Anforderungsanalyse an einen
konfigurierbaren HL7-Kommunikations-Server
mittels OMT und ausführbarer Modelle*

W. Hasselbring A. Kröber
Universität Dortmund, Informatik 10 (Software-Technologie), D-44221 Dortmund
Tel. 49-(231)-755-4712, Fax: 49-(231)-755-2061
E-Mail: {willi|kroeber}@ls10.informatik.uni-dortmund.de

1 Einleitung

Am Lehrstuhl für Software-Technologie der Universität Dortmund werden zur Zeit in enger Kooperation mit einigen Krankenhäusern verschiedene Komponenten für Krankenhausinformationssysteme (KIS) entwickelt. Dabei hat sich unmittelbar die Notwendigkeit ergeben, vorhandene Systeme mit den neuen Komponenten zu integrieren. Als erster Schritt in diese Richtung wurde die in diesem Beitrag vorgestellte Anforderungsanalyse an einen konfigurierbaren HL7-Kommunikations-Server durchgeführt.
Zur Standardisierung der Datenübertragung in verteilten KIS wurde das HL7-Protokoll [4, 6] entworfen, das in den USA von praktisch allen kommerziellen Applikationen für Krankenhausabteilungen unterstützt wird. In Deutschland wird dieses Protokoll bisher nur von wenigen Applikationen unterstützt. Das Ziel des in diesem Beitrag beschriebenen Projekts ist die Modellierung eines konfigurierbaren HL7-Kommunikations-Servers, der sowohl die Kommunikation mit dem HL7-Protokoll als auch mit anderen Protokollen zwischen vorhandenen Abteilungssystemen eines KIS gewährleisten kann. Der Server soll leicht in verschiedene Konfigurationen eingepaßt werden können. Auch sollen Lösungen für noch vorhandene Schwachstellen des HL7-Protokolls gefunden werden; insbesondere für das fehlende *Multicasting*, also das Versenden einer Nachricht an mehrere Empfänger. Die Integrietät der Daten, die z.B. in föderativen Datenbanken [10] wichtig ist, kann durch einen Kommunikations-Server nicht gewährleistet werden.
Das wesentliche Ziel dieses Projekts ist es, aus den isolierten Informationssystemen einzelner Krankenhausabteilungen mit Hilfe des Kommunikations-Servers ein verteiltes, kooperierendes Informationssystem schaffen zu können. Ein wichtiger Aspekt ist dabei die Möglichkeit zur Integration von existierenden heterogenen Systemen (legacy systems) mit modernen Informationssystemen.
Die Modellierung erfolgt mittels der objekt-orientierten Analysemethode OMT (Object Modeling Technique) [9]. In diesem Beitrag wird eine Unterscheidung zwischen erster Anforderungsanalyse und erweiterter Anforderungsanalyse vorgenommen. Zur Bewertung der ersten

*In: Tagungsband 2. GI/ITG Arbeitstreffen Entwicklung und Management verteilter Anwendungssysteme, Krehl Verlag, Münster, Reihe Informatik für Systementwickler, Oktober 1995

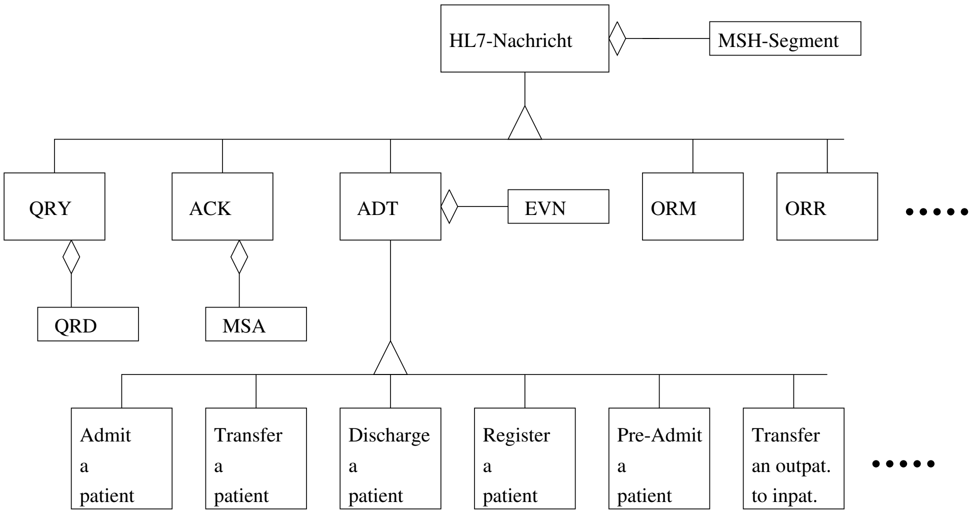

Abbildung 1: Ein Ausschnitt der Klassifikationshierarchie für HL7-Nachrichten.
Rechtecke stellen in der OMT-Notation Klassen dar, Dreiecke kennzeichnen Vererbungsrelationen und spitze Rauten Teil-von-Relationen.

Anforderungsanalyse, die mittels OMT erstellt wird, wird durch ausführbare Modelle (Prototypen) die Angemessenheit und Machbarkeit überprüft. Ziel ist auch die Evaluation der Technik, die objekt-orientierte Analyse mit explorativem Prototyping zur Analyse der Anforderungen an verteilte Anwendungssysteme zu kombinieren.

2 Das HL7-Protokoll (Version 2.2)

Die Abkürzung HL7 steht für Health Level 7, ein Protokoll zur Übertragung von medizinischen Daten in Krankenhäusern, welches Ebene 7 der ISO/OSI-Protokollhierarchie [11] zuzuordnen ist. Von HL7 werden diverse Bereiche des Informationsaustausches in KIS abgedeckt, wie z.B. Einlieferung, Entlassung und Transfer von Patienten, sowie der Austausch von Analyse- und Behandlungsdaten.
Eine HL7-Nachricht ist eine Zeichenkette, die aus einer Menge von obligaten und optionalen Segmenten besteht. Diese Segmente bestehen wiederum aus Feldern. Die Syntax einer Nachricht ist durch die Protokollbeschreibung festgelegt. Falls Informationen zu übermitteln sind, die nicht im Protokoll spezifiziert wurden, können sogenannte Z-Segmente angehängt werden, die frei definierbar sind.
Um einen schnellen Einblick in die Klassifizierung der HL7-Nachrichtentypen zu bekommen, haben wir zunächst die informelle Beschreibung der HL7-Nachrichtentypen aus [6] analysiert und dafür eine Klassifikationshierarchie in der OMT-Notation erstellt. Ein Ausschnitt der Klassifikationshierarchie ist in Abbildung 1 angegeben. Für die vollständige Hierarchie sei auf [8] verwiesen.
Leider ist das HL7-Protokoll nicht ausgereift genug, um alle Bedürfnisse von Krankenhausapplikationen abzudecken. Viele Bereiche, wie die Bilddatenübertragung, fehlen. In Zukunft sollen zwar HL7 und ACR/NEMA [7], ein Protokoll zur Übertragung komprimierter Bilddaten, im MEDIX-Standard integriert werden, doch vorläufig gibt es in der HL7-Literatur keine Vorgabe, wie die beiden Protokolle zu verbinden sind. Auch ist der Bereich der Abrechnungsdaten

nur auf das amerikanischen System zugeschnitten, so daß deutsche HL7-Applikationshersteller viele Informationen in nicht standardisierte Z-Segmente packen müssen. Desweiteren sieht HL7 keine Möglichkeit vor, eine Nachricht an mehrere Empfänger zu senden (Multicasting). Solange diese Schwachstellen nicht vom HL7-Gremium beseitigt werden, müssen dazu andere Methoden verwendet werden.

3 Anforderungen an einen HL7-Kommunikations-Server

Der wichtigste Aspekt bei der Spezifikation des HL7-Kommunikations-Servers ist die Konfigurierbarkeit. Der Einsatz eines Servers in verschiedenen Konfigurationen erfordert eine möglichst hohe Flexibilität seiner Schnittstellen. Voraussetzung hierfür ist die Kenntnis der verschiedenen HL7-Protokollversionen (Version 2.1 und 2.2) und deren landesspezifischen Anpassungen, sowie die Fähigkeit, mit verschiedenen herstellerabhängigen Formaten umgehen zu können. In Abbildung 2 ist eine mögliche Konfiguration eines HL7-Kommunikations-Servers dargestellt. In diesem Szenario sind jeweils ein Rechner des Labors, der Radiologie, einer Station, der Verwaltung und der Apotheke über den Server miteinander verbunden. Der Server besteht aus der zentralen Server-Komponente (ZSK) und Interface Agenten (IA) für jede Applikation. Die IA transformieren die Nachrichten in eine für die ZSK verwertbare Form. Nachrichten, die von der ZSK kommen, werden von den IA in eine für die Applikationen verwertbare Form umgewandelt.

Die Fähigkeit, ein heterogenes Netz zu steuern, hängt auch von applikationsspezifischen Eigenheiten ab. Der Server darf keine Probleme mit der Kommunikationsart der integrierten Komponenten haben. Die Integration alter Punkt-zu-Punkt-Verbindungen, die von einem bestimmten Kommunikationspartner ausgehen, also keine Adresse vorraussetzen, muß möglich sein. Desweiteren sollte neben der asynchronen Kommunikation die synchrone Kommunikation beherrscht werden, bei der eine sendende Applikation erst dann weiterarbeitet, nachdem sie eine Antwort auf die zuletzt verschickte Nachricht erhalten hat.

Zur Modellierung der Anforderungen und des Entwurfs wird OMT verwendet (siehe Abschnitt 3.1). Daraufhin werden diverse Szenarien für verschiedene Möglichkeiten der Datenübertragung in einem KIS studiert. Diese Szenarien werden mittels ausführbarer Modelle (Prototypen) evaluiert, um bei auftretenden Problemen das initiale OMT-Modell entsprechend zu modifizieren (siehe Abschnitt 3.2). Dieser Zyklus wird solange wiederholt, bis die Anforderungen stabil erscheinen, um dann die nächsten Phasen der Entwicklung zu starten. Die Aspekte des durch das Prototyping erweiterten Modells werden in Abschnitt 3.3 diskutiert.

3.1 Erste Anforderungsanalyse und Entwurf mittels OMT

Mit OMT wird zunächst ein *Objektmodell* zur Identifizierung aller beteiligten Klassen entworfen, das die statische Struktur beschreibt [9]. Das *dynamische Modell* dient dann zur Festlegung der Dynamik einzelner Objekte und das *funktionale Modell* stellt den Datenfluß dar.

In Abbildung 3 ist ein Ausschnitt des Objektmodells angegeben, welches die Integration von Applikationen mit einem HL7-Kommunikations-Server darstellt. Jede Applikation kommuniziert mit einem Interface-Agenten (IA). Dieser fügt vor die Nachricht einen internen Präfix, so daß der Server die Nachricht geeignet weiterleiten kann. Diese erweiterte Nachricht wird dann vom Server anhand von verschiedenen Tabellen bearbeitet und weitergeleitet. Diese Tabellen dienen der Konfiguration des Servers und werden in der zentralen Server-Komponente

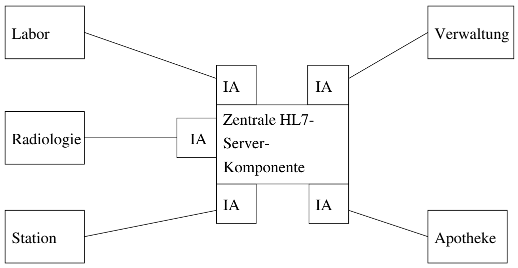

Abbildung 2: Eine mögliche Konfiguration für die Integration eines HL7-Kommunikations-Servers. In dieser Abbildung wird zum besseren Verständnis noch keine OMT-Notation verwendet.

verwaltet (Routing-Tabellen, Protokolltabellen, IA-Tabellen, Multicasting-Tabellen etc). Die Interface-Agenten sind Teil des Servers.

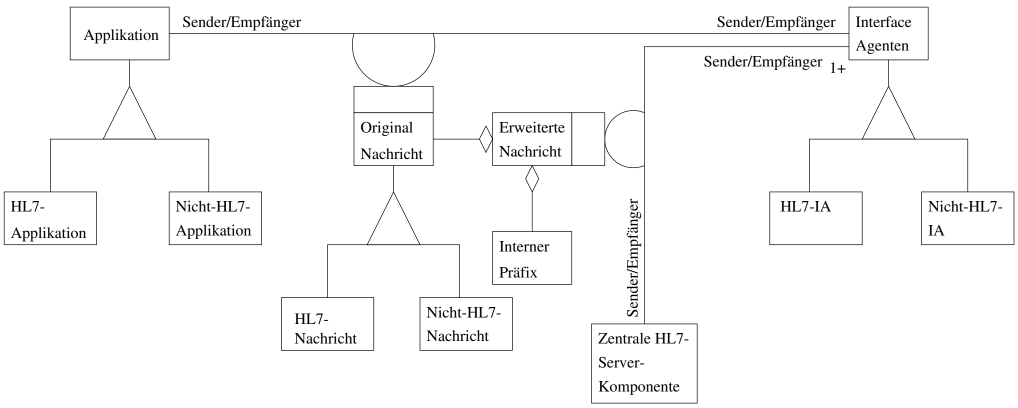

Abbildung 3: Erstes OMT-Modell der Integration von Applikationen mit einem HL7-Kommunikations-Server.
Linkattribute werden in OMT als Klassen dargestellt, die durch Schleifen an die zugehörigen Assoziationen gehängt werden. Beschriftungen an den Enden von Assoziationen geben Auskunft über die Rollen der Klassen in diesen Assoziationen. Siehe [9] für eine genaue Beschreibung der Semantik von OMT-Modellen.

Die erste Modellierung dieser globalen Sicht des Nachrichtenaustauschs mit Hilfe eines HL7-Kommunikations-Servers gestaltete sich einfach, da die Anforderungen recht klar sind.
Das für Netzwerke üblicherweise wichtige Multicasting von Nachrichten, welches von HL7 nicht unterstützt wird, kann mit Hilfe des Kommunikations-Servers realisiert werden. Dies sollte sich allerdings nicht auf normale Nachrichten beschränken, vielmehr sollte auch für Anfragen Multicasting möglich sein. Bei den Antworten auf Anfragen muß eine Lösung gefunden werden, um mögliche Inkonsistenzen durch unterschiedliche und fehlende Antworten zu vermeiden. Desweiteren sollte wegen der unterschiedlichen Wichtigkeit bestimmter Nachrichten eine Priorisierungsmöglichkeit im Server vorhanden sein. Bei der Verfeinerung der dynamischen

und der funktionalen Aspekte stellte sich hierbei heraus, daß eine genaue Modellierung mittels OMT kaum möglich ist, da die sinnvolle Funktionsweise noch nicht bekannt ist.

3.2 Anforderungsanalyse durch ausführbare Modelle

Mit OMT konnten noch nicht alle Anforderungen vollständig erfaßt werden, weil insbesondere eine vollständige analytische Modellierung der genauen Anforderungen an das nebenläufige Abarbeiten beim Multicasting von Nachrichten kaum möglich ist. Um die verbliebenen Lücken zu füllen, kombinieren wir die objekt-orientierte Analyse mit dem Prototyping-Ansatz zur Anforderungsanalyse [12]. Prototyping beschäftigt sich mit dem Entwurf, der Konstruktion und der Evaluierung von Prototypen, d.h. von *frühzeitig* ausführbaren Modellen des späteren Produktes.

Das Prototyping von parallelen Algorithmen (hier das nebenäufige Abarbeiten beim Multicasting) soll durch die Bereitstellung von Konstrukten mit hohem Niveau zur expliziten Parallelprogrammierung die Entwicklung paralleler Anwendungen wesentlich erleichtern, indem in den frühen Phasen zur Analyse der Anforderungen und der Machbarkeit mit Ideen für parallele Algorithmen praktisch experimentiert werden kann. In der vorliegenden Arbeit wurde PROSET-Linda [5] zur Prototypimplementation verwendet.

PROSET-Linda ist eine parallele Prototyping-Sprache, die ein flexibles Konzept zur Synchronisation und Kommunikation durch sogenannte *Tupelräume* bietet. Diese Tupelräume sind virtuelle gemeinsame Datenräume, über die die Prozesse kommunizieren können. Synchronisation und Kommunikation erfolgen in PROSET-Linda durch das Einfügen, Entfernen, Lesen und durch unteilbares Ändern von einzelnen Tupeln im Tupelraum. PROSET-Linda unterstützt im Gegensatz zu C-Linda [3] das Arbeiten mit mehreren Tupelräumen, um Modularität für die Kommunikation zu ermöglichen.

Die Konfiguration für die Integration eines verteilten KIS mittels eines HL7-Kommunikations-Servers aus Abbildung 2 ist als Prototyp implementiert. Durch das hohe sprachliche Niveau von PROSET-Linda ist es nun relativ leicht möglich, verschiedene Konzepte für das nebenläufige Abarbeiten beim Multicasting von Nachrichten schnell und kompakt zu implementieren und die Stärken und Schwächen dieser Konzepte mittels Ausführung der Prototypen praktisch zu evaluieren.

Um prinzipiell den Aufbau des Prototypen zu verstehen, sind in Abbildung 4 die verwendeten parallel laufenden Prozesse und deren Tupleräume (TR) zur Interprozeßkommunikation dargestellt. Jede simulierte Applikation kommuniziert für jede Übermittlungsrichtung unter Verwendung eines Tupleraums mit ihrem Interface-Agenten. Die Interface-Agenten kommunizieren wiederum über einen gemeinsamen TR mit der zentralen Server-Komponente (ZSK). In Abbildung 4 ist nur eine Applikation dargestellt. Weitere IA und Applikationen werden über den gemeinsamen Server-Tupelraum mit der ZSK verbunden.

Der Tupelraum zwischen den Interface-Agenten und dem Server kann zwei unterschiedliche Arten von Tupeln enthalten. Wenn ein Tupel an den Server geschickt wird, enthält es zwei Komponenten: Die Nachrichtenpriorität und die erweiterte Nachricht. Falls die Nachricht vom Server kommt, enthält sie den Bezeichner für die Zielapplikation und die Nachricht selbst. Zwischen Applikationen und deren Interface-Agenten werden nur die eigentlichen Nachrichten übertragen.

Konfigurierbar ist der HL7-Kommunikations-Server über Tabellen, z.B. könnte die IA-Tabelle in PROSET-Notation folgendes enthalten:

```
IA_Param_Table:={["App1","10"],
```

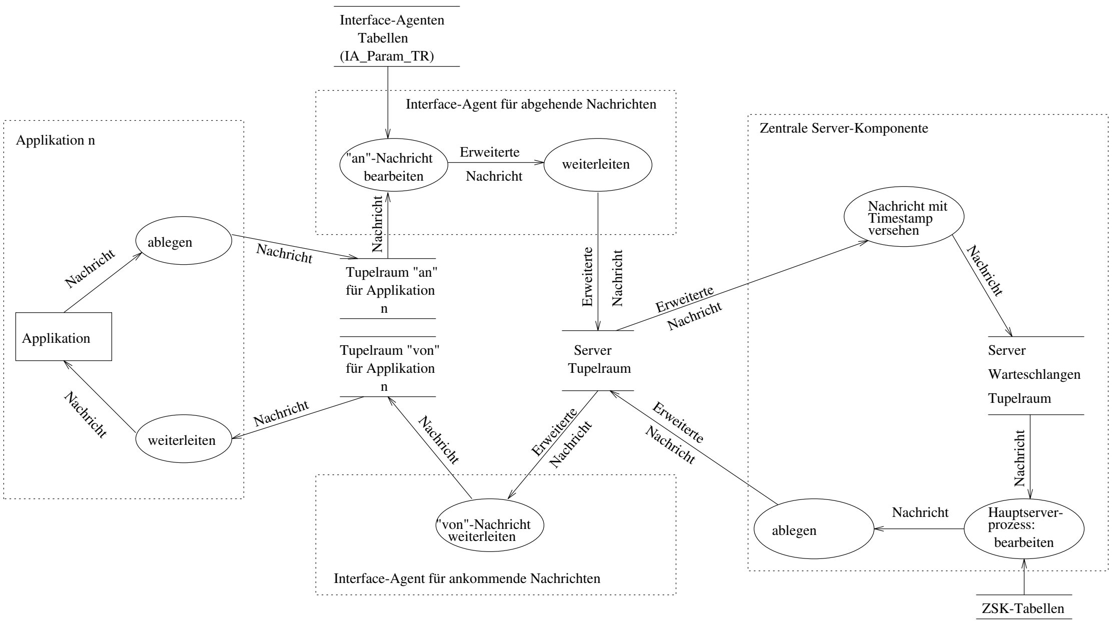

Abbildung 4: Die Interprozeßkommunikation des Protoypen.
Zur Darstellung wird die Datenflußdiagramm-Notation von OMT verwendet. Die Rahmen um die einzelnen Systemkomponenten sind keine OMT-Syntax und dienen nur der Steigerung der Übersichtlichkeit. Dem Datenspeicher 'Interface-Agenten Tabellen' können die IA Eigenschaften der Applikationen entnehmen, wie z.B. den Protokolltypen und die Applikationspriorität. Im Datenspeicher ZSK-Tabellen befinden sich alle für die ZSK notwendigen Konfigurationstabellen, wie z.B. die Routing- und die Multicasting-Listen.

```
["App2", "5"],
["App3", "5"]};
```

Für jeden TR gibt es zwei Felder: den Namen der Applikation und die Priorität der Applikation. Die einzelnen Elemente der Tabelle werden im Tupelraum `IA_Param_TR` abgelegt:

```
for entry in IA_Param_Table do
  deposit entry at IA_Param_TR end deposit;
end for;
```

Jeder Interface-Agent kann dann seine Parameter dem Tupelraum entnehmen.

3.3 Erweiterte Anforderungsanalyse und Entwurf mittels OMT unter Berücksichtigung der Ergebnisse des Prototypings

Durch die Konstuktion und Analyse der Prototypen sind die Anforderungen klarer geworden. Die interne Funktionsweise des Servers kann nun besser erfaßt werden und somit in ein erweitertes Modell umgewandelt werden. Bei der Analyse der Prototypen sichtbar bewordene Probleme können bei der Formulierung des OMT-Modells vermieden werden. Aufgefallen sind auch ungeschickte Modellierungen im ersten OMT-Modell. Z.B. werden die IA jetzt in zwei unabhängige Prozesse aufgeteilt, um Synchronisationsprobleme zu vermeiden. In der ersten Modellierung bestand jeder IA nur aus einem Prozeß, der dann durch Polling [11] auf Nachrichten aus beiden Richtungen warten mußte. Die Evaluation des entsprechenden Prototypen zeigte, daß diese Konstruktion fehleranfällig und ineffizient ist. Insbesondere die Analyse des nebenläufigen Abarbeitens hat gezeigt, daß Multicasting von Nachrichten bei synchroner Kommunikation nicht zufriedenstellend lösbar ist. Multicasting wird deswegen nur noch bei asynchroner Kommunikation beachtet werden.

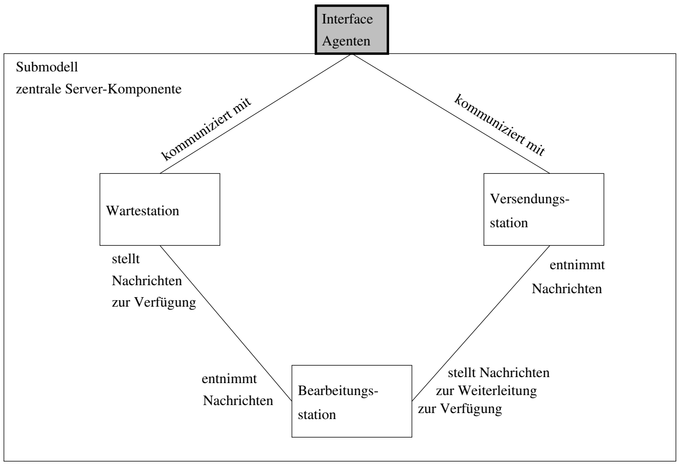

Abbildung 5: Das Submodell 'ZSK'. Die Ports werden grau unterlegt dargestellt.

Das ist das Ziel des explorativen Prototyping: Experimentieren mit Ideen für Algorithmen und Evaluation dieser Algorithmen, um frühzeitig die richtigen Entscheidungen treffen zu können. Rein analytische Bewertungen sind im allgemeinen nicht machbar.
Bei der Konstruktion großer Systeme ist es vorteilhaft, Modelle hierarchisieren zu können. Da OMT diese Möglichkeit nicht explizit zur Verfügung stellt, wurde die Notation um Submodelle erweitert. Jede Klasse eines Modells kann durch Submodelle verfeinert werden. Die Assoziationen zwischen externen Klassen und Submodell müssen durch *Ports* im Submodell dargestellt werden und mit internen Klassen verbunden sein. Dieser Ansatz zur Hierarchisierung von OMT-Modellen wurde auch in [2] verwendet. Jede Assoziation besitzt genau einen Port. Replikationen der Assoziationen, die von den Ports ausgehen, sind möglich. Attribute an Assoziationen werden im Submodell nicht erneut dargestellt. Vererbungs- und Aggregationskanten [9] müssen entsprechend über Ports im Submodell verbunden werden. Im Prinzip stellt diese Hierarchisierungsform einen Makro-Mechanismus dar. Die Submodelle könnten im Vatermodell expandiert werden. Abbildung 5 stellt die Verfeinerung der ZSK aus dem schon vorher in der ersten Anforderungsanalyse dargestellten globalen Objektmodell aus Abbildung 3 dar.
Die ZSK besteht aus einer Wartestation, einer Versendungsstation und einer Bearbeitungsstation, die wiederum verfeinert werden [8]. Die Bearbeitungsstation besteht aus einer Nachrichtenbearbeitungsstation, deren dynamisches Modell in Abbildung 6 dargestellt ist, und einer Anfragebearbeitungsstation.
Falls in der Nachrichtenbearbeitungsstation eine normale Nachricht ankommt, wird eine Sendeliste für die Versendungsstation und eine Antwortliste aufgebaut. Falls eine Bestätigung ankommt wird getestet, ob alle Antworten für die Nachricht angekommen sind. Falls ja, wird

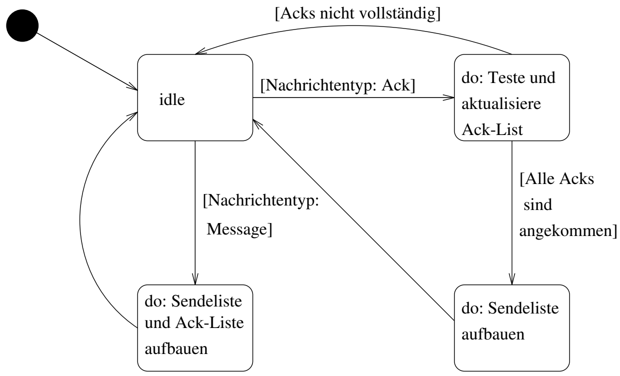

Abbildung 6: Das dynamische Modell der Nachrichtenbearbeitungsstation.

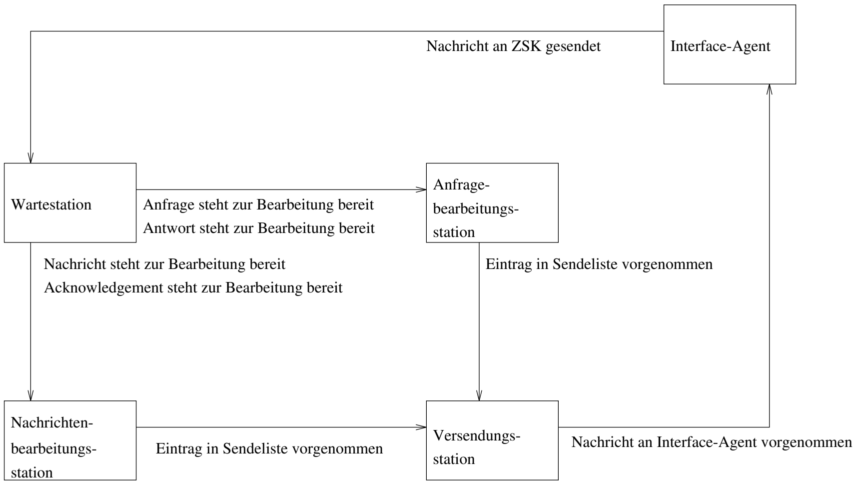

Abbildung 7: Das Ereignisflußdiagramm des HL7-Kommunikations-Servers.

die gemeinsame Acknowledgement-Nachricht in die Sendeliste eingetragen, falls nein, wird nur der Status der Ack-List geändert. Für jede Klasse, die ein nicht-triviales dynamisches Verhalten besitzt, gibt es ein dynamisches Modell. Anhand eines Ereignisflußdiagrammes wird die Dynamik des Gesamtmodells dargestellt. In Abbildung 7 finden wir alle Klassen des Servers als Ereignisflußdiagramm dargestellt. Auf den Links zwischen den einzelnen Klassen sind die Ereignisse, die Aktionen in anderen Klassen auslösen, dargestellt.

In dem funktionalen Modell in Abbildung 8 wird die interne Funktionsweise der Nachrichtenbearbeitungsstation festgelegt. Nach Analyse des Nachrichtentyps werden bei Nachrichten die Adressen der Applikationen, an die die Nachricht multicasted werden soll, der 'Multicasting-Liste' entnommen. Die dabei erhaltene Adreßliste wird an den Prozeß 'Sendeliste aufbauen' weitergeleitet. Dieser legt dann die erstellten Nachrichten in dem Datenspeicher 'Sendeliste' ab. Für eine detaillierte Beschreibung der Prototypimplementation und der Modellierung sei auf [8] verwiesen.

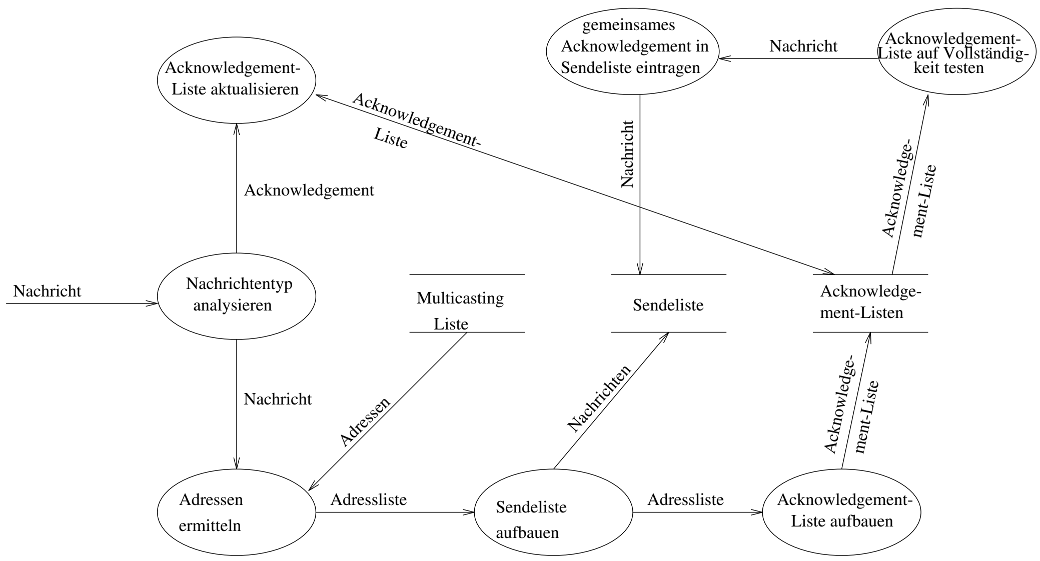

Abbildung 8: Das funktionale Modell der Nachrichtenbearbeitungsstation aus Abbildung 6.

4 Erfahrungen

Dieser Beitrag hat verschiedene Probleme und Lösungsansätze bei der Anforderungsanalyse für ein sehr spezielles verteiltes System aufgezeigt. Ein HL7-Kommunikations-Server, der über verschiedene Tabellen konfiguriert wird, soll die vorhandenen Schwachstellen des HL7-Protokolls überwinden helfen und existierende Applikationen, die das HL7-Protokoll nicht unterstützen, integrieren können, um so Abteilungssysteme eines KIS integrieren zu können. In diesem Beitrag konnten aus Platzgründen nur Teilaspekte skizziert werden.
Neben dieser Problematik wird die Kombination der objekt-orientierten Analyse- und Entwurfsmethode OMT mit dem Prototyping-Ansatz zur Anforderungsanalyse verteilter Systeme exemplarisch untersucht. Hier hat sich gezeigt, daß OMT zwar gut zur Analyse und zum Entwurf von Systemen mit *bekannten* Eigenschaften geeignet ist, doch bei einem System wie dem HL7-Kommunikations-Server, bei dem die gewünschten Eigenschaften noch nicht vollständig bekannt sind, bietet sich exploratives Prototyping, als geeignete Ergänzung an.
Grundsätzlich stellt sich die Frage, inwieweit OMT überhaupt zur Anforderungsanalyse eines neuen Systems geeignet ist. Zumindest die Notation unterstützt nur den Entwurf. Die Methodik, z.B. aus einem natürlichsprachigen Text zuerst die Substantive herauszusuchen, um daraus die Klassen zu bestimmen, wird in [9] nur informell behandelt. Es gibt jedoch keine explizite Unterstützung zur Anforderungsanalyse in OMT. Im hier beschriebenen Projekt wurde deshalb diese Vorgehensweise zum Herauszusuchen der Substantive nicht eingesetzt.
Embley et al. [1] haben dazu festgestellt, daß die existierenden objekt-orientierten Analyse- und Entwurfsmethoden (einschließlich OMT) keine angemessene Unterstützung für die Analyse bieten, da die Analyse bei diesen Methoden zu sehr mit dem Entwurf und der Implementation verwoben ist:

> “Analysis is neither design nor implementation. Analysis focuses on real-world problems, whereas design and implementation focus on computerized solutions. Care must be taken during analysis not to begin devising solutions . . . ” [1, Seite 19]

Vorgeschlagen wird in [1] eine neue Analysemethode, Object-Oriented Systems Analysis (OSA), die sich ausschließlich mit der Analyse befaßt. OSA ist ausführbar, da alle Modellierungskomponenten eine formale Syntax und Semantik haben. Dadurch wird auch das Prototyping direkt durch die Analysemethode unterstützt. In unserer Anforderungsanalyse mußten wir noch verschiedene Methoden für die objekt-orientierte Analyse und für das Prototyping verwenden. OMT wurde im wesentlichen zum Entwurf der u.a. durch das Prototyping gewonnenen Erkenntnisse verwendet.

Danksagung

An dieser Stelle sei Thomas Biedassek, Stefan Dissmann, Ernst-Erich Doberkat, Wolfgang Emmerich und Doris Schmedding für die Anmerkungen zu früheren Versionen dieses Beitrags gedankt.

Literatur

[1] D.W. Embly, R.B. Jackson und S.N. Woodfield. OO Systems Analysis: Is It or Isn't It? *IEEE Software*, 12(3):19–33, Juli 1995.

[2] W. Emmerich. *Tool Construction for Process-Centered Software Development Environments based on Object Database Systems*. Dissertation, Universität Paderborn, 1995.

[3] D. Gelernter. Generative communication in Linda. *ACM Transactions on Programming Languages and Systems*, 7(1):80–112, Januar 1985.

[4] W.E. Hammond. Health Level 7: A protocol for the interchange of healtcare data. In G.J.E. De Moor, C.J. McDonald und J.N. van Goor, Hrsg., *Progress in Standardization in Health Care Informatics*, Seiten 144–148. IOS Press, 1993.

[5] W. Hasselbring. *Prototyping Parallel Algorithms in a Set-Oriented Language*. Dissertation, Universität Dortmund, 1994. (Erschienen im Verlag Dr. Kovač, Hamburg).

[6] Health Level Seven, Inc, Ann Arbor, USA. *Health Level Seven: an application protocol for electronic data exchange in healthcare environments, version 2.2*, Dezember 1994.

[7] C. Jostes, J. Paczkowski und J. Schröder. Ganzheitliche Medizin. *IX*, Juli 1993.

[8] A. Kröber. Modellierung eines konfigurierbaren HL7-Kommunikations-Servers. Diplomarbeit, Universität Dortmund, FB Informatik, 1995. (in Vorbereitung).

[9] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy und W. Lorensen. *Objektorientiertes Modellieren und Entwerfen*. Hanser Verlag, 1994.

[10] A. Sheth und J. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183–236, 1990.

[11] A.S. Tanenbaum. *Computer Networks*. Prentice Hall, 1989.

[12] H. Züllighoven, W. Altmann und E.-E. Doberkat, Hrsg. *Requirements Engineering '93: Prototyping*, Band 41 von *Berichte des German Chapter of the ACM*. Teubner-Verlag, April 1993.