Anforderungsanalyse durch Kombination von OMT mit einem Ansatz zum explorativen Prototyping

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

Zusammenfassung

Der vorliegende Beitrag diskutiert Erfahrungen, die bei der Anforderungsanalyse für einen konfigurierbaren Krankenhauskommunikations-Server mit der Kombination von OMT und explorativem Prototyping gewonnen wurden. Eingesetzt wurde eine Vorgehensweise, die einige Schwächen der OMT-Methodologie in der Anforderungsanalyse beheben soll. Auch die Probleme mit der Spezifikation verteilter Systeme mittels OMT sowie die fehlenden graphischen Hierarchisierungsmöglichkeiten in der Notation für das OMT-Objektmodell werden diskutiert. Ein Konzept zur Hierarchisierung wird vorgestellt.

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 Krankenhauskommunikations-Server durchgeführt.

Zur Standardisierung der Datenübertragung in verteilten KIS wurde das HL7-Protokoll [10, 12] 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 Integrität der Daten, die z.B. in föderativen Datenbanken [21] 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 objektorientierten Methode OMT (Object Modeling Technique) [20]. Es ist bekannt, daß OMT noch über einige Schwächen verfügt:

- Die OMT-Notation liefert nur unzureichende Unterstützung für die Analyse der Anforderungen. Es ist eher eine Entwurfsnotation. Andere Ansätze, wie OOSE [14], bieten auch durch die Spezifikationsmethoden Unterstützung für die Anforderungsanalyse.
- OMT glänzt insbesondere in der Datenmodellierung durch ein erweitertes E/R-Modell, das hier Objektmodell genannt wird. Es fehlen allerdings Möglichkeiten zur hierarchischen Strukturierung von Objektmodellen. Für das in diesem Beitrag beschriebene Projekt wurde das OMT-Objektmodell daher um hierarchische Klassendefinitionen erweitert, die z.B. mit den Strukturierungsmechanismen hierarchischer Petri-Netze [6] vergleichbar sind.
- Eine besondere Schwäche der OMT-Notation liegt in der unzureichenden Unterstützung zur Spezifikation von verteilten Systemen. Mit OMT werden Aspekte der Kommunikation und Synchronisation im dynamischen Modell durch Zustandsübergangsdiagramme, Ereignisflußdiagramme und Interaktionsdiagramme für Szenarien, sowie im funktionalen Modell durch Datenflußdiagramme, mit denen auch (eingeschränkt) Kontrollfluß spezifiziert werden kann, modelliert. Die Probleme damit werden später diskutiert.

Es ist bekannt, daß sich Prototyping gut zur Anforderungsanalyse einsetzen läßt [23] und daß objektorientierte Modellierung dann besonders geeignet ist, wenn das zu lösende Problem gut verstanden wird. Um die Vorteile beider Ansätze zu kombinieren, wird in diesem Beitrag vorgeschlagen, eine Unterscheidung zwischen *erster* Anforderungsanalyse und *erweiterter* Anforderungsanalyse vorzunehmen. In der ersten Anforderungsanalyse wird ein initiales Modell mittels OMT konstruiert. Im vorgestellten Projekt konnten hier nur einige grundlegende Strukturen identifiziert werden. Zur Bewertung und Erweiterung der ersten Anforderungsanalyse wird durch ausführbare Modelle (Prototypen) die Angemessenheit und Machbarkeit überprüft. Die Auswertung des Prototyping dient dann zur Verfeinerung der OMT-Modelle. Die erweiterte Anforderungsanalyse durch das Prototyping und die Verfeinerung der OMT-Modelle wird mehrfach durchlaufen. Wir präsentieren hier nur eine Iteration. OMT wird dabei im wesentlichen zur Beschreibung der durch das Prototyping gewonnenen Erkenntnisse verwendet.

Abschnitt 2 stellt das HL7-Protokoll vor, bevor in Abschnitt 3 die Anforderungsanalyse an einen HL7-Kommunikations-Server beschrieben wird. Eine Diskussion vergleichbarer Ansätze und einige Abschlußbemerkungen folgen in den Abschnitten 4 und 5.

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 [22] 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.

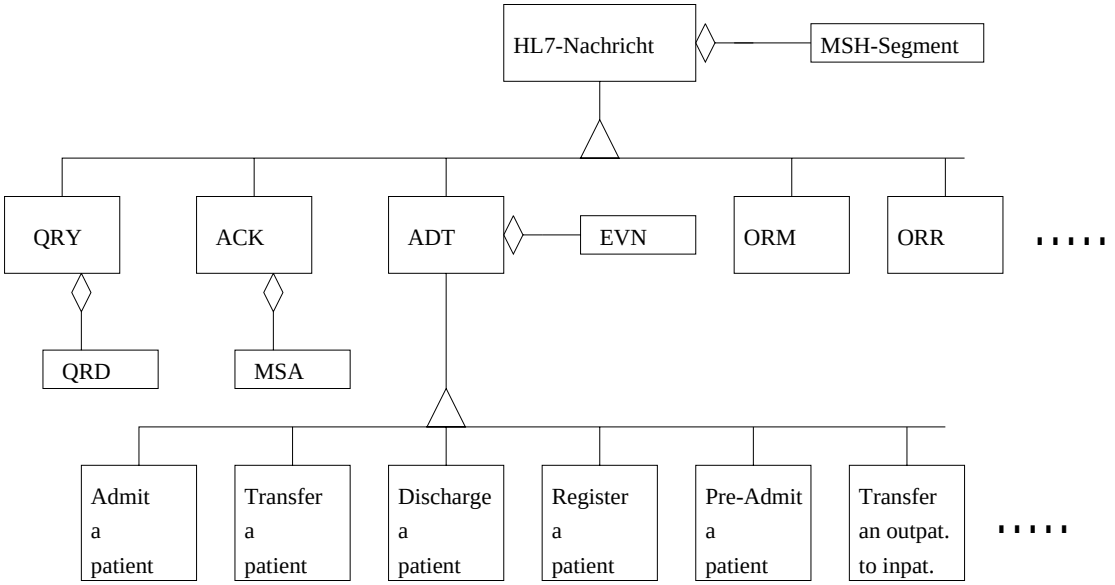


Abbildung 1: Ein Ausschnitt der Klassifikationshierarchie für HL7-Nachrichten. Rechtecke stellen in der OMT-Notation Klassen dar, Dreiecke kennzeichnen Vererbungsrelationen und spitze Rauten Teil-von-Relationen.

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 [12] 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 [17] 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 [15], 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 amerikanische System zugeschnitten, so daß deutsche HL7-Applikationshersteller viele Informationen in nicht standardisierte Z-Segmente packen müssen. Des weiteren sieht HL7 keine Möglichkeit vor, eine Nachricht an mehrere Empfänger zu senden (Multicasting). Solange diese Schwachstellen nicht vom HL7-Gremium beseitigt werden, müssen sie durch andere Methoden ausgeglichen werden.

3 Anforderungsanalyse an einen HL7-Kommunikations-Server

Ein wichtiger 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 System 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 jeweils einen 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 voraussetzen, muß möglich sein. Des weiteren 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. Das durch das Prototyping erweiterte Modell wird in Abschnitt 3.3 präsentiert.

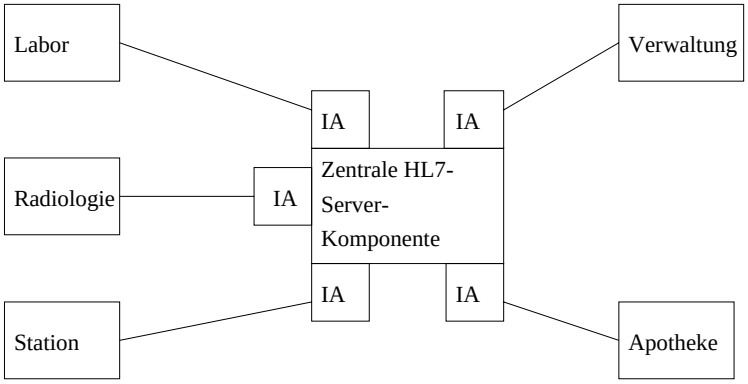


Abbildung 2: Eine mögliche Konfiguration für die Integration eines HL7-Kommunikations-Servers. In dieser Abbildung wird noch keine OMT-Notation verwendet.

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 [20]. Das *dynamische Modell* dient dann zur Festlegung der Dynamik einzelner Klassen und das *funktionale Modell* stellt den Datenfluß dar.

In Abbildung 3 ist ein Ausschnitt des ersten 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 ein internes 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 verwaltet (Routing-Tabellen, Protokolltabellen, IA-Tabellen, Multicasting-Tabellen etc). Die Interface-Agenten sind Teil des Servers.

Die erste Modellierung dieser globalen Sicht des Nachrichtenaustausches mit Hilfe eines HL7-Kommunikations-Servers gestaltete sich einfach, da die Anforderungen soweit 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. Weiterhin 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 schnell heraus, daß eine genaue Modellierung mittels OMT kaum möglich ist, da die sinnvolle Funktionsweise noch nicht bekannt ist.

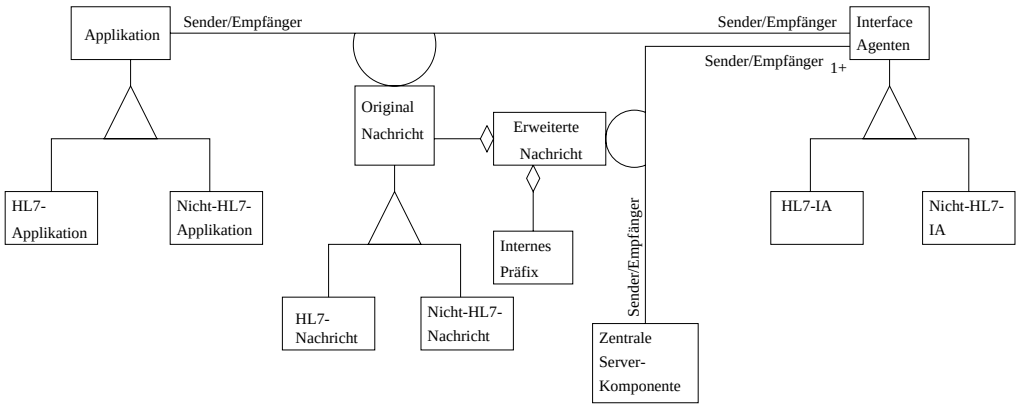


Abbildung 3: Erstes OMT-Modell der Integration von Applikationen mit einem HL7-Kommunikations-Server. Linkattribute werden in OMT als Klassen dargestellt, die durch Schleifen an die zugehörigen Assoziationen gehängt werden. Beschriftungen an den Enden von Assoziationen geben Auskunft über die Rollen der Klassen in diesen Assoziationen. Siehe [20] für eine ausführliche Beschreibung der OMT-Notation.

3.2 Anforderungsanalyse durch ausführbare Modelle

Mit OMT konnten noch nicht alle Anforderungen direkt 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 objektorientierte Analyse mit dem Prototyping-Ansatz zur Anforderungsanalyse [23]. 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 z.B. das nebenläufige 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 [11] 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 [7] 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 Tupelräume zur Interprozeßkommunikation dargestellt. Jede simulierte Applikation kommuniziert für jede Übermittlungsrichtung unter Verwendung eines Tupelraums mit ihrem Interface-Agenten. Die Interface-Agenten kommunizieren wiederum über einen gemeinsamen Tupelraum 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"],
                 ["App2", "5"],
                 ["App3", "5"]};
```

Für jeden Tupelraum 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. Die Evaluation der Prototypen wird im nächsten Abschnitt beschrieben.

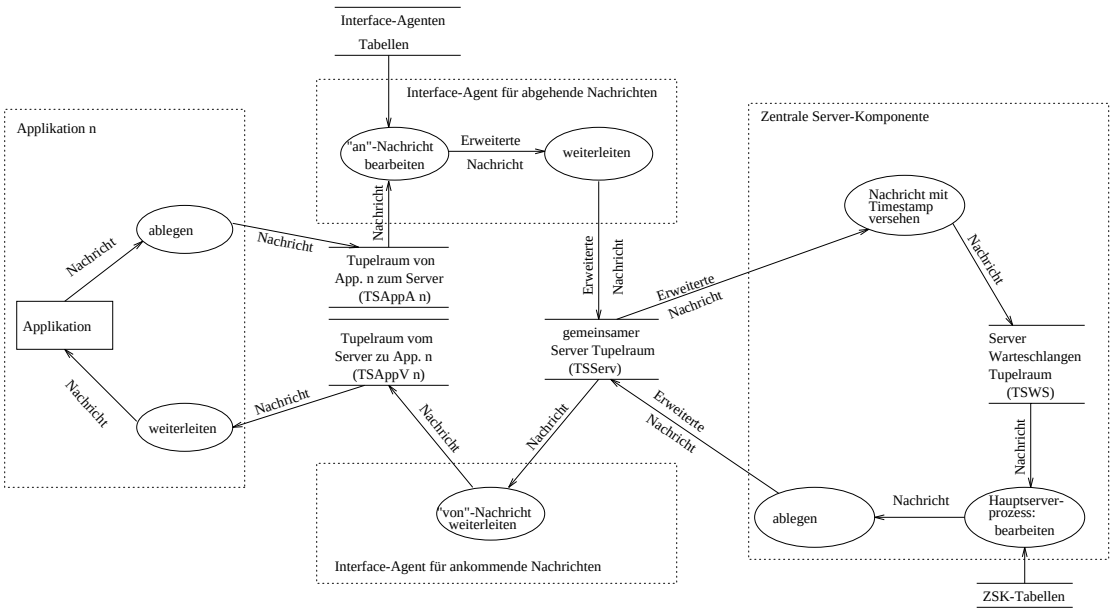


Abbildung 4: Die Interprozeßkommunikation des Protoypen. Zur Darstellung wird die Datenflußdiagramm-Notation von OMT verwendet. Die Rahmen um die einzelnen Systemkomponenten sind keine OMT-Syntax und dienen nur der Steigerung der Übersichtlichkeit. Dem Datenspeicher 'Interface-Agenten Tabellen' können die IAen 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.

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

Durch die Konstruktion 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 gewordene Probleme können bei der Formulierung des OMT-Modells vermieden werden. Aufgefallen sind auch ungeschickte Modellierungen im ersten OMT-Modell. Z.B. werden die IAen zur Vermeidung von Synchronisationsproblemen jetzt in zwei unabhängige Prozesse aufgeteilt.

In der ersten Modellierung bestand jeder IA nur aus einem Prozeß, der dann durch Polling [22] 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 berücksichtigt werden.

Das ist das Ziel des explorativen Prototypings: 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.

3.3.1 Das Objektmodell

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 ist mit den Hierarchisierungsmöglichkeiten hierarchischer Petri-Netze [6] vergleichbar. 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 [20] müssen entsprechend über Ports im Submodell verbunden werden. Im Prinzip stellt diese Hierarchisierungsform einen Makro-Mechanismus dar. Die Submodelle können im Vatermodell expandiert werden.

Mit Hilfe dieser Erweiterung und aufbauend auf den Ergebnissen des Prototypings wird die erweiterte Anforderungsanalyse mittels OMT durchgeführt. Abbildung 5 stellt das erweiterte globale Objektmodell dar. Unterschiede zu dem ursprünglichen Modell, welches in Abbildung 3 dargestellt war, sind die Ergänzung von Attributen und die Verfeinerung mit Submodellen. Zusätzlich wurden weitere Klassen für die ZSK und die IAen eingefügt. Die Klasse HL7-Nachricht wird über die HL7-Klassifikationshierarchie verfeinert, die aus Platzgründen in diesem Beitrag nicht dargestellt ist. Abbildung 6 stellt die Verfeinerung der ZSK dar. Die ZSK besteht aus einer Wartestation, einer Versendungsstation, einer Nachrichtenbearbeitungsstation und einer Anfragebearbeitungsstation.

3.3.2 Das dynamische Modell

Für jede Klasse, die ein nicht-triviales dynamisches Verhalten besitzt, wird mit OMT ein dynamisches Modell als Zustandsübergangsdiagramm spezifiziert.

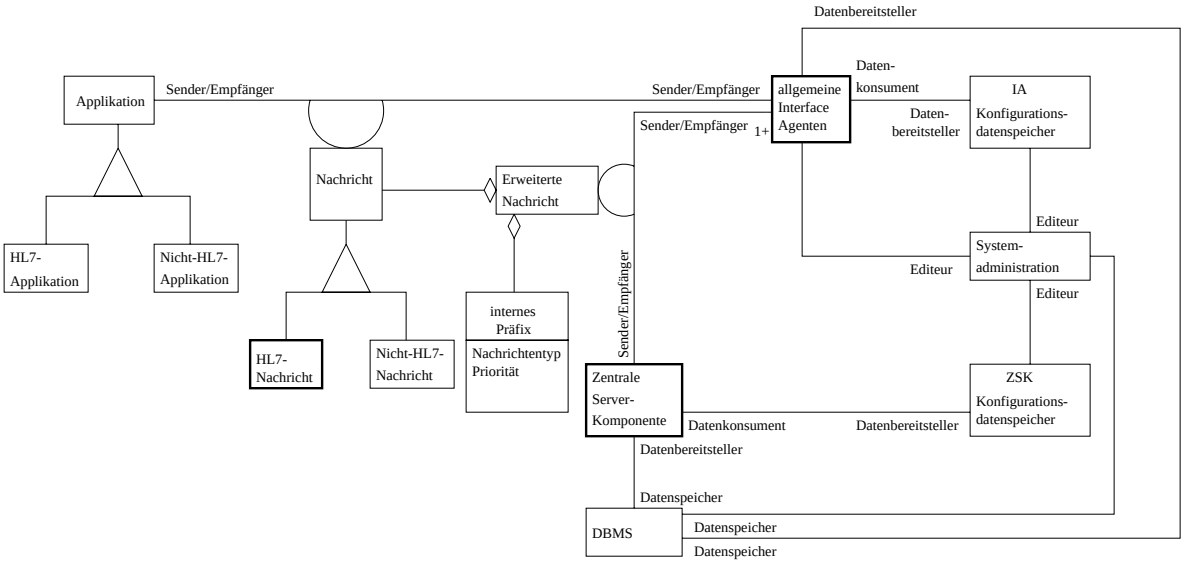


Abbildung 5: Das erweiterte globale OMT-Modell.

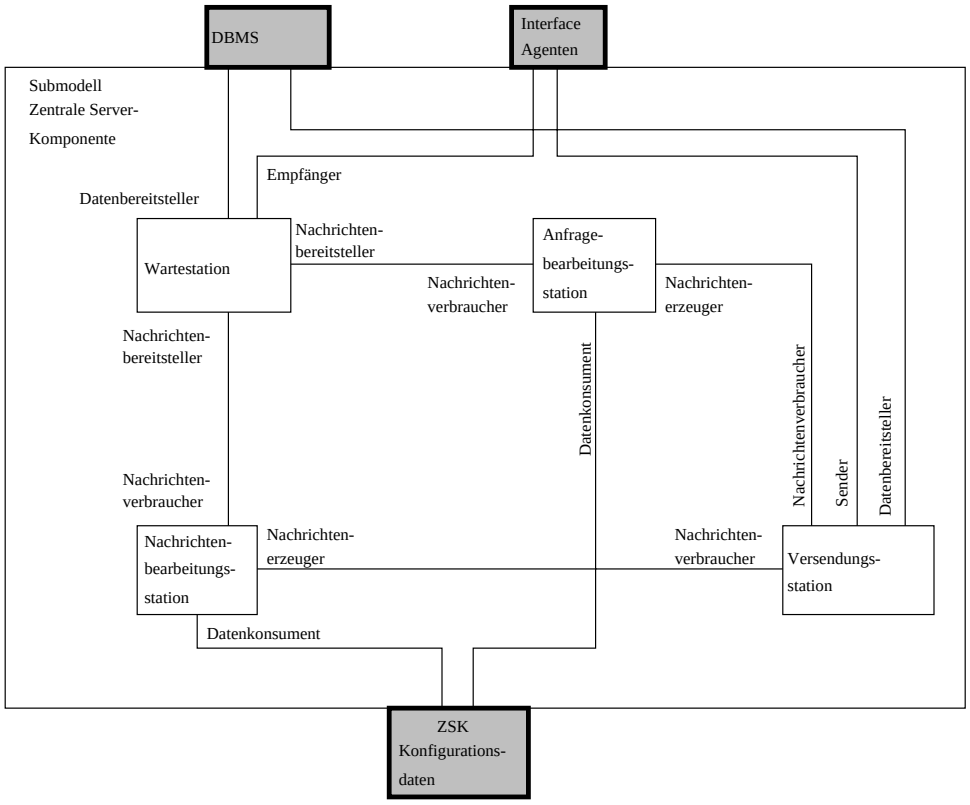


Abbildung 6: Das Submodell 'ZSK'. Die Ports werden grau unterlegt dargestellt.

In Abbildung 7 ist das dynamische Modell der Anfragebearbeitungsstation dargestellt. Als weiteres dynamisches Modell ist in Abbildung 8 das Modell der Versendungsstation dargestellt. Dieses Modell ist recht trivial und soll an dieser Stelle den Zusammenhang zwischen den einzelnen dynamischen Modellen aufzeigen. Aktivitäten werden in der Versendungsstation durch Ereignisse ausgelöst, die in der Anfragebearbeitungsstation auftreten. Um diesen globalen Zusammenhang zu veranschaulichen, werden von OMT Ereignisflußdiagramme zur Verfügung gestellt. Anhand eines Ereignisflußdiagrammes wird die Dynamik des Gesamtmodells dargestellt. In Abbildung 9 finden wir alle Klassen des Servers im Ereignisflußdiagramm dargestellt. Auf den Verbindungen zwischen den einzelnen Klassen sind die Ereignisse, die Aktionen in anderen Klassen auslösen, dargestellt. Das Ereignis 'Eintrag in Sendeliste' ist sowohl in der Versendungsstation und der Anfragebearbeitungsstation vorhanden, als auch im Ereignisflußdiagramm. Die Art der Synchronisation zwischen den Zustandsübergangsdiagrammen kann mit OMT nicht spezifiziert werden.

Für eine detaillierte Beschreibung der Prototypimplementation und der Modellierung sei auf [17] verwiesen.

4 Vergleichbare Ansätze

Wie im vorherigen Abschnitt zu sehen war, eignet sich die OMT-Notation nicht besonders gut zur Modellierung verteilter Systeme. Mit OMT wird das dynamische Verhalten auf der Basis von Zustandsübergangsdiagrammen für jede Klasse spezifiziert. Die Übergänge von einen Zustand in einen anderen werden üblicherweise durch interne oder externe Ereignisse angestoßen, die als Annotationen an den entsprechenden Pfeilen spezifiziert sind. Externe Ereignisse werden durch Aktionen, die in anderen Klassen spezifiziert sind, ausgelöst. Einen Überblick über den Fluß der externen Ereignisse soll das Ereignisflußdiagramm geben. Die Semantik der Kommunikation ist jedoch nicht ausreichend spezifiziert. Beispielsweise kann eine Unterscheidung zwischen synchroner und asynchroner Kommunikation nur informell als Kommentar erfolgen. Die Booch-Notation [3] bietet z.B. einige Möglichkeiten zur expliziten Spezifikation der Kommunikationsart auf Instanzebene in sogenannten Objektdiagrammen. Deren Semantik ist allerdings ebenfalls nicht formal spezifiziert. In [8] wird vorgeschlagen, zum Prototyping die Ausführung des dynamischen Modells durch die hierarchische Spezifikation der Zustandsübergangsautomaten zu ermöglichen. Den Autoren des vorliegenden Beitrags ist jedoch noch keine Implementierung dieses Ansatzes bekannt. Einen formalen Ansatz zur Spezifikation verteilter Systeme bietet z.B. die Requirements State Machine Language (RSML) [18]. RSML erweitert Zustandsübergangsdiagramme um Schnittstellenbeschreibungen sowie gerichtete Kommunikationskanäle zwischen einzelnen Automaten, um so eine angemessene Modellierung verteilter Systeme zu ermöglichen.

Um die unzureichende Unterstützung von OMT zur Spezifikation von verteilten Systemen zu beheben, wurden auch verschiedene andere Ansätze vorgeschlagen. Beispielsweise wird in [16] OMT mit der PARSE Methode (PARallel Software Engineering) [9] kombiniert. OMT wurde so erweitert, daß Aspekte der Parallelität und Verteilung aus dem Objektmodell auf die PARSE Prozeßgraph-Notation abgebildet werden. Diese Prozeßgraph-Notation erlaubt es, ein paralleles System als eine Hierarchie von Komponenten zu modellieren, die durch Nachrichtenaustausch miteinander kommunizieren. Im wesentlichen wird in diesem Ansatz das dynamische Modell von OMT durch die Prozeßgraph-Notation ersetzt. Im PARSE-Projekt wird auch die Übersetzung von Prozeßgraphen in Petri-Netze untersucht, um so Prototyping

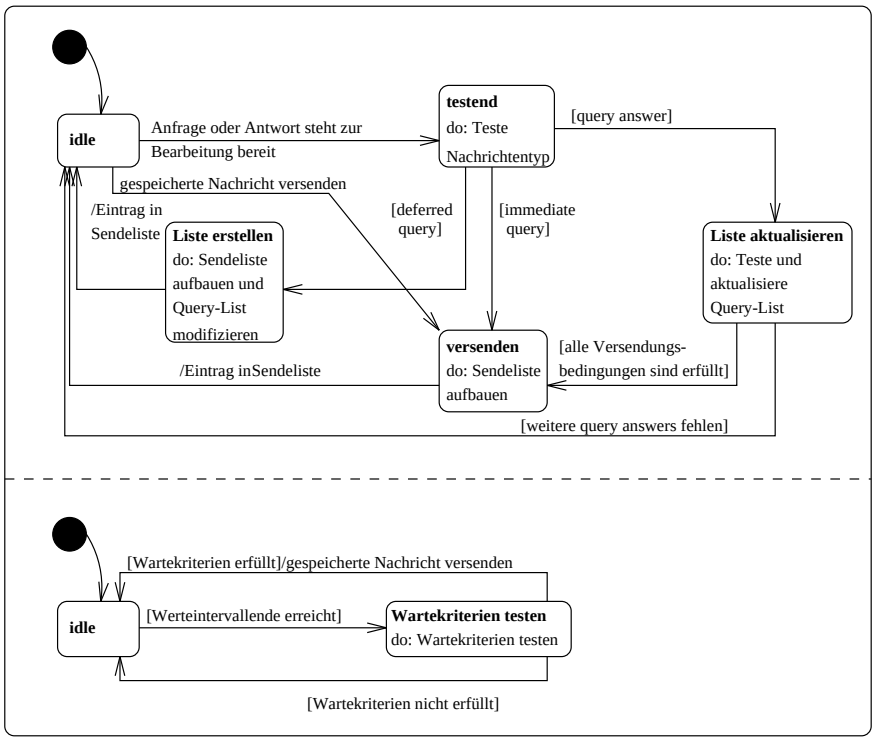


Abbildung 7: Das dynamische Modell der Anfragebearbeitungsstation.

Durch das Ereignis 'Anfrage oder Antwort steht zur Bearbeitung bereit' werden Aktivitäten in der Anfragebearbeitungsstation ausgelöst. Im Zustand **testend** wird getestet, ob die Nachricht eine 'deferred query', eine 'immediate query' oder eine 'query answer' ist. Eine 'immediate query' wird sofort in die Sendeliste eingetragen. Multicasting ist in diesem Fall nicht möglich. Bei 'deferred queries' wird neben dem Aufbau der Sendeliste auch die Query-List modifiziert. In dieser Liste befindet sich eine Liste der Applikationen, deren Antworten noch ausstehen. Die Antworten beider Anfrageklassen gelangen als 'query answers' zur Anfragebearbeitungsstation. Im Zustand **Liste aktualisieren** wird getestet, ob eine gemeinsame 'query answer' verschickt werden kann. Antworten auf 'immediate answers' und Antworten, die einzeln verschickt werden können, werden sofort weitergeleitet. Jedesmal, wenn Nachrichten in der Sendeliste eingetragen werden, wird die Aktion 'Eintrag in Sendeliste' ausgelöst. Die Anfragestation beinhaltet noch einen zweiten parallelen Prozeß. Dieser wird durch das Ereignis 'Warteintervallende erreicht' aktiviert. Hier wäre es vorteilhaft gewesen, wenn OMT Möglichkeiten zur Spezifikation von Realzeit-Eigenschaften bieten würde.

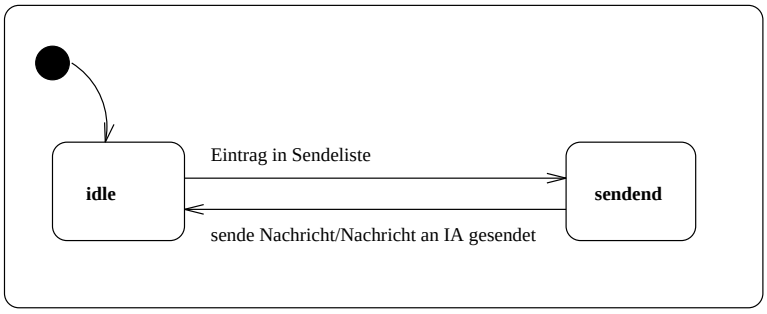


Abbildung 8: Das dynamische Modell der Versendungsstation.

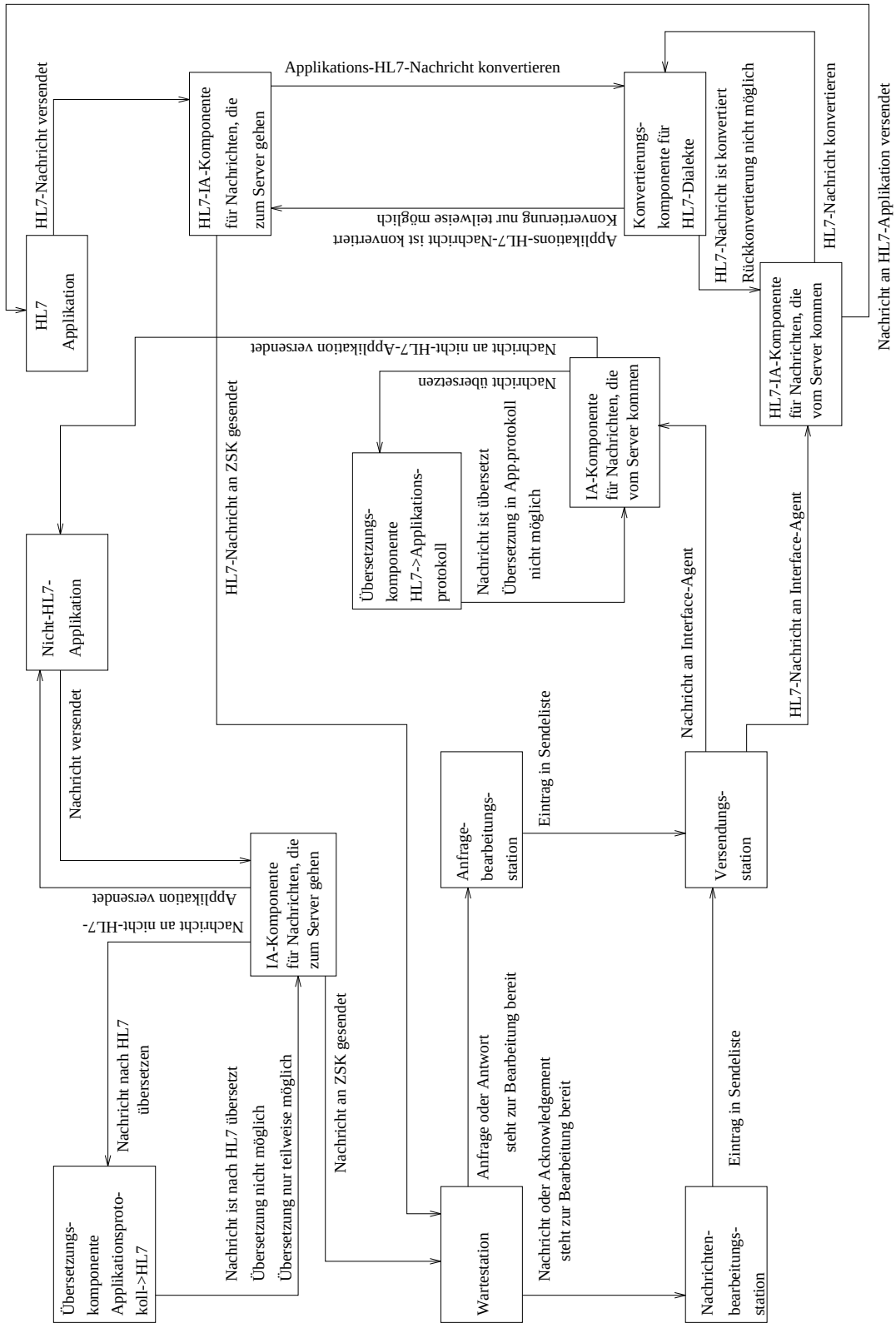


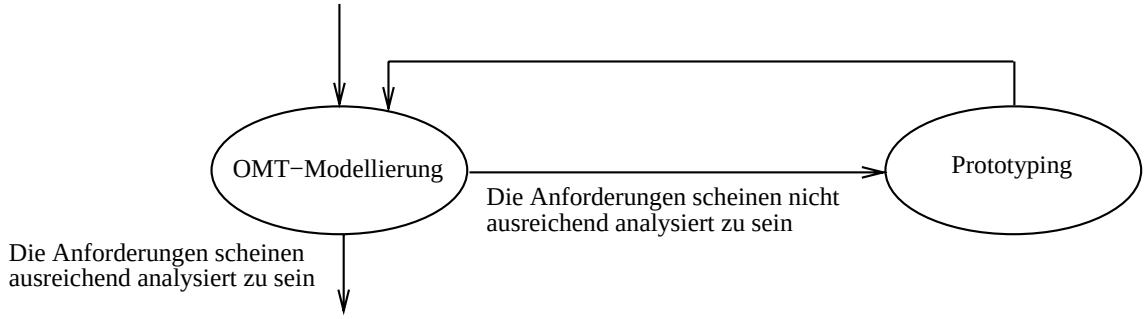
Abbildung 9: Das Ereignisflußdiagramm des HL7-Kommunikations-Servers.

durch Petri-Netz-Simulation zu ermöglichen [9]. Auch die direkte Verknüpfung von objektorientierter Analyse mit Petri-Netzen bietet sich an, da Petri-Netze gerade für die Spezifikation paralleler Systeme besonders geeignet sind [4, 19].

Im vorliegenden Beitrag wird OMT mit einem mengenorientierten Ansatz zum Prototyping paralleler Algorithmen kombiniert. Um durchgängig eine objektorientierte Entwicklungsmethode zu haben, wäre es sicherlich interessant, auch eine parallele objektorientierte Sprache zum Prototyping paralleler Algorithmen einzusetzen. Insbesondere Smalltalk wird häufig für das Prototyping sequentieller Algorithmen eingesetzt [2], so daß es naheliegend erscheint, hier eine parallele Erweiterung von Smalltalk zu verwenden. Beispiele wären Concurrent Smalltalk [13] und Distributed Smalltalk [1]. Solch ein System stand uns leider nicht zur Verfügung. Das grundsätzliche Vorgehen zur Anforderungsanalyse würde sich damit nicht ändern.

5 Abschlußbemerkungen

Dieser Beitrag hat verschiedene Probleme und Lösungsansätze bei der Anforderungsanalyse für ein spezielles verteiltes System aufgezeigt. Aus Platzgründen konnten nur Teilaspekte skizziert werden. Die Kombination der objektorientierten Analyse- und Entwurfsmethode OMT mit einem Prototyping-Ansatz zur Anforderungsanalyse für parallele Systeme wurde 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. Die Anforderungsanalyse stellt sich mit diesem Ansatz wie folgt dar:



Sobald die Anforderungen stabil erscheinen, können die nächsten Phasen im Entwicklungsprozeß beginnen. Die Entscheidung, *ob* die Anforderungen ausreichend analysiert wurden und ob Prototyping durchgeführt werden soll, hängt wesentlich vom bekannten Wissen über den Anwendungsbereich ab. Für die Neu-Implementierung eines bereits eingesetzten Systems, um z.B. nicht-funktionale Eigenschaften wie Effizienz und Zuverlässigkeit zu verbessern, kann auf das Prototyping evtl. ganz verzichtet werden. Für ein zu entwickelndes System, das Aufgaben lösen soll, für die bisher noch keine Software eingesetzt wurde, wird es sinnvoll sein, die Prototyping-Phase mehrfach zu durchlaufen. Das war auch in der hier vorgestellten Anforderungsanalyse an den Krankenhauskommunikations-Server der Fall. Selbstverständlich ist es auch sinnvoll im Sinne eines inkrementellen, iterativen Entwicklungsprozesses aus späteren Phasen wieder in die Anforderungsanalyse zurückzukehren, sobald erkannt wird, daß noch nicht alle Anforderungen ausreichend ermittelt wurden.

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 [20] nur informell behandelt. Es gibt jedoch keine explizite Unterstützung zur Anforderungsanalyse in OMT. Im hier beschriebenen Projekt wurde deshalb diese Vorgehensweise zum Heraussuchen der Substantive nicht eingesetzt.

Embley et al. [5] haben dazu festgestellt, daß die existierenden objektorientierten 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 Implementierung 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 ..."
> [5, Seite 19]

Vorgeschlagen wird in [5] 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 objektorientierte Analyse und für das Prototyping verwenden. OMT wurde im wesentlichen zum Entwurf der u.a. durch das Prototyping gewonnenen Erkenntnisse verwendet.

Literatur

[1] J.K. Bennett. Experience with Distributed Smalltalk. *Software: Practice and Experience*, 20(2):157–180, Februar 1990.

[2] W. Bischofberger und G. Pomberger. *Prototyping oriented software development concepts and tools*. Springer-Verlag, 1992.

[3] G. Booch. *Object-Oriented Analysis and Design with Applications*. Benjamin/Cummings, second edition, 1994.

[4] R. Burkhardt. *Modellierung dynamischer Aspekte mit dem Objekt-Prozeß-Modell*. Dissertation, TU Ilmenau, 1995.

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

[6] R. Fehling. *Hierarchische Petrinetze: Beiträge zur Theorie und formale Basis für zugehörige Werkzeuge*. Dissertation, Universität Dortmund, 1991. (Erschienen im Verlag Dr. Kovač, Hamburg).

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

[8] M. Glinz. Hierarchische Verhaltensbeschreibung in objekt-orientierten Systemmodellen — eine Grundlage für modellbasiertes Prototyping. In H. Züllighoven, W. Altmann und E.-E. Doberkat, Hrsg., *Requirements Engineering '93: Prototyping*, Band 41 von *Berichte des German Chapter of the ACM*, Seiten 175–192. Teubner-Verlag, April 1993.

[9] I. Gorton, J. Gray und I. Jelly. Object based modeling of parallel programs. *IEEE Parallel and Distributed Technology Journal*, 3(2):52–63, Summer 1995.

[10] W.E. Hammond. Health Level 7: A protocol for the interchange of healthcare 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.

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

[12] 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.

[13] W. Horwat, A. Chien und W. Dally. Experience with CST: Programming and Implementation. In *Proc. ACM SIGPLAN Conference on Programming Language Design and Implementation*, Seiten 101–109, Juni 1989.

[14] I. Jacobson, M. Christerson, P. Jonsson und G. Övergaard. *Object-Oriented Software Engineering – A Use Case Driven Approach*. Addison-Wesley, 1992.

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

[16] C. Knowles und P. Collingwood. Parallel software development using an object-oriented modeling technique. *Information and Software Technology*, 36(7):397–404, Juli 1994.

[17] A. Kröber. Modellierung eines konfigurierbaren HL7-Kommunikations-Servers. Diplomarbeit, Universität Dortmund, FB Informatik, Februar 1996.

[18] N. Leveson, H. Hildreth, M.P.E. Heimdahl und J.D. Reese. Requirements specification for process-control systems. *IEEE Transactions on Software Engineering*, 20(9):684–707, September 1994.

[19] D. Moldt. Extensions of normal OOA methods and techniques for distributed real-time systems. In A. Oberweis, Hrsg., *Proc. Fachgruppentreffen '95 Requirements Engineering für Informationssysteme*, Seiten 22–23, (EMISA Forum, Heft 1, 1996).

[20] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy und W. Lorensen. *Object-Oriented Modeling and Design*. Prentice-Hall, 1991.

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

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

[23] 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.