruf: `idikclient -nodisplay`

Ausgabedatei

Alle Stammdatensätze, die von *IdikClient* empfangen werden, werden in der Ausgabedatei gespeichert. Dateiname, Pfad und Schreibmodus der Datei können über die Konfigurationsdatei von *IdikClient* eingestellt werden (siehe Abschnitt E.2.2). Falls vor Aufruf von *IdikClient* die Ausgabedatei schon vorhanden ist, gibt der Schreibmodus an, ob diese überschrieben (Schreibmodus „`w`") oder erweitert werden soll (Schreibmodus „`a`"). Standardmäßig werden die folgenden Werte verwendet:

Dateiname:	`output.dat`
Pfad:	aktuelles Verzeichnis
Schreibmodus:	`w`

Es kann vorkommen, daß in der Ausgabedatei mehr Stammdatensätze erscheinen als in der aktuellen Sitzung angefordert wurden, obwohl der Schreibmodus `w` gewählt wurde. Es handelt sich dabei um Stammdatensätze, die während der letzten Sitzung nicht vom KS empfangen werden konnten und vom KS aufbewahrt wurden.

Beenden der Schnittstelle

IdikClient kann durch Eingabe des Schlüsselwortes „`exit`" bei der Eingabeaufforderung beendet werden.

Da die interaktive Anfrage eines Stammdatensatzes blockierend ist, kann es notwendig werden, *IdikClient* abzubrechen. Dazu wird der Unix-Shellbefehl `kill <Prozeßnummer>` oder die Tastenkombination `Ctrl C` verwendet. Wird die Schnittstelle auf diese Weise beendet, so werden die Verbindungen korrekt abgebaut und gegebenenfalls der Nachrichtenzähler gesichert.

Konnte ein Stammdatensatz nicht empfangen werden, obwohl die Anfrage (PATDATQUERY) erfolgreich an den KS gesandt wurde, so geht dieser nicht verloren. Er wird bei der nächsten Sitzung von *IdikClient* empfangen und in die Ausgabedatei geschrieben.

E.2.2 Konfiguration der Schnittstelle

Viele Parameter von *IdikClient* können über eine Konfigurationsdatei eingestellt werden. Dabei wird zwischen *optionalen* und *verbindlichen* Parametern unterschieden. Fehlt ein verbindlicher Parameter in der Konfigurationsdatei, so bricht *IdikClient* mit einer Fehlermeldung ab. Fehlt ein optionaler Parameter, so verwendet *IdikClient* Standardwerte.

Alle Parameter werden in der Form `<Schlüsselwort> = <Parameterwert>` angegeben. Kommentare in der Konfigurationsdatei müssen mit einem „`#`" in der ersten Spalte beginnen.

Im folgenden werden alle Parameter, die über die Konfigurationsdatei eingestellt werden müssen oder können, mit einem Beispiel kurz beschrieben. Ein Ausdruck einer Konfigurationsdatei befindet sich im Anhang E.2.4.

CLIENT-NAME (verbindlich) Hier wird der Name der verwendeten Schnittstelle angegeben. Dieser Name muß mit dem Betreiber des Kommunikationsservers abgestimmt werden und darf nur mit dessen Zustimmung geändert werden.

Beispiel: `CLIENT-NAME = HNO`

INPORT (verbindlich) Die verwendete Port-Nummer der TCP/IP-Verbindung zum Kommunikationsserver zum Empfangen der Stammdatensätze. Darf nur mit Abstimmung des Betreibers des Kommunikationsservers geändert werden.

Beispiel: `INPORT = 20100`

INPORT-WARTEN (optional) Gibt an, wieviele Sekunden *IdikClient* warten soll, bis ein erneuter Versuch gestartet wird, die Verbindung zum Kommunikationsserver zum Empfangen der Stammdatensätze aufzubauen. Standardmäßig werden 10 Sekunden verwendet.

Beispiel: `INPORT-WARTEN = 10`

INPORT-CCNAME (verbindlich) Name des *Communication Client* des Kommunikationsservers, von dem *IdikClient* die Stammdatensätze empfängt. Sollte nur nach Absprache mit dem Betreiber des Kommuniaktionsservers geändert werden.

Beispiel: `INPORT-CCNAME = Client-Out`

OUTPORT (verbindlich) Die verwendete Port-Nummer der TCP/IP-Verbindung zum Kommunikationsserver zum Senden der Aufnahmenummern. Darf nur mit Abstimmung des Betreibers des Kommunikationsservers geändert werden.

Beispiel: `OUTPORT = 20100`

OUTPORT-WARTEN (optional) Gibt an, wieviele Sekunden *IdikClient* warten soll, bis ein erneuter Versuch gestartet wird, die Verbindung zum Kommunikationsserver zum Senden der Aufnahmenummern aufzubauen. Standardmäßig werden 10 Sekunden verwendet.

Beispiel: `OUTPORT-WARTEN = 10`

OUTPORT-CCNAME (verbindlich) Name des *Communication Client* des Kommunikationsservers, an den *IdikClient* die Aufnahmenummern sendet. Sollt nur nach Absprache mit dem Betreiber des Kommuniaktionsservers geändert werden.

Beispiel: `OUTPORT-CCNAME = Client-In`

AUSGABEDATEI-NAME (optional) Der Dateiname der Ausgabedatei, in die die Stammdatensätze geschrieben werden. Standardmäßig wird `output.dat` verwendet.

Beispiel: `AUSGABEDATEI-NAME = output.dat`

AUSGABEDATEI-MODUS (optional) Gibt den Schreibmodus an, in dem die Ausgabedatei geöffnet werden soll. Dieser Schreibmodus wird nur zum Öffnen der Datei zu Beginn einer Sitzung verwendet. Während einer Sitzung werden alle Stammdatensätze an die Ausgabedatei angefügt.

Mögliche Parameterwerte sind:

Schreibmodus	Wirkung
w	Datei überschreiben, falls sie schon vorhanden ist. (Standardwert)
a	An die Datei anhängen, falls sie schon vorhanden ist. Sonst neu erzeugen.

Beispiel: `AUSGABEDATEI-MODUS = w`

AUSGABEDATEI-PFAD (optional) Das Verzeichnis, in dem sich die Ausgabedatei befindet. Standardmäßig wird das aktuelle Verzeichnis verwendet.

Beispiel: AUSGABEDATEI-PFAD = .

LOGGING (optional) Hier läßt sich einstellen, ob eine automatische Protokolldatei über die Prozesse in der Schnittstelle geführt werden soll (siehe auch Abschnitt E.2.3). Mögliche Parameterwerte sind „`JA`" (Standardwert) und "texttNEIN".

Beispiel: `LOGGING = JA`

LOGGING-ZEITSTEMPEL (optional) Alle Einträge in der Protokolldatei werden mit einem Zeitstempel versehen, dessen Format hier eingestellt werden kann. Das Format kann aus Steuerzeichen und Text bestehen. Die Tabelle E.1 auf Seite 150 listet alle möglichen Steuerzeichen auf. Standardwert ist `%H:%M:%S`.

Beispiel: `LOGGING-ZEITSTEMPEL = %H:%M:%S`

LOGGING-STATUS (optional) Über diesen Parameter läßt sich der Umfang der Protokolldatei steuern. Mögliche Werte sind „`1`" (nur das wichtigste wird protokolliert) und „`2`" (vollständiges Protokoll). Der Standardwert ist „`2`".

Beispiel: `LOGGING-STATUS = 2`

LOGGING-PFAD (optional) Das Verzeichnis, in dem die Protokolldatei abgelegt werden soll. Der Standardwert ist das aktuelle Verzeichnis.

Beispiel: `LOGGING-PFAD = .`

LOGGING-MODUS (optional) Gibt den Schreibmodus an, in dem die Protokolldatei geöffnet werden soll. Dieser Schreibmodus wird nur zum Öffnen der Datei zu Beginn einer Sitzung verwendet. Während einer Sitzung wird das Protokoll weitergeführt.

Mögliche Parameterwerte sind:

Schreibmodus	Wirkung
w	Protokolldatei überschreiben, falls sie schon vorhanden ist. (Standardwert)
a	An die Protokolldatei anhängen, falls sie schon vorhanden ist. Sonst neu erzeugen.

Beispiel: `LOGGING-MODUS = w`

ZAEHLER SICHERN (optional) Während einer Sitzung werden die Datensätze, die aus der eingegebenen Aufnahmenummer aufgebaut werden, durchgehend numeriert. Diese Nummer wird im Feld `MESSAGE-Nr` des Stammdatensatzes `PATDATREPLY` (vgl. Anhang D.1.2) zurückgegeben. Standardmäßig wird der Zähler bei jedem Aufruf von *IdikClient* neu initialisiert. Soll der Zählerstand am Ende einer Sitzung gesichert und bei einem erneuten Aufruf weitergeführt werden, so ist hier der Parameterwert „`JA`", sonst „`NEIN`" (Standardwert) anzugeben.

Beispiel: `ZAEHLER SICHERN = JA`

WARTEN-NACH-EXIT (optional) Gibt das Zeitintervall in Sekunden an, das im automatisiertem Modus nach dem Beenden der Schnittstelle mit `-exit` auf noch ausstehende Stammdatensätze gewartet werden soll. Standardwert ist 30 Sekunden.

Beispiel: `WARTEN-NACH-EXIT = 30`

TCP-ENCODING (verbindlich) Bestimmt das erweiterte Transportprotokoll, das für die Übertragung der Nachrichten verwendet wird. Mögliche Werte sind:

Wert	Bedeutung
HL7_LOWLEVEL	Es wird das *HL7 Lower-Level-Protocol* für die Übertragung der Nachrichten verwendet.
LENGTH_ENCODED_2BYTE	Den Nachrichten wird eine 2 Byte breite Längenangabe vorangestellt.
LENGTH_ENCODED_4BYTE	Den Nachrichten wird eine 4 Byte breite Längenangabe vorangestellt.

Beispiel: `TCP-ENCODING = HL7_LOWLEVEL`

ACK (verbindlich) Zeichen für ein positives Acknowledgement. Es ist der ASCII-Code des Zeichens als Dezimalzahl anzugeben. Darf nur nach Absprache mit dem Betreiber des Kommunikationsservers geändert werden.

Beispiel: `ACK = 06`

NAK (verbindlich) Zeichen für ein negatives Acknowledgement. Es ist der ASCII-Code des Zeichens als Dezimalzahl anzugeben. Darf nur nach Absprache mit dem Betreiber des Kommunikationsservers geändert werden.

Beispiel: `NAK = 21`

ETX (verbindlich) Zeichen für *Ende-Daten*. Es ist der ASCII-Code des Zeichens als Dezimalzahl anzugeben. Darf nur nach Absprache mit dem Betreiber des Kommunikationsservers geändert werden.

Beispiel: `ETX = 13`

STB (verbindlich) Zeichen für *Start-Block*. Es ist der ASCII-Code des Zeichens als Dezimalzahl anzugeben. Darf nur nach Absprache mit dem Betreiber des Kommunikationsservers geändert werden.

Beispiel: `STB = 01`

EOB (verbindlich) Zeichen für *Ende-Block*. Es ist der ASCII-Code des Zeichens als Dezimalzahl anzugeben. Darf nur nach Absprache mit dem Betreiber des Kommunikationsservers geändert werden.

Beispiel: `EOB = 02`

E.2.3 Protokollieren einer Sitzung

Jede Sitzung kann in einer Protokolldatei automatisch protokolliert werden. Dazu ist in der Konfigurationsdatei der Parameter `LOGGING` auf den Wert „`JA`" zu setzen. Der Name der Protokolldatei wird aus dem aktuellen Datum in der Form `ttmmjj.log` aufgebaut. Das Verzeichnis für die Protokolldateien läßt sich über die Konfigurationsdatei einstellen, standardmäßig wird das aktuelle Verzeichnis verwendet. In der Konfigurationsdatei läßt sich auch der Schreibmodus der Protokolldatei einstellen. Bei einem Schreibmodus „`w`" wird die Protokolldatei bei jeder Sitzung neu angelegt. Eine ältere Protokolldatei des selben Tages wird dabei überschrieben. Im Schreibmodus „`a`" wird die Protokolldatei eines Tages bei jeder neuen Sitzung erweitert.

Weitere Einstellungen zum automatischen Protokollieren lassen sich in der Konfigurationsdatei vornehmen (siehe Abschnitt E.2.2).

Ein Eintrag im Protokoll setzt sich aus einem Schlüsselwort, das die Art des Eintrages beschreibt, einem Zeitstempel und einem Informationsteil zusammen. Es können die folgenden Schlüsselwörter auftreten:

INFO Ein normaler Eintrag, der über den Prozeßverlauf in der Schnittstelle informiert.
Beispiel:
`INFO : [14:16:42] Aufnahmenummer <83273895> eingelesen.`

WARNUNG Es ist eine Unregelmäßigkeit im Ablauf aufgetreten, wie zum Beispiel eine ungültige Aufnahmenummer in einer Eingabedatei.
Beispiel:
`WARNUNG : [14:16:51] Falsche Aufnahmenummer '14' in <input.dat>. Wird ignoriert.`

FEHLER Es ist ein Fehler aufgetreten. Im Informationsteil wird die Art des Fehlers und die Lokalisation im Programmablauf angegeben. Zusätzlich wird die Fehlernummer angezeigt. Handelt es sich bei dem Fehler um einen fatalen Fehler, wird das Programm beendet.
Beispiel:

```
FEHLER  : [16:46:06] Binding-Fehler! (waehrend Init_Inport). Warte 10 Sek.
FEHLER  : [16:46:06] ERRNO = 48
```

Der Zeitstempel gibt die genaue Zeit an, wann das Ereignis, das den Eintrag verursacht hat, aufgetreten ist. Das Format des Zeitstempels läßt sich in der Konfigurationsdatei einstellen.

E.2.4 Ausdruck einer Konfigurationsdatei für *IdikClient*

```
################################################################################
#
# Konfigurationsdatei fuer IDIKCLIENT
#
################################################################################

# CLIENT-NAME (maximal 8 Zeichen lang) (nicht aendern)
CLIENT-NAME = CLIENT
#
# Inport (nicht aendern)
#
INPORT        = 20010
INPORT-WARTEN = 10
INPORT-CCNAME = CC-Out
#
# Outport (nicht aendern)
#
OUTPORT        = 20000
OUTPORT-HOST   = komed01
OUTPORT-WARTEN = 10
OUTPORT-CCNAME = CC-In
#
# Ausgabe-Datei
#
AUSGABEDATEI-NAME  = output.dat
AUSGABEDATEI-MODUS = w
AUSGABEDATEI-PFAD  = .
#
# Log-Datei
#
LOGGING            = JA
LOGGING-ZEITSTEMPEL = %H:%M:%S
LOGGING-STATUS     = 2
LOGGING-PFAD       = .
LOGGING-MODUS      = w
#
# Nachrichtenzaehler
#
ZAEHLER SICHERN = NEIN
#
# Warten nach Exit
#
WARTEN-NACH-EXIT = 30
#
# Envelope (nicht aendern)
#
TCP-ENCODING = HL7_LOWLEVEL
# HL7_LOWLEVEL | LENGTH_ENCODED_(2|4)BYTE
ACK = 06
NAK = 21
STB = 01
```

```
EOB = 02
ETX = 13
```

Literaturverzeichnis

[Dud89] J. Dudeck. Krankenhaus-Informationssysteme. In H. Meyer und A. Bedürftig, Hrsg., *Einsatz der EDV im Gesundheitswesen*, Seiten 75 – 95. Ueberreuter Wissenschaft, 1989.

[EDI95] DIN-Normdatenbank und Regelwerk EDIFACT. In Deutsche EDI-Gesellschaft Berlin, Hrsg., *EDI-Jahrbuch*. Deutsche EDI-Gesellschaft Berlin, 1995.

[GMD] GMDS. Abstrakts zur 41. Jahrestagung der Deutschen Gesellschaft für Medizinische Informatik, Biometrie und Epidemiologie (GMDS) e.V. Bonn.

[Goe93] M. Goedicke. *On the Structure of Software Description Languages: A Component Oriented View*. Habilitationsschrift, Universität Dortmund, Fachbereich Informatik, 1993. Forschungsbericht Nr. 473/1993.

[Grä94] Stefan Gräber. Konzeption und Einsatz eines Kommunikationsservers. In *HL7 Workshop: Implementierung von Kommunikationsstandards*. HL7-Benutzergruppe, Oktober 1994.

[GS97] B. Gesell und T. Schmal. Das pro7-System als Basis für den Aufbau eines verteilten krankenhausweiten Informationssystems. In W. Hasselbring, Hrsg., *Erfolgsfaktor Softwaretechnik für die Entwicklung von Krankenhausinformationssystemen*, Seiten 37 – 44. GMDS/GI-Workshop, Universität Dortmund, Krehl Verlag, Münster, Februar 1997.

[Hac85] W. Hackbusch. Multigrid Methods and Applications. In *Springer Series in Comp. Math.*, Band 4. Springer, 1985.

[Ham93] W. E. Hammond. HL7: A Protokol for the Interchange of Healthcare Data. In G.J.E. De Moore et al., Hrsg., *Progress in Standardization in Health Care Informatics*. IOS Press, 1993.

[Has96] W. Hasselbring. Gestaltung und Implementierung von Krankenhausinformationssystemen als föderierte Datenbanksysteme. In S. Conrad, M. Höding, S. Janssen und G. Saake, Hrsg., *Kurzfassungen des Workshops "Föderierte Datenbanken"*, Seiten 67–73, Magdeburg, April 1996. Bericht 96–01, ITI, University of Magdeburg.

[Hea96] Healthcare Communication Inc. Cloverleaf Toolset Users Guide Version 3.1.0P. Benutzerhandbuch zur Version 3.1, April 1996.

[Hei96] K. U. Heitmann. Kommunikationsserver – Konzepte und Produkte. In P. Haas, C.O. Köhler, K. Kuhn, P. Pietrzyk und H.U. Prokosch, Hrsg., *Praxis der Informationsverarbeitung im Krankenhaus*. ecomed, 1996.

[Jun95] G. Junkermann. Das pro7-Projekt – die Entwicklung eines HL7-Servers zur Kopplung verteilter Anwendungen. In *HL7-Workshop: Datenaustausch im Krankenhaus*. HL7-Benutzergruppe, Oktober 1995.

[Krö96] A. Kröber. Anforderungsanalyse an einen konfigurierbaren HL7-Kommunikations-Server mittels OMT und ausführbarer Modelle. Diplomarbeit, Fachbereich Informatik der Universität Dortmund, 1996.

[Lib94] D. Libes. *Exploring Expect. A Tcl-based Toolkit for Automating Interactive Programs.* O'Reilly and Associates /VVA, 1994.

[Ous94] J. K. Ousterhout. *Tcl and the Tk Toolkit.* Addison-Wesley, 1994.

[ÖV91] M. T. Özsu und P. Valduriez. Distributed database systems: where are we now? *IEEE Computer*, 24(8):68–78, 1991.

[Pro94] H.-U. Prokosch. *Ein Referenzmodell für den Einsatz vissensbasierter und wissensverarbeitender Funktionen innerhalb eines Krankenhaus-Informationssystems.* Habilitationsschrift, Fachbereich Humanmedizin der Justus-Liebig-Universität Gießen, 1994.

[pro95] prompt! Medizinische Informationssysteme GmbH. Leistungsbeschreibung prompt! HL7-Connection-Server. Prospekt, 1995.

[RBP+91] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy und W. Lorensen. *Object-Oriented Modeling and Design.* Prentice Hall Englewood Cliffs, New Jersey 07632, 1991.

[Sch88] M. Schwartz. *Telecommunication Networks: Protocols, Modeling and Analysis.* Addison-Wesley Publishing Company, 1988.

[Sch93] A. Schill. *DCE — Das OSF Distributed Computing Environment.* Springer, 1993.

[SL90] A. P. Sheth und J. A. Larson. Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. *ACM Computing Surveys*, 22(3):183 – 236, September 1990.

[Sof95a] Software Technologie Corporation. DataGate Documentation Version 2.6.2. Benutzeranleitung zur Version 2.6, 91066 Arcadia, California, USA, Januar 1995.

[Sof95b] Software Technologie Corporation. DataGate Documentation Version 3.0. Benutzeranleitung zur Version 3.0, 91066 Arcadia, California, USA, März 1995.

[Sof96] SoftCon. Hospital Communication System (HCS). Benutzerhandbuch zur Version 1.1, Oberhachingen, 1996.

[Tan96] A. S. Tanenbaum. *Computer Networks.* Prentice Hall, 1996.

Index