Objektorientierte Modellierung verteilter, technisch eingebetteter Anwendungssysteme am Beispiel eines Laborinformations- und Steuerungssystems

Dirk Weske

Diplomarbeit

Universität Hamburg
Fachbereich Informatik
Arbeitsbereich Softwaretechnik

August 1996

Erstbetreuer:
Prof. Dr. Heinz Züllighoven

Zweitbetreuer:
Prof. Dr. Gustav Pomberger

Diese Diplomarbeit wurde am Fachbereich Informatik der Universität Hamburg zur teilweisen Erfüllung der Anforderungen zur Erlangung des Titels Diplom-Informatiker eingereicht.

Erklärung

Ich versichere hiermit, diese Arbeit selbständig und unter ausschließlicher Zuhilfenahme der in der Arbeit aufgeführten Hilfsmittel erstellt zu haben.

Hamburg, den 31. August 1996

Dirk Weske

Petersdorfstr. 7b
D-21079 Hamburg

Betreuung

Prof. Dr. Heinz Züllighoven (Erstbetreuer)
Prof. Dr. Gustav Pomberger (Zweitbetreuer)

Prof. Dr. Heinz Züllighoven

Arbeitsbereich Softwaretechnik
Fachbereich Informatik
Universität Hamburg
Vogt-Kölln-Str. 30
D-22175 Hamburg
Deutschland

Prof. Dr. Gustav Pomberger

Institut für Wirtschaftsinformatik und Organisationsforschung
Johannes Kepler Universität Linz
Altenberger Str. 69
A-4040 Linz/Auhof
Österreich

Inhaltsverzeichnis

Inhaltsverzeichnis i

1 Einleitung 1

1.1 Motivation und Zielsetzung 1

1.2 Aufbau der Arbeit 3

2 Die Softwareentwicklungsmethode WAM 5

2.1 Der objektorientierte Softwareentwicklungsprozeß nach WAM 5

2.2 Das Leitbild vom Arbeitsplatz für qualifizierte menschliche Tätigkeit 7

2.3 Die Entwurfsmetaphern des WAM-Repertoires 9

2.3.1 Materialien 9

2.3.2 Werkzeuge 10

2.3.3 Automaten 12

2.3.4 Arbeitsumgebung 14

2.4 Interaktive und technisch eingebettete Anwendungssysteme 15

3 Klinische Großlaboratorien als Anwendungsbereich von Laborinformations- und Steuerungssystemen 18

3.1 Einflußfaktoren und Randbedingungen 18

3.2 Arbeitsweise und Organisation 19

4 Das Laborprojekt 20

4.1 Das Laborinformations- und Steuerungssystem XLab 20

4.2 Der automatisierte Transport von Proben 23

4.3 Der Proben-Transporter 26

4.3.1 Lösungsansatz 26

4.3.2 Grenzen 27

4.4 Das Transport-Subsystem 30

4.4.1 Lösungsansatz 30

4.4.2 Grenzen 36

5 Modellierung verteilter, technisch eingebetteter Anwendungssysteme 38

5.1 Analyse des Fehler-Szenarios 38

5.2 Anlagen- und Anwendungsmodell 40

5.3 Sensoren und Aktuatoren 43

5.4 Steuerungsmodell 46

6 Entwurf und Konstruktion verteilter, technisch eingebetteter Anwendungssysteme **54**

6.1 Synchronisierte Zustände des Steuerungsmodells 54

6.2 Aufgabenbezogene Schnitte 57

6.3 Bearbeitungsstationen 61
6.4 Bandmodule 64

7 Zusammenfassung und Ausblick 67

Literaturverzeichnis 70

Anhang: Semantik von Prozeß-Interaktions-Diagrammen 74

1 Einleitung

In den letzten Jahren haben objektorientierte Ansätze zunehmend an Bedeutung für die Softwareentwicklung gewonnen. Ursprünglich vor allem zur Entwicklung interaktiver Anwendungssysteme mit graphischen Benutzungsoberflächen verwendet, finden objektorientierte Konzepte und Methoden mittlerweile auch Eingang in die Modellierung von Anwendungssystemen, die in technisch geprägte Umgebungen einzubetten sind. Die vorliegende Arbeit soll einen softwaretechnischen Beitrag zu dieser Thematik leisten. Sie handelt von der Konzeption und Realisierung eines objektorientierten Laborinformations- und Steuerungssystems und fokussiert dabei vor allem auf die konzeptionellen Unterschiede zwischen der Modellierung interaktiver und der Modellierung technisch eingebetteter Anwendungssysteme. Das erste Kapitel behandelt Motivation und Zielsetzung der Arbeit, die nachfolgend im Überblick präsentiert wird.

1.1 Motivation und Zielsetzung

Die vorliegende Arbeit ist vor dem Hintergrund der Softwareentwicklungsmethode WAM entstanden, die am Arbeitsbereich Softwaretechnik des Fachbereichs Informatik der Universität Hamburg in Forschung und Lehre vertreten wird. Zudem ist sie geprägt durch die Tätigkeit des Autors als Leiter eines Projektes zur Entwicklung eines Laborinformations- und Steuerungssystems (LIS) für den Betrieb klinischer Großlaboratorien. Innerhalb dieses Projektes, das im weiteren kurz als *Laborprojekt* bezeichnet werden soll, wurde nach der Methode WAM vorgegangen.

WAM ist eine Methode zur objektorientierten Entwicklung von Software. Die Bezeichnung WAM steht dabei für Werkzeug, Automat und Material, den aus konstruktiver Sicht wichtigsten Komponenten eines nach WAM entworfenen Softwaresystems. WAM vertritt als Methode einen menschenzentrierten und anwendungsnahen Ansatz, der neben der Verwendung objektorientierter Techniken vor allem durch evolutionäres und partizipatives Prototyping und die Verwendung allgemeinverständlicher Metaphern geprägt ist (vgl. [BZ90, BCS92, BKK+92, BZ92]). Grundlage für diese Herangehensweise ist, daß Softwareentwicklung von allen Beteiligten als wechselseitiger Kommunikations- und Lernprozeß anerkannt wird, in dem Herstellung und Einsatz des Produktes Software miteinander verschränkt sind. Diese Sichtweise wird in [Flo87] als Komplementarität von Prozeß und Produkt bezeichnet und erweitert damit die traditionell produktorientierte Auffassung von Softwareentwicklung. Weiter liegt WAM ein bewußt eingenommenes Leitbild zugrunde, das allen an der Softwareentwicklung Beteiligten eine Orientierungshilfe bietet, um Prozeß und Produkt wahrzunehmen, zu verstehen und zu gestalten. Dieses Leitbild ist der *Arbeitsplatz für qualifizierte menschliche Tätigkeit* (vgl. [GZ92]).

Die Methode WAM wurde im Verlauf der letzten Jahre vor allem für die Entwicklung *interaktiver Anwendungssysteme* im Bankenbereich eingesetzt (vgl. [BGZ95a, BGZ95b]). Im Zusammenhang dieser Arbeit ist das wichtigste Kennzeichen interaktiver Anwendungssysteme, daß alle Aktivitäten innerhalb der Software stets als Reaktion auf Handlungen ihres Benutzers erfolgen, niemals jedoch das Anwendungssystem selbst seinem Benutzer den "Arbeitstakt" vorgibt.

Der Anwendungsbereich des Laborprojektes unterscheidet sich von den ursprünglichen Anwendungsbereichen der Methode WAM in vielerlei Hinsicht. So handelt es sich bei dem im Rahmen des Laborprojektes entwickelten LIS deutlich nicht um ein interaktives Anwendungssystem, in dem ausschließlich der Benutzer der jeweiligen Arbeitsumgebung Aktivitäten innerhalb der Software veranlaßt. Statt dessen sind es gerade die unabhängig vom Benutzer stattfindenden Ereignisse innerhalb der technischen Umgebung eines LIS, die einen wichtigen Teil der Aktivitäten innerhalb der Software auslösen. Aus diesem Grunde läßt sich das in dieser Arbeit thematisierte LIS in die Kategorie der *technisch eingebetteten Anwendungssysteme* einordnen.

Anliegen der vorliegenden Arbeit ist es nun, die konzeptionellen Unterschiede zwischen interaktiven und technisch eingebetteten Anwendungssystemen offenzulegen und dabei, aus einer softwaretechnischen Perspektive, insbesondere auf den Aspekt der Modellbildung zu fokussieren. Damit soll ein Beitrag zur Erweiterung des Anwendungsbereichs von WAM geleistet werden.

Als Einstieg sollen zunächst die nachfolgend plakativ formulierten Thesen zur Modellierung technisch eingebetteter Anwendungssysteme dienen. Aufgabe der weiteren Kapitel dieser Arbeit ist es, diese Thesen auszuarbeiten, zu motivieren und argumentativ zu belegen.

1. Im Vergleich zur Modellierung interaktiver Anwendungssysteme erfordern die in der Umgebung eines technisch eingebetteten Anwendungssystems stattfindenden Arbeitsprozesse und Ereignisse die Bildung zusätzlicher Modelle bei dessen Entwurf und Konstruktion. Dabei handelt es sich um Anlagen- und Steuerungsmodell, die das eigentliche Anwendungsmodell ergänzen. Das Anlagenmodell beschreibt die einbettende technische Umgebung des Anwendungssystems. Das Steuerungsmodell wird in Software implementiert und dient zur Steuerung der zu betreibenden Anlage. Dazu bildet es u.a. die zur Verfügung stehenden Schnittstellen zwischen Hard- und Software ab.
2. Zur Laufzeit des technisch eingebetteten Anwendungssystems müssen Anlagenmodell und Steuerungsmodell miteinander synchronisiert werden, um dem Anwendungsmodell ein synchronisiertes Bild des Anlagenmodells präsentieren zu können.
3. Der Entwurf der einbettenden Hardware muß Probleme der Synchronizität von Anlagen- und Steuerungsmodell berücksichtigen, darf also nicht unabhängig vom (Software-) Entwurf des Steuerungs- bzw. des Anwendungsmodells erfolgen.
4. Sogenannte Schnitte durch die drei Modelle müssen softwaretechnisch durch geeignete Entwurfskomponenten umgesetzt werden. Die Entwurfsmetaphern des WAM-Repertoires stellen dafür bislang keine geeigneten Abstraktionen zur Verfügung. Das Repertoire muß daher entsprechend erweitert werden.

Die Argumentation der vorliegenden Arbeit ist eng an dem bereits erwähnten Laborprojekt ausgerichtet. Sie wird daher von einer einführenden Beschreibung der innerhalb des Projektes verwendeten Softwareentwicklungsmethode WAM, sowie einem Überblick über den Anwendungsbereich der klinischen Großlaboratorien eingeleitet.

Ausgehend von der Architektur des im Rahmen des Laborprojektes entwickelten LIS, wird ein konkretes Teilproblem innerhalb des Projektes herausgegriffen und auf Ebene der beteiligten Hardwarekomponenten beschrieben. Anhand dieses Teilproblems sollen die Thesen der Arbeit festgemacht und im weiteren diskutiert werden.

Dazu werden mehrere Lösungsansätze herausgearbeitet und ihre jeweiligen Grenzen aufgezeigt. Diese bieten Ansatzpunkte, um vom konkreten Problem zu abstrahieren und die konzeptionellen Unterschiede zwischen der Modellierung interaktiver und der Modellierung technisch eingebetteter Anwendungssysteme herauszuarbeiten.

Die Abstraktion der konzeptionellen Unterschiede führt direkt zu den ersten drei Thesen dieser Arbeit. Diese fordern die Bildung zusätzlicher Modelle bei Entwurf und Konstruktion technisch eingebetteter Anwendungssysteme und erfahren nun ihre argumentative Bestätigung.

Auch die in der vierten These vertretene Forderung nach einer Erweiterung des von der Methode WAM zur Verfügung gestellten Repertoires an Entwurfsmetaphern läßt sich mit der Bestätigung der ersten drei Thesen motivieren. Die neue Entwurfsmetapher wird auf konstruktiver Ebene umgesetzt und ihre Anwendbarkeit exemplarisch verdeutlicht.

1.2 Aufbau der Arbeit

In **Kapitel 2** soll das methodische Umfeld der vorliegenden Arbeit mit einem Überblick über die objektorientierte Softwareentwicklungsmethode WAM verdeutlicht werden. Dazu wird zunächst die von WAM eingenommene Perspektive auf den objektorientierten Softwareentwicklungsprozeß beschrieben und das WAM zugrundeliegende Leitbild vom Arbeitsplatz für qualifizierte menschliche Tätigkeit vorgestellt. Auf dieser Grundlage wird es möglich, die bislang eher informale Unterscheidung zwischen interaktiven und technisch eingebetteten Anwendungssystemen zu präzisieren. Das Kapitel wird durch einen Überblick über die verwendeten Entwurfsmetaphern abgeschlossen, die zur konstruktiven Umsetzung des Leitbildes dienen. Diese bilden die Grundlage für eine neue Entwurfsmetapher, die sich im weiteren Verlauf dieser Arbeit als notwendig erweisen wird.

Kapitel 3 beinhaltet eine einführende Beschreibung klinischer Großlaboratorien. Diese bilden den Anwendungsbereich des dieser Arbeit zugrundeliegenden Projektes zur Entwicklung eines LIS. Dort herrschende Einflußfaktoren und Randbedingungen werden aufgezeigt und durch einen Überblick über Arbeitsweise und Organisation klinischer Großlaboratorien ergänzt.

In **Kapitel 4** wird die Architektur des im Rahmen des Laborprojektes entwickelten LIS im Überblick vorgestellt. Mit dem automatisierten Transport von Proben zum Arbeitsplatz oder Analyseautomaten wird ein konkretes Teilproblem innerhalb des Projektes herausgegriffen und ein erster Lösungsansatz vorgestellt. Dessen technische und konzeptionelle Grenzen führen zu einem zweiten, verteilten Lösungsansatz, der zwar die technischen Probleme des ersten Ansatzes beseitigt, auf konzeptioneller Ebene aber nach wie vor Grenzen aufweist.

Diese konzeptionellen Grenzen werden in **Kapitel 5** analysiert und auf die im vorangegangenen Abschnitt aufgestellte These zurückgeführt, daß mit Anlagen- und Steuerungsmodell zusätzliche Modelle für den Entwurf und die Konstruktion technisch eingebetteter Anwendungssysteme notwendig werden. Die zusätzlich benötigten Modelle werden motiviert und anhand des konkreten Teilproblems verdeutlicht.

Kapitel 6 beschäftigt sich mit dem Entwurf und der Konstruktion verteilter, technisch eingebetteter Anwendungssysteme. Dazu wird eine softwaretechnische Umsetzung der im vorangegangenen Kapitel motivierten Modelle vorgestellt. In diesem Zusammenhang erweist

es sich als notwendig, das von WAM zur Verfügung gestellte Repertoire an Entwurfsmetaphern um die sogenannte Bearbeitungsstation zu erweitern.

Kapitel 7 faßt die in dieser Arbeit erzielten Ergebnisse zusammen und zeigt offen gebliebene Punkte sowie Themen für weitere Arbeiten auf.

2 Die Softwareentwicklungsmethode WAM

Das vorliegende Kapitel beinhaltet eine einführende Beschreibung der objektorientierten Softwareentwicklungsmethode WAM. Diese wurde eingesetzt, um das im Rahmen des Laborprojektes entwickelte LIS objektorientiert zu realisieren.

Zusammen mit dem nachfolgenden Kapitel über klinische Großlaboratorien dient dieses Kapitel als Grundlage, um die konzeptionellen Unterschiede zwischen interaktiven und technisch eingebetteten Anwendungssystemen anhand eines konkreten Anwendungsbereichs diskutieren zu können.

2.1 Der objektorientierte Softwareentwicklungsprozeß nach WAM

Objektorientierte Ansätze haben in den letzten Jahren zunächst im wissenschaftlichen, später auch im kommerziellen Bereich zunehmenden Einfluß auf die moderne Softwaretechnik gewonnen. Basierend auf den schon in den sechziger Jahren mit der Sprache Simula gewonnenen Erfahrungen (vgl. [ND78]), erheben sie den Anspruch, der im traditionellen Software-Engineering vorgenommenen Trennung zwischen fachlichem und technischem Entwurf eines Softwaresystems und dem dadurch verursachten Bruch zwischen Methoden, Modellen und Darstellungsmitteln beider Ebenen durch eine einheitliche Modellbildung bei Analyse (**O**bject **O**riented **A**nalysis), Entwurf (**O**bject **O**riented **D**esign) und Implementierung (**O**bject **O**riented **P**rogramming) entgegenzutreten.

Wie bei fast allen sich rasant entwickelnden Forschungs- und Anwendungsfeldern der Informatik entstand jedoch zunächst einmal unter dem Oberbegriff "Objektorientierung" eine schon fast verwirrende Methodenvielfalt (vgl. etwa [Mey88, SM88, WWW90, Boo91, CY91, RBP+91, Jac92]). Während aber die konzeptionelle Diskussion über objektorientierte Programmierung mit der Arbeit von Wegner (vgl. [Weg90]) zu einem tragfähigen Kompromiß geführt hat und auch die verwendeten Methoden, zumindest auf der Ebene von Notationen und Entwurfsdokumenten wie *Object Models* und *Use Cases*, u.a. durch die Integration mehrerer Ansätze in den neueren Arbeiten von Rumbaugh (vgl. [Rum95]) konvergieren, sind die Ansichten über Projektmodelle, wie die objektorientierte Sichtweise auf den Entwicklungsprozeß übertragen werden soll, bis heute kontrovers geblieben. Aus diesem Grunde soll im folgenden in Anlehnung an [Zül92] und [KGZ+94] die von WAM eingenommene Perspektive auf den objektorientierten Softwareentwicklungsprozeß dargestellt werden.

Viele objektorientierte Methoden verkörpern die Auffassung, daß die formale Modellierung der "realen Dinge" des Anwendungsbereichs und deren Überführung in Objekte eines lauffähigen Anwendungssystems dem Aspekt der Modellbildung bereits hinreichend Sorge tragen (vgl. etwa [Mey88, Tay93]). Diese Auffassung wird durch die in [Abb93] vorgeschlagene Methode, Objekte aus den in einer Beschreibung des Anwendungsbereiches vorkommenden Substantiven herzuleiten, auf die Spitze getrieben.

Im Gegensatz dazu fokussiert WAM bei Analyse und Modellbildung auf die selbstverständlichen Dinge des alltäglichen Umgangs am Arbeitsplatz (vgl. [Zül92], S. 150):

> "Unsere *Grundidee der objektorientierten Modellierung* ist, von den selbstverständlichen Dingen des alltäglichen Umgangs (dem »Zeug«) auszugehen. Damit sehen und verstehen wir die Dinge in ihrer Zweckmäßigkeit für uns und betten sie in die Bezüge einer menschlichen Arbeits- und Lebenswelt ein. Die so erschlossenen Dinge wollen wir in ein formales Modell überführen und sie dann als Objekte im laufenden Anwendungssystem zur Verfügung stellen."

Die *Analyse* des alltäglichen Umgangs mit den am Arbeitsplatz vorgefundenen Arbeitsgegenständen und Arbeitsmitteln erlaubt es, diese anhand von gemeinsamen Umgangsformen zu klassifizieren und Bezüge zwischen Arbeitsgegenständen zur Strukturierung dieser Umgangsformen zu nutzen. Dabei werden die dahinterstehenden relevanten Begriffe der (Benutzer-) Fachsprache rekonstruiert. Umgangsformen und Begriffe charakterisieren also die für die Anwendung relevanten Gegenstände, die als Objekte des Anwendungssystems modelliert werden sollen. Als Umgangsformen werden dabei sowohl die Informationen, die an den Gegenständen unmittelbar ablesbar sind, als auch die Veränderungen, die an ihnen vorgenommen werden können, ohne daß sie zerstört oder in andersartige Gegenstände umgewandelt werden, betrachtet.

Diese Modellbildung führt nun im nächsten Schritt, dem *fachlichen Klassenentwurf*, zur Anordnung von Begriffen in Form von Begriffsgerüsten und Klassifikationshierarchien. Grundlage dafür sind Gemeinsamkeiten und Verwandtschaftsbeziehungen, die sich aus ähnlichen Strukturen und Umgangsformen ergeben und sich im Modell als gemeinsame Oberklassen niederschlagen.

Auf Basis des fachlichen Klassenentwurfs kann nun eines der Versprechen der Objektorientierung eingehalten werden. Die Verwendung des softwaretechnischen Repertoires der objektorientierten Programmierung (Vererbung, Abstrakte Datentypen, Zusicherungen) erlaubt es, die Klassifikationshierarchie des fachlichen Entwurfs direkt in einen *technischen Klassenentwurf* zu überführen. Die dazu verwendeten Notationen scheinen, wie schon anfangs erwähnt, in die Richtung von [Rum95] zu konvergieren.

Der technische Klassenentwurf kann nun mit Hilfe einer konkreten objektorientierten Programmiersprache, etwa Eiffel [Mey92] oder C++ [Str91], durch die Implementation von nach außen sichtbaren Schnittstellenoperationen und die Festlegung von nur intern bekannten Speicherstrukturen und Hilfsoperationen in ein lauffähiges Anwendungssystem umgesetzt werden. Auch wenn die Erfahrungen der Praxis zeigen, daß der versprochene bruchlose Übergang von objektorientiertem Entwurf zu objektorientierter Programmierung - wenn möglich sogar automatisiert (vgl. etwa die Arbeit von [SM92] zum Thema *Recursive Design*) - wahrscheinlich nicht vollständig zu realisieren ist, so liegt die *Implementation von Klassen* doch deutlich näher an diesem Wunschziel als etwa die Überführung eines daten- und funktionsorientierten Entwurfs in eine modulare Architektur.

In diesem Abschnitt wurde die von WAM eingenommene Perspektive auf objektorientierte Softwareentwicklung beschrieben. Dabei wurde aufgezeigt, wie aus der fachlichen Beschreibung der relevanten Gegenstände und Begriffe eines Anwendungsbereichs eine

bruchlose Überführung in ein technisches Modell möglich ist. Dagegen ist offen geblieben, welche Gegenstände und Begriffe des Anwendungsbereichs als relevant zu betrachten sind.

2.2 Das Leitbild vom Arbeitsplatz für qualifizierte menschliche Tätigkeit

Die am Ende des vorangegangenen Abschnitts aufgeworfene Frage nach den relevanten Gegenständen und Begriffen des Anwendungsbereichs ist eng verknüpft mit der grundsätzlichen Sichtweise, wie das angestrebte System zu gestalten und in die Arbeitszusammenhänge der zukünftigen Benutzer einzubetten ist. Eine Klärung dieser grundlegenden Sichtweise kann orientierend bei der Beantwortung der eingangs aufgeworfenen Frage nach den relevanten Gegenständen und Begriffen des Anwendungsbereichs wirken. Zu diesem Zwecke definiert Gryczan den Begriff des *Leitbildes* wie folgt (vgl. [Gry96], S. 99):

> "Ein Leitbild ist eine benannte, mit Absicht eingenommene grundsätzliche Sichtweise. Es ist eine Orientierung, die von Menschen angenommen wird, anhand der sie einen Ausschnitt von Realität wahrnehmen, verstehen und gestalten. Ein Leitbild bestimmt, wie Entwickler und Benutzer bei der Systementwicklung wechselseitig miteinander umgehen und beschreibt die Gestaltungsziele bei der Softwareentwicklung."

Maaß weist in diesem Zusammenhang darauf hin, daß Leitbilder, im Gegensatz zu den sonst im Rahmen der Softwareentwicklung anzutreffenden Rollenkonzepten, eher Gemeinsamkeiten in den Sichtweisen von Entwicklern und Benutzern betonen, als daß sie auf die Verschiedenartigkeit der Perspektiven, zum Beispiel in den Rollen von Entwicklern und Benutzern, abstellen (vgl. [Maa94]). Während Maaß aber die von ihr analytisch aufgearbeiteten Leitbilder als nicht strikt handlungsleitend einschätzt, fokussiert WAM gerade auf die orientierende Wirkung von Leitbildern.

Dazu wird der Systementwicklung nach WAM das *Leitbild vom Arbeitsplatz für qualifizierte menschliche Tätigkeit* zugrundegelegt. Die zukünftigen Benutzer werden dabei als Experten ihres Anwendungsbereichs verstanden, die über das notwendige Wissen und ausreichende Erfahrung verfügen, um ihre Arbeit selbständig und qualifiziert zu erledigen. Deren Arbeit soll nun durch geeignete Software-Artefakte unterstützt werden. Für diese Sichtweise, die aufgabenangemessene Unterstützung menschlicher Tätigkeit durch flexibel verwendbare Software-Werkzeuge, wird in [Gry96] der Begriff *Unterstützungssicht* geprägt. Kennzeichnend für unter dieser Sichtweise entwickelte Systeme ist, daß sie dem Benutzer Hilfsmittel an die Hand geben, deren Einsatz dieser aber zu jedem Zeitpunkt selbst bestimmt. Auf technischer Ebene führt diese Sichtweise zur Konstruktion von sogenannten "interaktiven Anwendungssystemen" mit graphischen Benutzungsoberflächen, bei denen alle Aktivitäten des Systems durch Aktionen des Benutzers ausgelöst werden.

Dieser unterstützenden Sichtweise läßt sich der Begriff der *ablaufsteuernden Sichtweise* kontrastiv gegenüberstellen. Für ablaufsteuernde Systeme ist es kennzeichnend, menschliches Arbeitshandeln zu regeln und zu steuern.

Die hierzu verwendeten Pläne sind im Idealfall durch Algorithmen auf Maschinen ausführbar und sollen, wo möglich, menschliches Arbeitshandeln ganz ersetzen (vgl. [Maa94], S. 4):

> "Der mit der Maschine arbeitende Mensch wird zum potentiellen Störfaktor, der möglichst weitgehend zu eliminieren ist. Soweit seine Mitarbeit nötig ist, wird er zum Maschinenbediener degradiert, der die Vorgänge, die er anstößt, nicht durchschaut. Die Entwickler sehen sich als Ingenieure und wirken gleichzeitig als Maschineneinrichter für die Bediener, die an der vorgefundenen Bedienungsschnittstelle nichts verändern können und sollen."

Unter der ablaufsteuernden Sichtweise entwickelte Systeme zeichnen sich i.d.R. durch eine im voraus von den Entwicklern festgelegte Abfolge von Menüs und Bildschirmmasken aus, die es dem Benutzer nicht erlauben, flexibel auf die jeweilige Arbeitssituation zu reagieren. Diese mangelnde Flexibilität führt, obwohl sie in bestimmten Arbeitszusammenhängen durchaus erwünscht sein mag, häufig zur erzwungenen Anpassung der betrieblichen Arbeitsabläufe an das Softwaresystem und damit fast zwangsläufig zu Akzeptanzproblemen und insgesamt negativen Auswirkungen auf die Qualität der Arbeit.

Zusammenfassend läßt sich also sagen, daß das WAM zugrundeliegende Leitbild vom Arbeitsplatz für qualifizierte menschliche Tätigkeit auf der Annahme basiert, daß die zukünftigen Benutzer in ihrer Tätigkeit ausreichend qualifiziert sind, um angebotene Freiheitsgrade sinnvoll und verantwortlich zu nutzen. Sie sollen in ihrer Arbeitstätigkeit durch flexibel verwendbare Software-Werkzeuge unterstützt werden, die es ihnen erlauben, aufgabenangemessen auf die Anforderungen der konkreten Arbeitssituation zu reagieren.

Festgestellt werden muß an dieser Stelle aber auch, daß sich das Leitbild vom Arbeitsplatz für qualifizierte menschliche Tätigkeit deutlich am Anwendungsbereich interaktiver Anwendungssysteme orientiert. Im noch folgenden Abschnitt 2.4 über die Unterscheidung zwischen interaktiven und technisch eingebetteten Anwendungssystemen wird in diesem Zusammenhang aufgezeigt, daß sich vor allem die Arbeitsplätze in technisch eingebetteten Anwendungssystemen qualitativ von denjenigen interaktiver Anwendungssysteme unterscheiden. Die sich unmittelbar ergebende Frage, ob das WAM zugrundeliegende Leitbild auch zu Arbeitsplätzen in technisch eingebetteten Anwendungssystemen paßt, ist allerdings noch ungeklärt und bedarf der weiteren Diskussion.

Im nächsten Abschnitt soll nun gezeigt werden, wie die Entwurfsmetaphern des WAM-Repertoires helfen, die Annahmen des Leitbildes konstruktiv in ein zukünftiges Softwaresystem umzusetzen.

2.3 Die Entwurfsmetaphern des WAM-Repertoires

Die Komplexität und der abstrakte Charakter von Software erschweren die Diskussion über Software. Dieses gilt in verstärktem Maße vor dem unterschiedlichen Erfahrungshorizont und den unterschiedlichen Rollen, die die an der Diskussion beteiligten Personen im Softwareentwicklungsprozeß einnehmen. Eine häufig in der Softwareentwicklung anzutreffende Methode, diese "Verständnislücke" zu schließen, ist der Gebrauch von Metaphern, also der Versuch, weniger bekannte Gegenstandsbereiche durch Anspielung auf vertrautere Bereiche zu beschreiben (vgl. [MO92], S.234):

> "Metaphors, carefully chosen and used to design the interface, can also serve as a learning vehicle in situations of computer use. They make systems self-explanatory or may be used explicitly in user training to explain new systems or additional aspects of systems already in use. The idea is to exploit users' prior knowledge and increase their initial familiarity with the system by presenting its handling in familiar terms. Typical examples are the well-known *folders* and *trashcans* in desktop systems or the idea of providing *menus* to choose actions or objects from."

Neben dieser, eher impliziten, Funktion von Metaphern im Sprachgebrauch der am Softwareentwicklungsprozeß beteiligten Personen, können Metaphern auch explizit dazu benutzt werden, die Annahmen des Leitbildes konstruktiv in das zukünftige Softwaresystem umzusetzen. Diese *Entwurfsmetaphern* werden dabei passend zu dem zugrundeliegenden Leitbild gewählt und begleiten ein Softwaresystem auf diese Weise von der Analyse über den Entwurf bis hin zu seiner Verwendung.

Die Softwareentwicklungsmethode WAM verwendet mittlerweile ein ganzes Repertoire von Entwurfsmetaphern. Sie alle basieren auf der zentralen (und namensgebenden) *Metapher vom Werkzeug und Material*. Die Metapher vom Werkzeug und Material schafft die Verbindung von den bisher am Arbeitsplatz vorhandenen vertrauten Dingen zu den neuen Elementen eines Anwendungssystems. Die Arbeitsmittel des täglichen Umgangs werden dabei als Werkzeuge, die Arbeitsgegenstände als Materialien modelliert. Auf softwaretechnischer Ebene finden diese ihre Entsprechung in Werkzeug- und Materialklassen. Thema der nächsten Abschnitte soll es sein, die zentralen Entwurfsmetaphern *Werkzeug* und *Material*, sowie die ergänzenden Metaphern *Automat* und *Arbeitsumgebung* zu motivieren.

2.3.1 Materialien

Bei der Beschreibung des objektorientierten Softwareentwicklungsprozesses nach WAM konnte bereits in Abschnitt 2.1 gezeigt werden, daß eine Analyse des alltäglichen Umgangs mit den am Arbeitsplatz vorgefundenen Arbeitsgegenständen es erlaubt, diese anhand von gemeinsamen Umgangsformen zu klassifizieren und als Objekte eines laufenden Anwendungssystems zur Verfügung zu stellen.

WAM beschreibt solchermaßen gefundene Arbeitsgegenstände metaphorisch als *Materialien*. Dabei umfaßt die Metapher vom Material nicht nur die konkreten Arbeitsgegenstände, wie etwa Formulare oder Ordner bei der Büroarbeit, sondern auch abstrakte Gegenstände, wie

zum Beispiel Konditionen oder Zinssätze, mit denen die Benutzer alltäglich umgehen. Kennzeichnend für Materialien sind ihre fachlichen Umgangsformen, d.h. diejenigen Informationen, die an den Materialien unmittelbar ablesbar sind, sowie die Veränderungen, die an ihnen vorgenommen werden können, ohne daß diese zerstört oder in andersartige Materialien umgewandelt werden.

Objektorientiert werden Materialien als *Materialklassen* modelliert. Diese werden anhand ihrer Umgangsformen in Begriffshierarchien angeordnet und definieren Operationen zum Sondieren und Verändern des zugrundeliegenden fachlichen Materials. Materialklassen lassen sich beim Übergang vom fachlichen zum technischen Klassenentwurf relativ nahtlos in ein technisches Klassenmodell überführen.

Ein gutes Beispiel für eine Materialklasse innerhalb eines nach WAM entworfenen LIS ist die sogenannte *Probe*. Proben sind der zentrale Arbeitsgegenstand innerhalb eines klinischen Laboratoriums und bestehen i.d.R. aus Blut oder anderen Körperflüssigkeiten. Häufig vorkommende Umgangsformen mit Proben sind beispielsweise, den Füllstand zu überprüfen oder eine bestimmte Menge an Flüssigkeit zu entnehmen. Diese fachlichen Umgangsformen können nahtlos durch sondierende bzw. verändernde Operation an der Klassenschnittstelle nachgebildet werden.

Entscheidendes Kennzeichen von Materialklassen ist, daß diese niemals eigenständige Ein- oder Ausgabeoperationen durchführen. Sie bedürfen dazu immer des Anstoßes von außen, beispielsweise durch interaktive Werkzeuge. Dieses Verhalten reflektiert auf technischer Ebene die aus der täglichen Arbeit gewonnene Erfahrung, daß Arbeitsgegenstände niemals von sich aus aktiv werden.

2.3.2 Werkzeuge

Im vorangegangenen Abschnitt ist deutlich geworden, daß Arbeitsgegenstände, ganz im Einklang mit der verwendeten Metapher vom Material, niemals von sich aus aktiv werden. Statt dessen werden sie von Arbeitsmitteln benutzt, um Arbeitstätigkeiten an ihnen zu verrichten. Dieser Zusammenhang wird durch die Metapher vom *Werkzeug* ausgedrückt, das der Benutzer einsetzt, um Materialien zu bearbeiten.

Analog zu Materialien werden Werkzeuge softwaretechnisch als *Werkzeugklassen* umgesetzt. Während Materialklassen Arbeitsgegenstände abbilden, modellieren Werkzeugklassen Arbeitsmittel. Die Umgangsformen einer Anwendung werden dem Benutzer als Satz von Kommandos zur Verfügung gestellt mit denen Materialien sondiert oder verändert werden können.

Dabei präsentieren Werkzeuge ihrem Benutzer eine konsistente Sicht auf das gerade bearbeitete Material, d.h. Veränderungen an diesem werden dem Benutzer über die Benutzungsschnittstelle des Werkzeuges unmittelbar transparent gemacht. Da die Auswahl und Reihenfolge von Werkzeugkommandos ausschließlich in der Hand des Benutzers liegt, verhalten sich Werkzeuge stets reaktiv. Damit stehen sie, bezogen auf die Ausführungen des Abschnittes 2.2, im Einklang mit dem zugrundegelegten Leitbild vom Arbeitsplatz für qualifizierte menschliche Tätigkeit.

Ein weiterer Aspekt der Metapher von Werkzeug und Material muß noch beleuchtet werden. So kann es vom jeweiligen Verwendungszusammenhang abhängen, ob Dinge begrifflich als Werkzeuge oder als Materialien einzuordnen sind (vgl. [KGZ+94], S. 31):

> "Ein Ding wird vielmehr dadurch zum Material, daß wir es mit einem anderen Ding, das dadurch zum Werkzeug wird, bearbeiten, um ein Arbeitsergebnis zu erzielen. [...] Zwar verwenden wir einen Hammer meist als Werkzeug, aber wenn wir den Hammerstiel erneuern müssen, dann wird der Hammer zu Material, das wir mit anderen Werkzeugen bearbeiten."

Ferner läßt sich anhand der Metapher von Werkzeug und Material fachlich motivieren, daß ein Material ebenso von verschiedenen Werkzeugen bearbeitet werden kann, wie ein Werkzeug für unterschiedliche Materialien verwendbar ist. So ist leicht vorstellbar, daß ein Werkzeug *Auflister* nicht nur für das Material *Formular* geeignet ist, sondern ebenso für das Material *Vertrag*. Umgekehrt kann beispielsweise ein *Vertrag* sowohl von einem *Auflister* als auch von einem *Vertragseditor* bearbeitet werden.

Vor dem Hintergrund dieser zyklischen Verwendbarkeit der Begriffe Werkzeug und Material stellt sich damit die Frage, wie das "Zusammenpassen" von Werkzeugen und Materialien, eine in der fachlichen Realität meist durch Erfahrung beantwortbare Fragestellung, in das objektorientierte Modell des zukünftigen Anwendungssystems aufgenommen werden soll.

Im technischen Klassenentwurf nach WAM entworfener Softwaresysteme wird dieses Zusammenpassen durch sogenannte *Aspektklassen* modelliert. Aspektklassen haben dabei die Funktion von Schnittstellen, die den Verwendungszusammenhang von Materialien spezifizieren. Sie schränken dazu die eigentliche Materialschnittstelle auf die für einen bestimmten Verwendungszusammenhang relevanten Methoden ein. Greift nun jedes Werkzeug, statt direkt die Methoden der von ihm zu bearbeitenden Materialien aufzurufen, grundsätzlich über eine zu ihm passende Aspekt-Schnittstelle auf die Methoden der Materialien zu, so kann das Zusammenpassen von Werkzeugen und Materialien explizit modelliert werden.

Auf softwaretechnischer Ebene werden Aspektklassen unter Verwendung des Repertoires der objektorientierten Programmierung meist durch abstrakte Klassen implementiert, die durch Deklaration von sogenannten aufgeschobenen (*deferred*) Methoden ein abstraktes Protokoll spezifizieren, daß durch die von der jeweiligen Aspektklasse erbende Materialklasse zu implementieren ist[1].

Der Einsatz von Aspektklassen als Entwurfsmuster zur Kopplung von Werkzeugen und Materialien bewirkt zusätzlich eine aus softwaretechnischer Sicht wünschenswerte Unabhängigkeit bei der Implementation von Werkzeug- und Materialklassen, da Werkzeuge

---

[1]Die Verwendung des Vererbungsmechanismus zur Realisierung von Aspektklassen ist, u.a. wegen der dort auftretenden Mehrfachvererbung, nicht unumstritten. Als alternative Implementationstechnik wurden bereits mehrfach sogenannte *Wrapper* vorgeschlagen. Die in diesem Zusammenhang geführte Diskussion ergab eine Differenzierung von Aspektklassen in *fachliche Aspektklassen* zur Einschränkung der Materialschnittstelle unter Verwendung des Vererbungsmechanismus und *generische Aspektklassen* zur Abstraktion gemeinsamer Materialeigenschaften unter Verwendung von Wrapper-Techniken. Ein Beispiel für letzteres wäre die Abstraktion der durchaus mehreren Materialien inhärenten Eigenschaft, auf einem Postscript-Drucker ausgebbar zu sein, in einer generischen Aspektklasse `PostscriptDruckbar`.

nur die zu ihnen passenden Aspekte, nicht aber die konkret zu bearbeitenden Materialien kennen müssen, und Materialien lediglich das durch die Aspektklasse spezifizierte Protokoll, unabhängig von konkreten Werkzeugen, zu implementieren haben.

Eine im Anwendungsbereich klinischer Großlaboratorien häufig vorkommende Tätigkeit ist es, sogenannte *Kommentare* bei Proben zu vermerken. Kommentare können sich u.a. auf den Zustand der Probe beziehen und dienen dem Laboratoriumspersonal als Hinweis, die kommentierte Probe besonders zu behandeln.

Das im Rahmen des Laborprojektes entwickelte LIS unterstützt diese Tätigkeit, indem es einen sogenannten *Kommentierer* als Werkzeug zur Verfügung stellt. Dieser erlaubt es beispielsweise, einer Probe neue Kommentare hinzuzufügen oder bestehende Kommentare zu verändern. Dazu bearbeitet das Werkzeug Kommentierer das Material Probe, indem es die entsprechenden Operationen der Probe über die Schnittstelle der Aspektklasse *Kommentierbar* aufruft.

Der Vorteil eines über die Aspektklasse Kommentierbar entkoppelten Zugriffs auf das Material Probe zeigt sich insbesondere daran, daß nicht nur Proben, sondern beispielsweise auch *Analyseergebnisse* kommentierbar sind. Damit aber kann das Werkzeug Kommentierer zur Kommentierung von Analyseergebnissen wiederverwendet werden.

2.3.3 Automaten

Werkzeuge und Materialien als zentrale Entwurfsmetaphern des WAM-Repertoires erlauben die Modellierung der aus dem täglichen Umgang vertrauten Arbeitsmittel und Arbeitsgegenstände. Die Konstruktion von Anwendungssystemen, bestehend aus reaktiven Werkzeugen und Materialien, deren Funktionalität nur über Werkzeuge zugänglich ist, ermöglicht dem Benutzer eine angemessene Erledigung seiner Aufgaben in Einklang mit dem zugrundegelegten Leitbild vom Arbeitsplatz für qualifizierte menschliche Tätigkeit.

Die Analyse der täglichen Arbeitspraxis zeigt aber auch, daß diese nicht nur von der situativen Erledigung qualifizierter Tätigkeiten geprägt ist, sondern gleichermaßen von "lästigen" Routinehandlungen. Diese werden durch die Entwurfsmetapher vom Werkzeug unzureichend unterstützt. Es gilt also, eine adäquate Metapher für die Erledigung von Routinehandlungen zu finden, die sich nahtlos neben die bisher vorgestellten Entwurfsmetaphern Werkzeug und Material einfügen läßt und überdies nicht mit dem zugrundegelegten Leitbild bricht[2].

Zur Beschreibung des Umgangs mit lästigen Routinehandlungen formuliert Gryczan zunächst den Begriff der Routine (vgl. [Gry96], S. 121):

> "Unter Routine wird eine wiederkehrende Handlung mit bekanntem Ergebnis verstanden."

[2]Die nun folgende Argumentation entstammt im wesentlichen der Arbeit von Gryczan und ist dort ausführlich nachzulesen (vgl. [Gry96], Kap. 5).

Soll der Benutzer davon entlastet werden, diese sich wiederholenden Handlungen jedesmal erneut "von Hand" durchführen zu müssen, so kann der Versuch unternommen werden, diese Routine durch Software zu automatisieren.

Dazu ist sie zunächst zu formalisieren (vgl. [Gry96], S. 123):

> "Unter formalisierter Routine wird eine benennbare, auf einem Rechner ausführbare Operationsfolge mit formal festgelegtem
>
> (1) Effekt auf Objekten oder
>
> (2) Ergebnis
>
> verstanden."

Als Entwurfsmetapher für die konstruktive Umsetzung formalisierter Routine in einem nach WAM entworfenen Softwaresystem wird der Begriff des *Automaten* verwendet und von Gryczan wie folgt definiert (vgl. [Gry96], S. 123):

> "Ein Automat ist die Vergegenständlichung einer formalisierten Routine, die über längere Zeiträume ohne äußere menschliche Eingriffe ablaufen kann. Ein Automat realisiert eine formalisierte Routine, die weitgehend ohne Kontextinformationen oder interaktive Steuerung auskommt. Einmal eingestellt, läuft die formalisierte Routine ab und produziert vorab festgelegte Resultate."

Automaten ermöglichen es demnach, den formalisierbaren Anteil routinisierter Handlungen zu modellieren und in einem Anwendungssystem softwaretechnisch umzusetzen. Entscheidend ist, daß solchermaßen definierte Automaten niemals für sich alleine existieren, sondern immer in den Kontext von Werkzeugen zur Erledigung des nicht-formalisierbaren Anteils menschlichen Arbeitshandelns eingebettet sind.

Aus der Definition des Automaten geht weiter hervor, daß Automaten, zumindest prinzipiell, ohne Eingriffe des Benutzers ablaufen können. Eine genauere Analyse des alltäglichen Umgangs mit Automaten aller Art, etwa einem *Kopierer*, zeigt, daß auch routinisierte Handlungen wie das Kopieren eines Dokumentenstapels der Einstellung, des Anstoßens und des Beendens bedürfen. Zusätzlich sehen Automaten in der Regel Möglichkeiten zur Abfrage und zur Darstellung ihres Verarbeitungszustandes, sowie zur Unterbrechung ihres Ablaufes, etwa in Form eines "Not-Aus-Schalters", vor. Aus diesem Grunde verwenden nach WAM entworfene Softwaresysteme häufig eine Kombination aus *Einstellwerkzeug* zur Realisierung der minimalen Benutzungsschnittstelle des Automaten und eigentlichem Automaten zur Realisierung der formalisierten Routinehandlung.

Die vorstehende Argumentation liefert damit eine *fachliche* Motivation des Automatenbegriffs. Auf softwaretechnischer Ebene lassen sich aber noch weitere Einsatzmöglichkeiten der Automatenmetapher finden. So erfordert beinahe jedes Anwendungssystem die persistente Speicherung von Daten. Aus softwaretechnischer Sicht ist es nun wünschenwert, die Details der Datenversorgung, etwa die dahinterstehende Datenbank oder das Dateisystem, vor der eigentlichen Anwendung verborgen zu halten.

Zu diesem Zwecke bietet sich in einem nach WAM entworfenen Softwaresystem der Einsatz von *technisch motivierten Automaten* zur Kapselung der Spezifika von angrenzenden

Softwaresystemen, in diesem Falle also der Datenbank, an[3]. Sie implementieren dazu technische Algorithmen und stellen sie den anderen Werkzeugen und Automaten des Softwaresystems in Form von Dienstleistungen zur Verfügung.

2.3.4 Arbeitsumgebung

Die bisher vorgestellten Entwurfsmetaphern des WAM-Repertoires erlauben es, Arbeitsmittel und Arbeitsgegenstände, sowie routinisierte Handlungen als Werkzeuge, Materialien und Automaten zu modellieren. Auf der anderen Seite sind es Menschen aber auch gewohnt, sich den eigenen Arbeitsplatz entsprechend ihrer jeweiligen Aufgaben, Ordnungsprinzipien und Gewohnheiten einzurichten.

Eine solchermaßen entstandene individuelle *Arbeitsumgebung* bildet den fachlichen Rahmen für die in der täglichen Arbeit verwendeten Arbeitsmittel und Arbeitsgegenstände und sollte daher in Form einer Entwurfsmetapher auch Eingang in nach WAM entworfene Softwaresysteme finden. Dazu wird sie nachfolgend fachlich motiviert und in den Kontext des Leitbildes vom Arbeitsplatz für qualifizierte menschliche Tätigkeit gestellt.

Betrachtet man das Urbild der Metapher, den Schreibtisch bei der Büroarbeit oder die Werkbank in der Werkstatt, so ist die Arbeitsumgebung zunächst einmal der Ort, an dem Werkzeuge, Materialien und Automaten ihren Platz haben. Der arbeitende Mensch kann Materialien mit Werkzeugen oder Automaten bearbeiten, neue Werkzeuge aufnehmen, Automaten einstellen oder Materialien ablegen. Kennzeichnend für diese Art von Arbeit ist, daß keinerlei Annahmen über die Reihenfolge getroffen werden, in der Arbeitsgegenstände mit Arbeitsmitteln bearbeitet werden. Diese Auswahl - Gryczan spricht in diesem Zusammenhang von *Ordnung* (vgl. [Gry96], S. 138) - obliegt ausschließlich dem qualifiziert arbeitenden Menschen, der für die fachlich korrekte Benutzung von Werkzeugen und Automaten verantwortlich ist.

Ein solches Verständnis von Arbeit entspricht genau dem zugrundegelegten Leitbild vom Arbeitsplatz für qualifizierte menschliche Tätigkeit und soll demnach auch als Konstruktionsprinzip für softwaregestützte Arbeitsumgebungen dienen. In Abgrenzung zur in Abschnitt 2.2 beschriebenen ablaufsteuernden Sichtweise und insbesondere zu aktuellen Tendenzen im Bereich des "Workflow Management"[4] soll durch die folgende Definition von Gryczan vermieden werden, steuernde Abläufe, etwa das automatische Starten von Werkzeugen durch übergeordnete kontrollierende Instanzen, in eine Arbeitsumgebung zu integrieren und dadurch mit dem zugrundegelegten Leitbild zu brechen (vgl. [Gry96], S. 138):

> "Eine Arbeitsumgebung zur Unterstützung qualifizierter menschlicher Tätigkeiten ist der Ort für eine anwendungsfachlich motivierte Zusammenstellung von Werkzeugen, Automaten und Materialien. Durch die Arbeitsumgebung werden keine Reihenfolgebedingungen für die Verwendung von Werkzeugen und Automaten festgelegt."

---

[3]Eine ausführliche Diskussion der Anbindung von Datenbanken an nach WAM entworfene Softwaresysteme findet sich in [WW94].

[4]Eine tiefergehende Diskussion und ausführliche Abgrenzung des hier vorgestellten Ansatzes zu den angesprochenen Tendenzen im Bereich des "Workflow Management" findet sich in [Gry96] und [Wul95].

Bislang wurde implizit davon ausgegangen, daß es sich bei der betrachteten Arbeitsumgebung um einen sogenannten *Einzelarbeitsplatz* handelt, d.h. das vorgestellte Konzept einer softwaregestützten Arbeitsumgebung enthält keinerlei Mittel zur Modellierung anderer Arbeitsumgebungen, mit denen der Benutzer dieser Arbeitsumgebung zwecks gemeinsamer Erledigung von Aufgaben zu kooperieren hat. Diese Annahme ist aber weder auf fachlicher Ebene, mit dem Trend zu Arbeitsteilung und Gruppenarbeit, noch auf softwaretechnischer Ebene, vor dem Hintergrund der fortschreitenden Vernetzung softwaregestützter Arbeitsplätze, zu halten.

Die Konsequenzen des Übergangs vom abgeschotteten Einzelarbeitsplatz zu Arbeitsplätzen in verteilten interaktiven bzw. verteilten, technisch eingebetteten Anwendungssystemen sind allerdings vielschichtig. Eine ausführliche Diskussion der fachlichen und technischen Implikationen würde den Rahmen dieser Arbeit weit übersteigen. Der nachfolgende Abschnitt vermittelt davon einen Eindruck.

2.4 Interaktive und technisch eingebettete Anwendungssysteme

Die Entwurfsmetaphern des WAM-Repertoires haben sich bislang vor allem bei Entwurf und Konstruktion *interaktiver Anwendungssysteme* bewährt. Diesbezüglich wurde bereits in der Einleitung dieser Arbeit festgehalten, daß alle Aktivitäten innerhalb interaktiver Anwendungssysteme stets als Reaktion auf Handlungen des Benutzers erfolgen, niemals jedoch das Anwendungssystem selbst seinem Benutzer den "Arbeitstakt" vorgibt.

Für diese Arbeit soll daher gelten, daß sich interaktive Anwendungssysteme aufgrund ihres reaktiven Charakters auch in die allgemeinere Klasse der *reaktiven Anwendungssysteme* einordnen lassen[5].

Wie schon im vorangegangenen Abschnitt erwähnt, lag der Fokus bei der Entwicklung interaktiver Anwendungssysteme mit der Methode WAM zunächst auf dem Entwurf und der Konstruktion geeigneter Software-Werkzeuge für sogenannte *Einzelarbeitsplatzumgebungen*, die ihre Benutzer bei der Erledigung qualifizierter Aufgaben angemessen unterstützen.

Neuere Arbeiten aus dem Kontext des Arbeitsbereichs Softwaretechnik erweitern die Methode WAM um Konzepte zur Unterstützung kooperativer Tätigkeiten (vgl. [Wul95, Gry96]). Diese führen zu sogenannten *kooperativen Arbeitsumgebungen*, die ihren Benutzern über den Austausch von gemeinsam benutzten Arbeitsgegenständen die Einbindung in kooperative Arbeitsprozesse erlauben. Auf technischer Ebene ist damit der Übergang vom Einprozeßraum zum Mehrprozeßraum und damit zu sogenannten *verteilten Anwendungssystemen* verbunden[6].

Auch kooperative, verteilt ablaufende Arbeitsumgebungen sind nach wie vor als interaktive Anwendungssysteme im oben beschriebenen Sinne zu verstehen. Grund dafür ist, daß die Koordination der Zusammenarbeit der einzelnen Arbeitsumgebungen ausschließlich über den

---

[5]Der Begriff des reaktiven Anwendungssystems (*reactive system*) ist in der wissenschaftlichen Diskussion keinesfalls eindeutig geklärt. So wird er teilweise auch verwendet, um die im weiteren Verlauf dieses Abschnittes motivierten *soft real-time systems* von sogenannten *hard real-time systems* abzugrenzen, die wiederum häufig einfach als *embedded systems* bezeichnet werden.

[6]Die für diese Arbeit maßgebliche Definition des Begriffs verteiltes Anwendungssystem findet sich nachfolgend in Abschnitt 4.1.

Austausch von Arbeitsgegenständen, nicht jedoch durch direkten Eingriff in andere Arbeitsumgebungen erfolgt. Als direkter Eingriff in eine andere Arbeitsumgebung wäre in diesem Zusammenhang etwa zu interpretieren, wenn der Benutzer einer Arbeitsumgebung ein Werkzeug, das sich in der Arbeitsumgebung eines anderen Benutzers befindet, starten und diesem mit der Aufforderung zur Benutzung "in die Hand" geben könnte.

Durch den bewußten Verzicht auf die Implementation derartiger Funktionalitäten kann sichergestellt werden, daß auch in kooperativen Arbeitsumgebungen alle Aktivitäten innerhalb der Software ausschließlich als Reaktion auf Handlungen des jeweiligen Benutzers der Arbeitsumgebung erfolgen. Damit entscheidet der Benutzer einer solchen kooperativen Arbeitsumgebung grundsätzlich selbst, ob, wann und wie er einen von einer anderen Arbeitsumgebung erhaltenen Arbeitsgegenstand bearbeitet bzw. ob und wann er diesen einer anderen Arbeitsumgebung zwecks Weiterbearbeitung zukommen läßt.

Das im Rahmen des Laborprojektes entwickelte LIS ist kein interaktives Anwendungssystem im Sinne der oben beschriebenen Einzelarbeitsplatzumgebungen bzw. kooperativen Arbeitsumgebungen. Statt dessen wurde es bereits in der Einleitung dieser Arbeit als *technisch eingebettetes Anwendungssystem* qualifiziert. Diese Unterscheidung läßt sich sowohl auf fachlicher als auch auf technischer Ebene belegen.

Auf fachlicher Ebene zeigt sich, daß technisch eingebettete Anwendungssysteme deutlich weniger der Unterstützung menschlichen Arbeitshandelns dienen, als vielmehr dem Betrieb *technischer Anlagen* (vgl. [Bro86], S. 745):

> "technische Anlage: Gesamtheit der maschinellen u.a. Ausrüstungen eines Betriebes, die zur Produktion oder Fertigung (Produktions- oder Fertigungs-A.), zur Energieerzeugung (Kraft-A.), zu Förder- oder Transportzwecken (Förder- oder Transport-A.) u.a. erforderlich sind."

Zum Betrieb einer technischen Anlage ist es notwendig, diese überwachen bzw. steuern zu können. Der Begriff der Steuerung entstammt der Regelungstechnik und wird dort wie folgt verwendet (vgl. [Jäg87], S. 1):

> "Steuern ist der Vorgang in einem System, bei dem eine oder mehrere Eingangsgrößen die Ausgangsgrößen aufgrund der Gesetzmäßigkeiten des Systems beeinflussen."

Im Vorgriff auf die nachfolgenden Kapitel sei schon an dieser Stelle angemerkt, daß bei dem in dieser Arbeit thematisierten LIS speziell der Aspekt des automatisierten Transports von Proben diskutiert werden soll. Im Sinne der obigen Begriffe ist dieser Teil eines Laborinformations- und *Steuerungssystems* als Transport-Anlage zur Überwachung und Steuerung des automatisierten Proben-Transports zu interpretieren, bei dem die Eingangsgröße (das LIS) die Ausgangsgröße (das Transportband für Proben) aufgrund der Gesetzmäßigkeiten des Systems (Algorithmen des LIS, Topologie des Transportbandes) beeinflußt.

Als Grundidee technisch eingebetteter Anwendungssysteme kann also auf der fachlichen Ebene festgehalten werden, daß diese dem Menschen die Überwachung und Steuerung technischer Anlagen ermöglichen sollen.

Ohne tiefergehende Diskussion sei ferner angemerkt, daß auch die innerhalb technisch eingebetteter Anwendungssysteme anzutreffenden Arbeitsplätze sich von den bislang im Kontext der Methode WAM modellierten Einzelarbeitsplatzumgebungen bzw. kooperativen Arbeitsumgebungen unterscheiden. So liegt der Fokus hier weniger auf der Unterstützung individueller menschlicher Tätigkeit, als vielmehr auf der personenunabhängigen Erfüllung von Funktionen innerhalb eines Gesamtsystems. Zu diesem Zweck werden sogenannte *Funktionsarbeitsplätze* mit einem begrenzten Repertoire an Arbeitsmitteln und Arbeitsgegenständen ausgerüstet, die ihrem Benutzer genau die Erbringung der angestrebten Funktionen innerhalb des Gesamtsystems gestatten.

Auf technischer Ebene wurde bezüglich interaktiver Anwendungssysteme festgehalten, daß ausschließlich der jeweilige Benutzer Aktivitäten innerhalb der Software veranlaßt. Ganz anders dagegen technisch eingebettete Anwendungssysteme. Dort sind es gerade die unabhängig vom Benutzer stattfindenden Ereignisse innerhalb der technischen Umgebung eines LIS, die einen wichtigen Teil der Aktivitäten innerhalb der Software auslösen.

Technisch eingebettete Anwendungssysteme reagieren also nicht nur auf Handlungen ihrer Benutzer, sondern zusätzlich auch auf Ereignisse, die in ihrer technischen Umgebung stattfinden. Sie sollen daher, genau wie die interaktiven Anwendungssysteme, in die allgemeinere Klasse der reaktiven Anwendungssysteme eingeordnet werden. Der Begriff des technisch eingebetteten Anwendungssystems (*embedded system*) läßt sich allerdings noch weiter differenzieren. Existieren hinsichtlich der einbettenden technischen Umgebung qualitative zeitliche Anforderungen an die Reaktionszeit oder Ausführungsdauer eines solchen Systems, zum Beispiel "... das System muß reagiert haben, bevor das nächste Ereignis eintritt ...", so wird vielfach von einem "weichen Echtzeitsystem" (*soft real-time system*) gesprochen. Existieren gar quantitativ spezifizierbare Anforderungen an das System, zum Beispiel "... das System muß innerhalb von einer Sekunde reagiert haben ...", so spricht man häufig von einem "harten Echtzeitsystem" (*hard real-time system*). Die Spezifikation von Realzeiteigenschaften objektorientierter, technisch eingebetteter Anwendungssysteme ist derzeit ein aktuelles Forschungsthema, kann jedoch nicht im Rahmen dieser Arbeit behandelt werden (vgl. statt dessen [BS95a, BS95b]). Es wird sich jedoch zeigen, daß bestimmte Komponenten des hier thematisierten LIS zumindest "weiche" Realzeiteigenschaften aufweisen müssen.

Im weiteren Verlauf dieser Arbeit wird außerdem deutlich werden, daß die Einbettung eines LIS in die komplexen Arbeitsprozesse eines klinischen Großlaboratoriums die Koordination der Zusammenarbeit von verschiedenen Komponenten des Systems und damit auf softwaretechnischer Ebene eine *verteilte* Realisierung erfordert.

3 Klinische Großlaboratorien als Anwendungsbereich von Laborinformations- und Steuerungssystemen

Klinische Großlaboratorien bilden den Anwendungsbereich von LIS. Dieser Anwendungsbereich ist, zumindest im Vergleich zu den bekannten Formen von Büroorganisationen, relativ unbekannt.

Bevor im nächsten Kapitel das im Rahmen des Laborprojektes entwickelte LIS vorgestellt wird, erscheint daher zunächst eine Beschreibung der im Anwendungsbereich herrschenden Einflußfaktoren und Randbedingungen notwendig. Diese wird nachfolgend durch einen Überblick über Arbeitsweise und Organisation klinischer Großlaboratorien komplettiert.

3.1 Einflußfaktoren und Randbedingungen

Der medizinische Bereich ist in den letzten Jahren zu einem stark expandierenden Einsatzfeld für Softwaresysteme geworden. Die Einführung moderner maschineller Diagnosemethoden, etwa im Bereich der computergestützten Tomographie (CT), erfordert deren adäquate Unterstützung durch Software. Dabei beschränken sich Softwaresysteme nicht nur auf die Steuerung von Diagnose-Hardware, sondern finden in verstärktem Maße auch bei der Interpretation der Ergebnisse durch den behandelnden Arzt Verwendung. Zudem führt der Zwang zur Kosteneinsparung im Gesundheitswesen im Rahmen des Gesundheits-Struktur-Gesetzes (GSG) dazu, daß erstmals auch im medizinischen Bereich ernsthaft über Rationalisierung nachgedacht werden muß.

Im Bereich der klinischen Laboratoriumsdiagnostik sind diese Rationalisierungsbemühungen zum Teil bereits weit gediehen. So wird bereits heute ein Großteil der routinemäßig anfallenden Laboruntersuchungen, zum Beispiel die Erstellung von Blutbildern oder die Bestimmung von Blutzuckerwerten, in zentralisierten Großlaboratorien, sogenannten Laborgemeinschaften, abgewickelt. Um die dabei auftretenden Volumina von mehreren tausend Analysen pro Tag abwickeln zu können, bedarf es des Einsatzes von Analyseautomaten und deren Einbettung in ein sie steuerndes und überwachendes Softwaresystem.

Während herkömmliche manuelle Analysemethoden durch den Einsatz von Technik immer weiter zurückgedrängt werden, wächst auf der anderen Seite der Zwang zu Qualitätskontrolle und Überwachung der automatisierten Analytik. Dieses führt zu einer Verschiebung im Arbeitsplatzprofil des Laboratoriumspersonals. Statt Analysen "von Hand" durchzuführen, besteht die tägliche Arbeit des Laboratoriumspersonals, neben der eigentlichen Bedienung und Beschickung der Analyseautomaten, mehr und mehr darin, die von Analyseautomaten erzeugten Ergebnisse durch den Einsatz statistischer Methoden und auf Basis von Erfahrungswerten auf ihre Plausibilität zu überprüfen. Auch diese Tätigkeiten sind ohne den Einsatz von unterstützender Software nicht denkbar.

Dabei stellt insbesondere die Durchführung von klinischen Feldversuchen an neuentwickelten Medikamenten, sogenannten Pharmastudien, hohe Ansprüche an die Einhaltung von Qualitätsstandards. Aus diesem Grunde existieren mit der *DIN EN 45 001* (vgl. [DIN90]), den *Grundsätzen der Guten Laborpraxis* (vgl. [GLP91]) und den sogenannten *CAP-Richtlinien*

(College of American Pathologists, vgl. [CAP88]) strenge nationale und internationale Qualitätsnormen, die auch bei Entwicklung und Betrieb eines LIS einzuhalten sind.

3.2 Arbeitsweise und Organisation

Betrachtet man nun die Arbeitsweise und Organisation klinischer Großlaboratorien, so lassen sich diese klassischerweise in die Bereiche *Präanalytik*, *Analytik* und *Postanalytik* aufteilen.

Die Präanalytik besteht aus der Entgegennahme der von den einzelnen *Einsendern* eingeschickten *Proben*, die i.d.R. aus Blut oder anderen Körperflüssigkeiten bestehen. Diese sind, etwa durch Zentrifugation, vorzubereiten und zum jeweiligen Arbeitsplatz oder Analyseautomaten zu transportieren. Zudem ist es Aufgabe der Präanalytik, die mitgelieferten *Anforderungen*, die angeben, welche Analysen aus dem angebotenen Analysespektrum mit einer bestimmten Probe durchgeführt werden sollen, zu erfassen. Dabei kommen in der Regel sogenannte Anforderungs-Karten und passende Beleglesegeräte, aber auch elektronische Datenträger und das Telefon zum Einsatz.

Die Analytik besteht, neben der eigentlichen Durchführung der angeforderten Analysen an Arbeitsplätzen oder Analyseautomaten, auch aus der *Validierung* der Ergebnisse durch das Laboratoriumspersonal. Dabei wird unter Einsatz von statistischen Methoden die Plausibilität der Ergebnisse anhand sogenannter *Kontrollen* überprüft. Erscheint ein Ergebnis nicht plausibel, so ordnet das Laborpersonal eigenverantwortlich eine Wiederholung der Analyse an, um etwa ein unwahrscheinlich hohes Ergebnis zu bestätigen oder dieses durch ein plausibleres zu ersetzen.

Im Bereich der Postanalytik erfolgt die Zusammenstellung der einzelnen Ergebnisse einer Anforderung zu einem sogenannten *Befund*. Dieser kann auch eventuell vorhandene Vorbefunde umfassen, um dem behandelnden Arzt einen Überblick über die Krankengeschichte zu ermöglichen. Befunde werden den Einsendern sowohl auf dem Postwege als auch per Fax oder elektronischem Datenaustausch zur Verfügung gestellt. In besonders wichtigen Fällen werden validierte Ergebnisse auch telefonisch übermittelt. Die telefonische Übermittlung von Befunden ist dabei Aufgabe der *Befundauskunft*, sie umfaßt auch die Beratung des behandelnden Arztes bei der Interpretation der Ergebnisse und die Entgegennahme von Folge-Anforderungen. Die Postanalytik wird durch die *Abrechnung* abgeschlossen. Hier werden dem einsendenden Arzt die durch das Labor erbrachten Leistungen in Rechnung stellt.

Zusammenfassend läßt sich also festhalten, daß die vollständige Bearbeitung einer Anforderung in einem klinischen Laboratorium eine anforderungs- und ergebnisabhängige Anzahl von Teilarbeitsschritten erfordert, zwischen denen i.d.R. kausale und temporale Abhängigkeiten bestehen. So ist es beispielsweise nicht möglich, eine Analyse durchzuführen, bevor die dazu benötigte Probe an dem entsprechenden Arbeitsplatz eingetroffen ist. Diesem Umstand wurde bei der Konzeption des im folgenden Kapitel beschriebenen LIS Rechnung getragen.

4 Das Laborprojekt

In diesem Kapitel wird zunächst die Architektur des im Rahmen des Laborprojektes entwickelten LIS *XLab* überblicksartig vorgestellt. XLab dient dieser Arbeit als Szenario, um die konzeptionellen Unterschiede zwischen interaktiven und technisch eingebetteten Anwendungssystemen anhand eines konkreten Anwendungsbereichs diskutieren zu können.

Aufbauend auf dem Überblick über die Architektur von XLab wird mit dem automatisierten Transport von Proben zum Arbeitsplatz oder Analyseautomaten ein konkretes Teilproblem innerhalb des Laborprojektes aufgegriffen.

Der sogenannte Proben-Transporter stellt einen ersten Lösungsansatz dar, für den sich allerdings schnell Grenzen auf technischer und konzeptioneller Ebene abzeichnen.

Sie führen mit dem sogenannten Transport-Subsystem zu einem zweiten, verteilten Lösungsansatz, der zwar die technischen Probleme des ersten Ansatzes beseitigt, auf konzeptioneller Ebene aber nach wie vor Grenzen aufweist.

4.1 Das Laborinformations- und Steuerungssystem XLab

XLab ist der Name eines nach der Methode WAM entworfenen LIS zum Betrieb klinischer Großlaboratorien. Als Hardware-Plattform kamen bei der Entwicklung vernetzte Unix-Workstations der Firma Sun unter dem Betriebssystem Solaris zum Einsatz. XLab ist in der Programmiersprache C++ implementiert, verwendet das relationale Datenbanksystem Sybase und realisiert die Benutzungsschnittstelle auf Basis des Netzwerk-Fenstersystems X-Windows, das auch bei der Namensgebung Pate gestanden hat.

Die Anforderung zur Einbettung von XLab in eine heterogene Hardwareumgebung, bestehend aus den Analyseautomaten diverser Hersteller und verschiedenen selbstentwickelten Komponenten, etwa eines Förderbandes zum Transport von Probenröhrchen, führte fast zwangsläufig zu der Entscheidung, XLab als verteiltes Anwendungssystem zu konzipieren, dessen einzelne Funktionseinheiten über gemeinsam genutzte Daten in der Datenbank und Mechanismen der Interprozeßkommunikation miteinander kooperieren.

Der Begriff des *verteilten Anwendungssystems* wurde im eher intuitiven Sinne bereits bei der Unterscheidung interaktiver und technisch eingebetteter Anwendungssysteme verwendet (vgl. Abschnitt 2.4). Er soll daher in Anlehnung an die Begriffsbildung von Heuser und Schill wie folgt präzisiert werden (vgl. [HS95], S. 104):

Begriff 4-1: Verteiltes Anwendungssystem

Ein verteiltes Anwendungssystem ist ein Anwendungsprogramm, das auf mehrere vernetzte Rechner ohne gemeinsamen Speicher verteilt ist und sich zudem durch dezentrale Verwaltung und Kooperation seiner (Software-) Komponenten auszeichnet.

Die in der Definition genannten Softwarekomponenten spielen bei Entwurf und Konstruktion verteilter Anwendungssysteme eine wichtige Rolle. Sie können dazu jeweils auf einer fachlichen und einer technischen Ebene betrachtet werden[7].

Auf der fachlichen Ebene sollen Softwarekomponenten in verteilten Anwendungssystemen im weiteren abstrakt als *Funktionseinheiten* bezeichnet werden (zur Begriffsbildung vgl. [JV87], S. 19). Jede Funktionseinheit ist dafür verantwortlich, einen bestimmten Teil der fachlichen Aufgaben des Anwendungssystems zu erbringen. In diesem Sinne können sowohl kooperative Arbeitsumgebungen in verteilten interaktiven Anwendungssystemen als auch Funktionsarbeitsplätze oder Steuerungsprogramme in technisch eingebetteten Anwendungssystemen als Funktionseinheiten interpretiert werden.

Funktionseinheiten sollen als elementare Softwarekomponenten eines verteilten Anwendungssystems verstanden werden, d.h. sie werden aus Sicht des Anwendungssystems nicht weiter unterteilt. Auf technischer Ebene wird dieser Umstand dadurch reflektiert, daß Softwarekomponenten in verteilten Anwendungssystemen als eigenständige *Prozesse* implementiert werden.

In Abschnitt 3.2 wurde bereits motiviert, daß die vollständige Bearbeitung einer Anforderung in einem medizinischen Labor eine anforderungs- und ergebnisabhängige Anzahl von Teilarbeitsschritten erfordert, zwischen denen in der Regel kausale Abhängigkeiten bestehen.

Um diesem Umstand Rechnung zu tragen, wird eine neu eintreffende *Anforderung* innerhalb des XLab-Systems zunächst durch die Funktionseinheit *Annahme* erfaßt und in einzelne Teilarbeitsschritte, die sogenannten *Aufträge*, zerlegt. Diese Aufträge werden mit den zwischen ihnen geltenden kausalen Abhängigkeiten versehen, so daß eine netzartige Struktur, ein sogenanntes *Auftragsnetz*, entsteht.

Das Netzen dieser Art zugehörige mathematische Modell ist in der theoretischen Informatik unter dem Begriff *Kausalnetz* bekannt (vgl. [JV87], S. 26). Es handelt sich dabei um einen speziellen zyklenfreien Typ von Petri-Netzen. Petri-Netze sind mathematische Strukturen, bestehend aus zwei Sorten von Elementen (sog. Stellen und Transitionen) und einer jeweils zwei verschiedenartige Elemente verbindenden Relation.

Interpretiert man nun die Stellen eines solchen Kausalnetzes als Aufträge und die Transitionen als Übergang von einem erfüllten Auftrag zu seinem Folgeauftrag, so können damit die oben beschriebenen Abhängigkeiten zwischen den (Teil-) Aufträgen einer Anforderung modelliert werden.

[7]Im Vorgriff auf das sechste Kapitel dieser Arbeit sei schon an dieser Stelle angemerkt, daß Softwarekomponenten in verteilten, technisch eingebetteten Anwendungssystemen, die nach WAM entworfen worden sind, durch die neue Entwurfsmetapher von der *Bearbeitungsstation* konstruktiv umgesetzt werden.

Abbildung 4-1 zeigt beispielhaft ein solches Netz von Aufträgen:

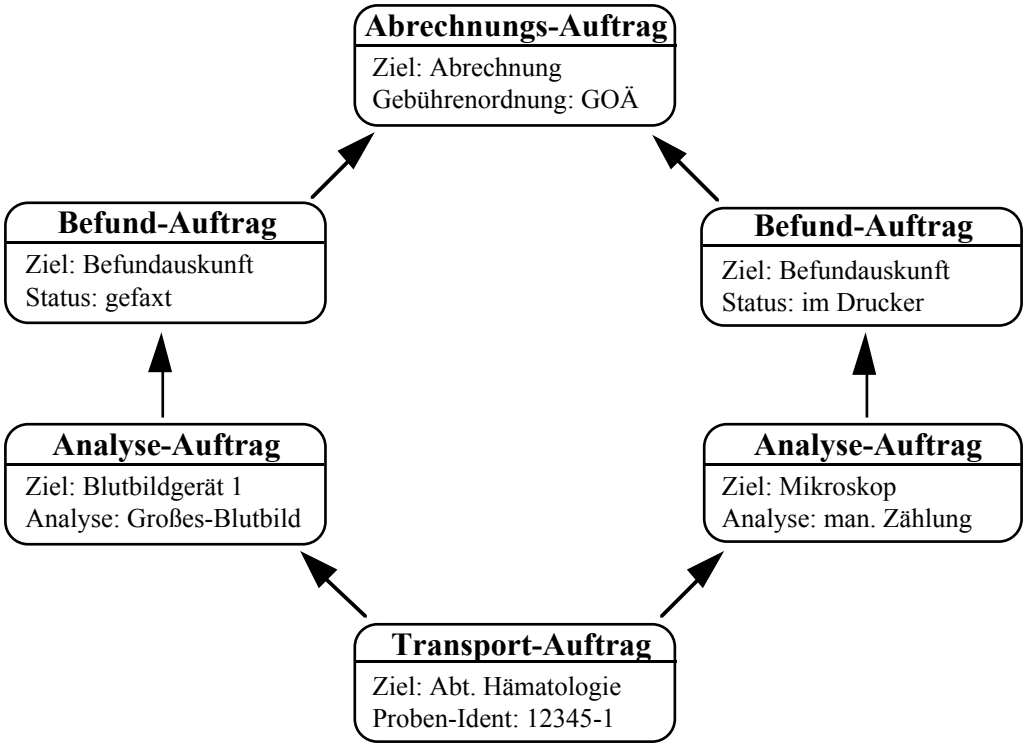

Abb. 4-1: Auftragsnetz (stark vereinfachte Darstellung)

Die Annahme legt das neu erzeugte Netz in der Datenbank ab und meldet dieses Ereignis einer zentralen Funktionseinheit, der sogenannten *Auftragssteuerung*[8].

Aufgabe der Auftragssteuerung ist es nun, das erzeugte Netz aus der Datenbank zu rekonstruieren und aufgrund der kausalen Abhängigkeiten zwischen den Aufträgen den nächsten ausführbaren Auftrag zu bestimmen. Der dazu verwendete Algorithmus ist ebenfalls dem mathematischen Modell entlehnt und basiert auf der Annahme, daß ein Auftrag nur dann bearbeitbar ist, wenn alle seine Vorgänger (-Aufträge) bereits erledigt sind.

In jedem Auftrag ist vermerkt, welche Funktionseinheit für seine Erfüllung zuständig ist. Funktionseinheiten können innerhalb des XLab-Systems Analyseautomaten, das Transportband, Arbeitsplätze oder ein beliebiger Hintergrundprozeß sein. Die Auftragssteuerung ist anhand dieser Informationen in der Lage, der zuständigen Funktionseinheit den zu erfüllenden Auftrag zwecks Abarbeitung zu übergeben. Hat die Funktionseinheit den zugewiesenen Auftrag erledigt, so meldet sie dieses der Auftragssteuerung, die nun ihrerseits durch erneute Anwendung des Algorithmus feststellt, ob noch weitere Aufträge im Rahmen dieser Anforderung zu vergeben sind.

Der Zusammenhang von Auftragssteuerung und weiteren Funktionseinheiten des XLab-Systems soll abschließend durch Abbildung 4-2 verdeutlicht werden:

---

[8]Eine Diskussion der fachlichen und technischen Implikationen des Austauschs von Ereignissen zwischen den Funktionseinheiten bzw. Prozessen eines verteilten Anwendungssystems ist nicht Thema dieser Arbeit. Eine Einführung in verschiedene Techniken zur Interprozeßkommunikation, z.B. Sockets, TLI oder CORBA, findet sich beispielsweise in [VS95].

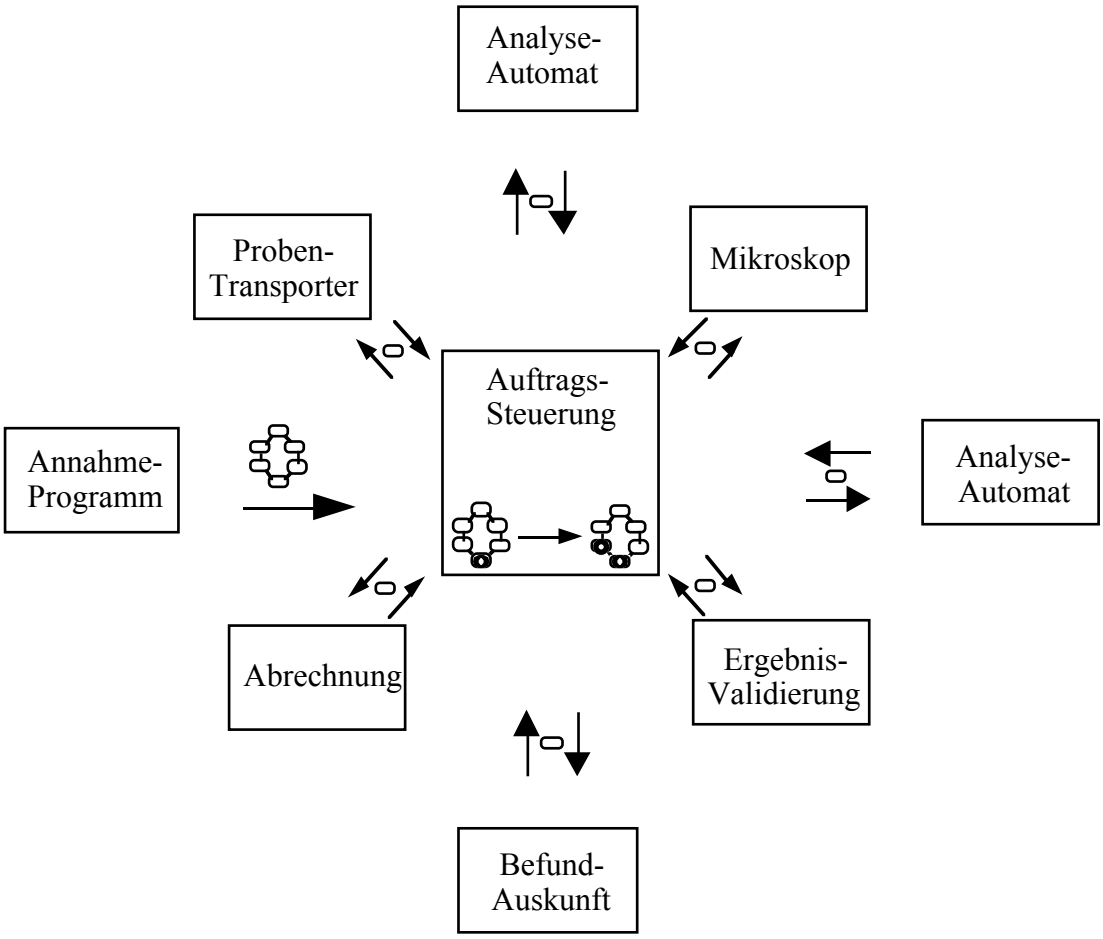

Abb. 4-2: Auftragssteuerung und weitere Funktionseinheiten

4.2 Der automatisierte Transport von Proben

Zentrale Aufgabenstellung beim Betrieb eines klinischen Großlaboratoriums ist die tägliche Erfassung von mehreren tausend Probenröhrchen, deren Transport zum jeweiligen Arbeitsplatz oder Analyseautomaten und die dort stattfindende (Massen-)Analytik. Zusätzlich muß die fachgerechte Aufbewahrung des eingesandten Probenmaterials bis zum Ablauf des jeweiligen Verfalldatums gesichert sein, um ein Wiederauffinden der Probenröhrchen bei eventuellen Nachforderungen zu gewährleisten.

Derartige Volumina an Proben und angeforderten Analysen können nur durch den Einsatz von Analyseautomaten mit vertretbarem Aufwand abgewickelt werden (vgl. Abschnitt 3.2). Weiteres Rationalisierungspotential ergibt sich bei einer Kombination von automatisiertem Transport des Probenröhrchens zum Arbeitsplatz oder Analyseautomaten, der dort durchgeführten automatischen Analytik und der roboterunterstützten Aufbewahrung von Probenmaterial in einer sogenannten Entnahmestation.

Diese automatisierte Kombination von Transport, Analytik und Aufbewahrung ist allerdings weltweit erst in Einzelfällen realisiert worden. Insbesondere weisen die handelsüblichen Analyseautomaten in der Regel weder mechanische noch softwaretechnische Schnittstellen zu den wenigen verfügbaren Transportsystemen auf.

Die im Anwendungsbereich von XLab realisierte Kombination von *Transportband*, unterschiedlichen *Blutbildgeräten* und *Entnahmestation* mußte daher mit einem hohen Aufwand für die Entwicklung eigener Hardware (Transportband, Entnahmeroboter) und die Adaption der Blutbildgeräte von Fremdherstellern erkauft werden. Auf softwaretechnischer Ebene resultierte daraus eine heterogene Hardwareumgebung, bei der, wie noch zu zeigen sein wird, die vorgefundenen mechanischen Randbedingungen und Restriktionen die Entwicklung des Softwaresystems XLab teilweise maßgeblich beeinflußten.

Das Transportband läßt sich dabei modellhaft als Ring vorstellen, auf dem sogenannte *Probenträger* kreisen (vgl. Abbildung 4-3).

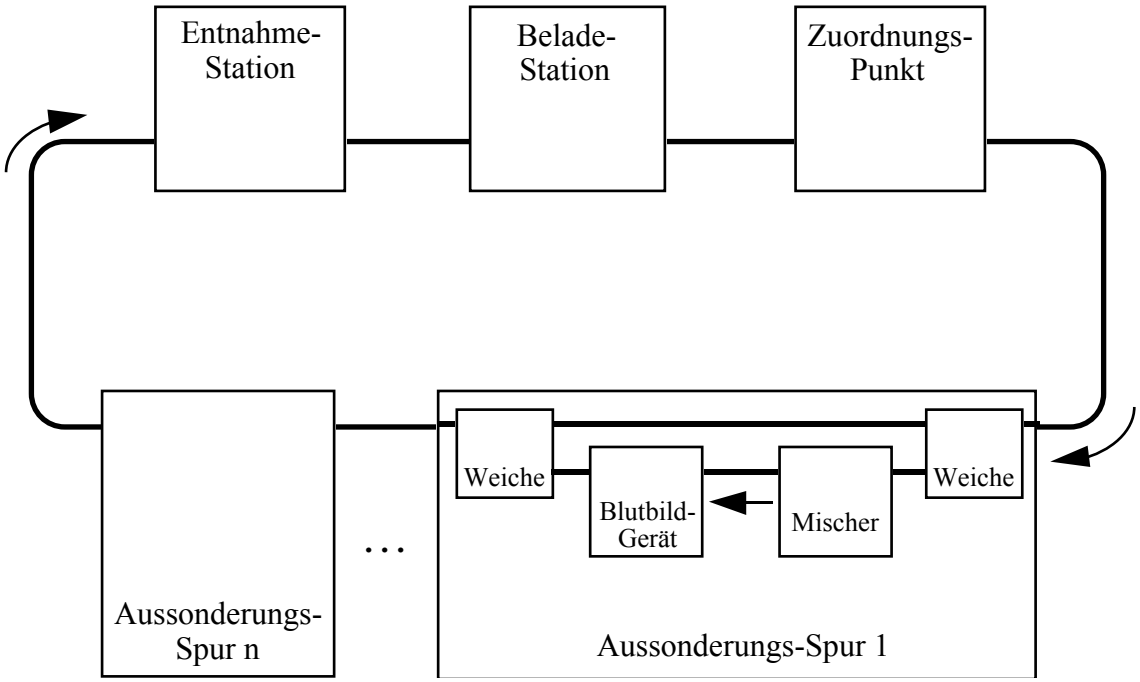

Abb. 4-3: Kombination von Transportband, Blutbildgeräten und Entnahmestation

Probenträger sind aus Kunststoff gefertigte Blöcke, die auf der Oberseite mit einer Bohrung für die Aufnahme genau eines Probenröhrchens versehen sind. In die Probenträger ist auf der Unterseite u.a. eine sogenannte *Tagcode-Pille* eingebaut, die es entsprechenden *Tagcode-Lesern* erlaubt, jede Tagcode-Pille, und damit auch jeden Probenträger, auf elektromagnetischem Wege eindeutig anhand des sogenannten *Tagcodes* zu identifizieren.

Das Transportband besteht nun, neben dem eigentlichen Ring zum Transport der Probenträger, aus einer *Beladestation*, dem sogenannten *Zuordnungspunkt* und einer konfigurierbaren Anzahl von *Aussonderungsspuren*.

An der Beladestation werden freie Probenträger vom Laborpersonal mit Probenröhrchen bestückt. Diese fahren nun auf dem Ring durch den Zuordnungspunkt. Der Zuordnungspunkt besteht aus einer Barcode-Leseeinheit und einem Tagcode-Leser. Werden nun gleichzeitig der Barcode auf dem Probenröhrchen und der Tagcode des Probenträgers gelesen, so kann eine eindeutige Zuordnung von Probenröhrchen und Probenträger vorgenommen werden, die für die weitere Steuerung der Probe durch das automatisierte Labor von entscheidender Bedeutung ist. In der täglichen Praxis kommt es aber immer wieder vor, daß der Barcode,

zum Beispiel aufgrund von Verschmutzungen, nicht lesbar ist. In diesem Fall muß er eingegeben, die Zuordnung also "von Hand" durchgeführt werden.

Die Arbeitsplätze oder Analyseautomaten sind über die sogenannten *Aussonderungsspuren* an das Transportsystem angekoppelt. Eine Aussonderungsspur besteht aus einem ca. 2 Meter langen Stück Transportband, das parallel zum eigentlichen Ring verläuft. Aussonderungsspuren besitzen jeweils am Anfang und am Ende eine *Weiche*, mit deren Hilfe Probenträger in die Aussonderungsspur hinein- bzw. wieder auf den Ring hinausgesteuert werden können. Dazu besitzen Weichen ebenfalls einen Tagcode-Leser, um Probenträger anhand ihres Tagcodes zu identifizieren und ggf. aussteuern zu können.

Wird nun ein Probenträger zwecks Bestimmung eines Blutbildes in eine solche Aussonderungsspur gesteuert, so muß das Probenmaterial zunächst gemischt werden, um eine gleichmäßige Verteilung der einzelnen Blutbestandteile innerhalb des Probenröhrchens zu gewährleisten. Dazu hält der Probenträger zunächst unter einem sogenannten *Mischer*, wo das Probenmaterial mittels eines in das Probenröhrchen eingeführten *Stempels* durchmischt wird.

Im Anschluß an den Mischvorgang fährt der Probenträger weiter bis unter das Blutbildgerät. Dieses führt eine *Ansaugpipette* in das Probenröhrchen ein, entnimmt die für die photometrische Messung der Parameter eines Blutbildes notwendige Menge an Analysematerial und führt die eigentliche Bestimmung des Blutbildes durch. Ist dieser Vorgang erfolgreich abgeschlossen, so wird der Probenträger über die Weiche am Ende der Aussonderungsspur zurück auf den eigentlichen Ring gesteuert.

Die weitere Verwendung des Probenmaterials, respektive die weitere Steuerung des Probenträgers, ist sowohl von den Ergebnissen des durchgeführten Blutbildes als auch von eventuell weiter durchzuführenden Analysen abhängig. Konnte beispielsweise die für das Blutbild notwendige Zellzählung aufgrund eines schlechten Zustandes des Probenmaterials nicht zuverlässig durchgeführt werden, so kann sowohl eine Wiederholung der Messung als auch eine manuelle Zählung der Blutzellen mittels Mikroskop erforderlich werden.

Ferner ist es möglich, daß das Probenmaterial zwecks Durchführung weiterer Analysen noch an anderen Arbeitsplätzen oder Analyseautomaten benötigt wird. In diesem Falle ist es erneut zu einer entsprechenden Aussonderungsspur zu transportieren.

Wird das Probenmaterial dagegen vorerst nicht weiter benötigt, so kann der Probenträger zur Entnahmestation gesteuert werden, wo das Probenröhrchen durch den Entnahmeroboter dem Probenträger entnommen und in einem *Probenmagazin* für einen eventuellen, späteren Zugriff aufbewahrt wird.

Allerdings ist auch das hier auf Ebene von Hardwarekomponenten beschriebene Modell einer Kombination von Transportband, Blutbildgeräten und Entnahmestation idealisiert. In der Praxis ist immer wieder mit mechanisch oder durch Verschmutzung bedingten Problemen beim Mischen oder Pipettieren des Probenmaterials zu rechnen, die bis zur vollständigen Blockade einer Aussonderungsspur führen können. Da eine solche mechanisch bedingte Blockade nur schwer durch die steuernde Software erkannt werden kann, treten in einem solchen Falle "Rückstaus" auf, die letztlich im Stillstand des gesamten Transportbandes resultieren.

Aus diesem Grunde wurde für das Laborpersonal die Möglichkeit geschaffen, Weichen, Mischer und Ansaugpipette der jeweils betroffenen Aussonderungsspur mit Hilfe des

sogenannten *Bypass-Knopfes* in einen "sicheren", initialen Zustand zurückzuversetzen. In diesem Fall werden alle sich bereits auf der Aussonderungsspur befindlichen Probenträger, ohne weitere Misch- oder Analysevorgänge durchzuführen, direkt auf den eigentlichen Ring zurückgesteuert. Ferner wird eine erneute Aussteuerung von Probenträgern auf die betroffene Aussonderungsspur bis zur Wiederfreigabe durch das Laborpersonal mechanisch unterbunden.

Aus softwaretechnischer Sicht entscheidendes Kennzeichen des Bypass-Knopfes ist jedoch, daß dieser eine mechanische Unterbrechung des durch die Software gesteuerten Ablaufs ermöglicht, die von der Software weder verhindert, noch unmittelbar bemerkt oder zurückgenommen werden kann. Anhand der weiteren Ausführungen dieser Arbeit wird deutlich werden, daß die Existenz von externen mechanischen Unterbrechungsmöglichkeiten schwerwiegende Auswirkungen auf Entwurf und Konstruktion der steuernden Software hat.

4.3 Der Proben-Transporter

In den vorangegangenen Abschnitten wurde das LIS XLab beschrieben und mit dem automatisierten Transport von Proben ein konkretes Teilproblem auf der Ebene von Hardwarekomponenten vorgestellt. Die sich daraus ergebende softwaretechnische Aufgabenstellung soll zunächst stichpunktartig zusammengefaßt und in einen ersten Lösungsansatz überführt werden. Eine nähere Betrachtung läßt allerdings die Grenzen des Ansatzes schnell deutlich werden. Diese sollen nachfolgend auf technischer und konzeptioneller Ebene diskutiert werden.

4.3.1 Lösungsansatz

Eine innerhalb des XLab-Systems für den automatisierten Transport von Proben zu Arbeitsplätzen oder Analyseautomaten verantwortliche Funktionseinheit

- ist in eine heterogene Hardwareumgebung, bestehend aus den Analyseautomaten unterschiedlicher Hersteller und diversen eigenentwickelten Komponenten, einzubetten
- ist für die eindeutige Zuordnung von Probenröhrchen und Probenträger zuständig; dazu muß sie den Probenträger beim Passieren des Zuordnungspunktes anhand des Tagcodes identifizieren und gleichzeitig den Barcode auf dem Probenröhrchen lesen
- ist für die auslastungsabhängige Verteilung der Proben auf die für die Durchführung der angeforderten Analyse(n) geeigneten Analyseautomaten verantwortlich[9]
- berechnet den jeweils günstigsten Weg zum ausgewählten Analyseautomaten und nimmt die dazu notwendige Ansteuerung der einzelnen Weichen vor
- hat eine adäquate Reaktion auf die Betätigung des Bypass-Knopfes zu implementieren

[9]In der Realität sind die hier geschilderten Zusammenhänge noch komplizierter. So muß eine Auswahl des für die Durchführung geeigneten Analyseautomaten nicht nur den jeweiligen Typ des Automaten berücksichtigen, sondern auch den Typ des verwendeten Probenröhrchens und die Frage, ob der betreffende Analyseautomat vom Laborpersonal auf Basis der Messung spezieller Kontrollproben und Kalibratoren überhaupt für die Durchführung patientenbezogener Analytik freigegeben worden ist.

Bereits in Abschnitt 4.1 wurde (graphisch) angedeutet, die hier stichpunktartig zusammengefaßte Aufgabenstellung einer einzigen Funktionseinheit des XLab-Systems, dem sogenannten *Proben-Transporter*, zu übertragen (vgl. Abb. 4-2).

Dieser wird von der Auftragssteuerung mit den in Abbildung 4-1 angedeuteten *Transport-Aufträgen* versorgt. Transport-Aufträge wiederum enthalten hinreichend viele Informationen, um dem Proben-Transporter die Erbringung seiner Aufgaben gegenüber der Auftragssteuerung zu ermöglichen. Anders ausgedrückt ist der Proben-Transporter mit Hilfe von Transport-Aufträgen in der Lage, den automatisierten Transport von Proben zum Arbeitsplatz oder Analyseautomaten ohne die Hilfe weiterer Komponenten abzuwickeln.

Weitere Aufgabe des Proben-Transporters ist es, Transport-Aufträge nach erfolgter Zuordnung und abgeschlossenem Transport als erfüllt zu markieren und diesen Umstand der Auftragssteuerung zu melden. Damit wird die Auftragssteuerung wiederum in die Lage versetzt, durch Anwendung des eigenen Algorithmus den nächsten zu bearbeitenden Auftrag an die zuständige Funktionseinheit zu vergeben.

4.3.2 Grenzen

Der Betrieb des Transportbandes erfordert eine hohe Stabilität und Performanz der steuernden Software. Ausfälle können nicht nur zu den bereits erwähnten "Rückstaus" führen, sondern, bedingt durch die als Folge entstehende Beeinträchtigung des Probentransports, auch zu schwerwiegenden Verzögerungen aller Arbeitsabläufe innerhalb des Labors. Eine schlechte Performanz der Software resultiert in dem verzögerten Schalten von Weichen, dadurch verursachten "Ehrenrunden" und letzendlich wiederum in durch hohes "Verkehrsaufkommen" bedingten Stauungen.

Zudem erfordert das vorgestellte Konzept eines Proben-Transporters aber auch die Möglichkeit, Benutzereingaben, etwa zur manuellen Zuordnung von Probenträger und Probenröhrchen, entgegenzunehmen und in die weitere Verarbeitung einfließen zu lassen. Im Falle von XLab sollte also eine Benutzungsschnittstelle des Proben-Transporters auf Basis des Netzwerk-Fenstersystems X-Windows zur Verfügung gestellt werden.

Die Erfahrungen beim Einsatz von Anwendungen mit hohen Stabilitäts- und Performanzanforderungen, sowie bei der Verwendung graphischer Benutzungsoberflächen lassen allerdings eine Erreichung beider Ziele innerhalb einer Funktionseinheit bzw. innerhalb eines Prozesses als wenig aussichtsreich erscheinen. So ist nicht nur die Performanz von Anwendungen mit graphischen Benutzungsoberflächen in der Regel schlechter als die von reinen Hintergrundprozessen, auch die Stabilität der Anwendungen leidet deutlich unter den durch den Einsatz eines Fenstersystems bedingten, zusätzlichen Fehlerquellen[10].

[10]Die Stabilität von Anwendungen, die auf graphischen Benutzungsoberflächen basieren, ist nicht Thema dieser Arbeit. Daher sei an dieser Stelle nur kurz angemerkt, daß in der Kontrolle von Anwendungsprogrammen durch das übergeordnete Fenstersystem ein nicht unbeträchtliches Fehlerpotential steckt. Dazu zählt nicht nur der Benutzer, der das kritische Anwendungsprogramm durch das Zerstören eines entsprechenden Fensters "abschießt". Ein ebenso großes Unsicherheitspotential steckt prinzipiell in der Verwendung eines Fenstersystems selbst. So ist leicht einzusehen, daß zum reibungslosen Ablauf eines Anwendungsprogramms mit graphischer Benutzungsoberfläche nicht nur das Betriebssystem, sondern auch das Fenstersystem stabil laufen muß. Damit aber werden, legt man die Komplexität und Fehleranfälligkeit von (Netzwerk-) Fenstersystemen zugrunde, die Voraussetzungen für einen reibungslosen Ablauf um zusätzliche Faktoren erschwert.

Das Konzept eines Proben-Transporters stößt also an dieser Stelle an eine erste, vorwiegend durch technische Gesichtspunkte bedingte, Grenze.

Eine weitere Schwachstelle des Proben-Transporters wurde bei der Modellierung einer adäquaten Reaktion auf die Betätigung des Bypass-Knopfes deutlich. So wurde bei der Konzeption des Proben-Transporters davon ausgegangen, daß dieser mit Hilfe von Transport-Aufträgen in der Lage ist, den automatisierten Transport von Proben zum Arbeitsplatz oder Analyseautomaten ohne die Hilfe weiterer Komponenten abzuwickeln (vgl. Abbildung 4-4).

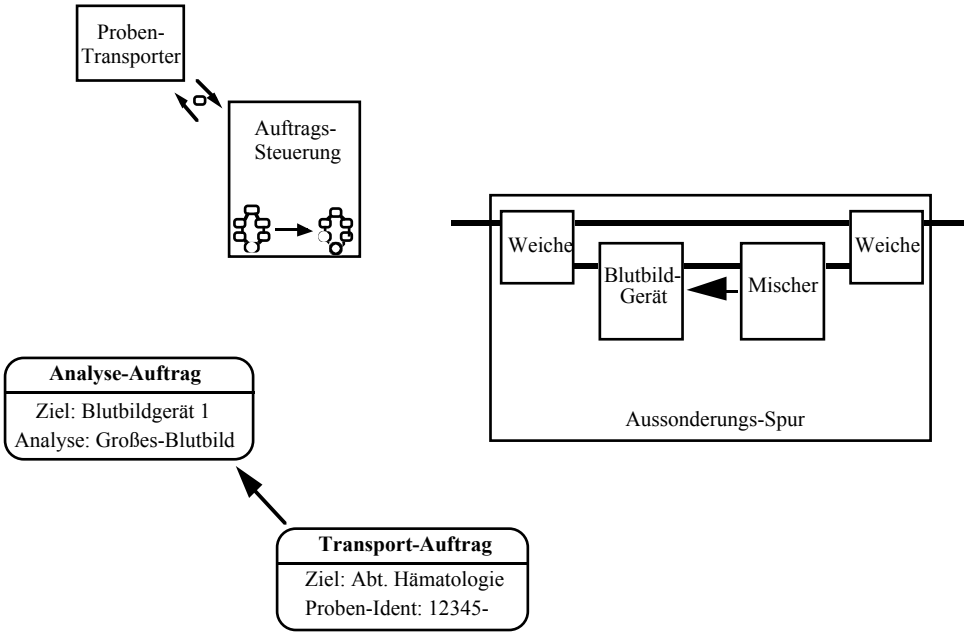

Abb. 4-4: Automatisierter Transport von Proben zum Analysegerät

Dieses Konzept stieß jedoch bei der Modellierung des Kriteriums, ab wann ein Transport-Auftrag als erledigt zu betrachten sei, an seine Grenzen.

Interpretiert man dazu die Verantwortlichkeiten des Proben-Transporters im oben beschriebenen Sinne, so müßte dieser selbständig entscheiden können, ob und wann ein Transport-Auftrag als erfolgreich abgeschlossen zu betrachten ist. Das ist aber bei näherer Betrachtung der durch die einbettenden Hardware-Komponenten gegebenen Randbedingungen nicht der Fall.

So wurde bislang davon ausgegangen, daß der Proben-Transporter anhand eines Transport-Auftrages über hinreichende Informationen verfügt, um einen durch den Zuordnungspunkt identifizierten Probenträger an einer bestimmten Weiche in eine Aussonderungsspur auszusteuern. Ein solchermaßen erfüllter Transport-Auftrag kann nun vom Proben-Transporter an die Auftragssteuerung gemeldet werden, die ihrerseits den nachfolgenden Analyse-Auftrag an den jeweiligen Analyseautomaten vergeben kann.

Dieses Modell erweist sich allerdings nur unter der Voraussetzung einer einwandfreien Funktion von Mischer und Gerät als hinreichend, es ist also in der Praxis nicht aufrechtzuerhalten (vgl. Abschnitt 4.2).

Zur Verdeutlichung dieses Zusammenhangs soll die im vorangegangenen Abschnitt beschriebene Funktionalität des Bypass-Knopfes herangezogen werden. Sie bestand darin,

Weichen, Mischer und Ansaugpipette der jeweils betroffenen Aussonderungsspur auf mechanischem Wege in einen "sicheren", initialen Zustand zurückzuversetzen. Ziel einer Betätigung des Bypass-Knopfes war es dabei, alle sich bereits auf der Aussonderungsspur betroffenen Probenträger, ohne weitere Misch- oder Analysevorgänge durchzuführen, direkt auf den eigentlichen Ring zurückzusteuern. Der Bypass-Knopf übernimmt damit die Rolle einer externen mechanischen Unterbrechung des durch die Software gesteuerten Ablaufs, die von dieser weder verhindert, noch zurückgenommen werden kann.

Weiter oben wurde die Aussteuerung des Probenträgers vom eigentlichen Ring des Transportbandes auf die Aussonderungsspur aus Perspektive des Proben-Transporters als Kriterium eines erledigten Transport-Auftrags motiviert. Für den Proben-Transporter gilt somit, daß der betreffende Transport-Auftrag aus der Menge der verwalteten Transport-Aufträge entfernt werden kann.

Wird der Ablauf "*Aussteuern, Mischen, Analytik durchführen*" nun aber durch die Betätigung des Bypass-Knopfes mechanisch unterbrochen, so ist der Transport des Probenträgers zum Gerät aus der Perspektive des Gesamtsystems tatsächlich keinesfalls abgeschlossen. Statt dessen muß der Probenträger neu verteilt und zu einem anderen geeigneten Gerät transportiert werden.

Dieser Umstand ist aber für den Proben-Transporter nicht ersichtlich, da dieser den betreffenden Transport-Auftrag bereits "vergessen" hat. Anders ausgedrückt gilt, daß die Fälle "*Aussteuern, Mischvorgang und Analytik erfolgreich abgeschlossen*", sowie "*Aussteuern, Bypass-Knopf betätigt, Mischvorgang und Analytik nicht durchgeführt*" für den Proben-Transporter ununterscheidbar und damit nicht adäquat behandelbar sind.

Damit wird deutlich, daß das Konzept des Proben-Transporters bei der Modellierung einer adäquaten Reaktion auf die Betätigung des Bypass-Knopfes nun auch unter konzeptionellen Gesichtspunkten an seine Grenzen stößt.

Als Ergebnis dieses Abschnitts soll daher festgehalten werden:

- Auf technischer Ebene erfordern die Stabilitäts- und Performanzanforderungen an eine Software zum Betrieb des Transportbandes die Trennung kritischer Kernfunktionalität von denjenigen Komponenten, die Interaktion mit dem Benutzer erfordern. Als kritische Kernfunktionalität kann im Falle des Proben-Transporters vor allem die Ansteuerung der Weichen und das Verhindern von Stauungen angesehen werden. Interaktion mit dem Benutzer ist vor allem bei der manuellen Zuordnung von Probenröhrchen zu Probenträger aufgrund mangelnder Lesbarkeit des Barcodes notwendig.
- Auf konzeptioneller Ebene führt die Existenz des Bypass-Knopfes als externe mechanische Unterbrechungsmöglichkeit dazu, daß das Konzept eines Proben-Transporters um zusätzliche Mechanismen zur Behandlung von Ausnahmen ergänzt werden muß.

4.4 Das Transport-Subsystem

Als Konsequenz aus dem Ergebnis des vorangegangenen Abschnitts wird nachfolgend ein erweiterter Lösungsansatz für das Problem des automatisierten Transports von Proben vorgestellt. Das sogenannte *Transport-Subsystem* sucht die konzeptionellen und technischen Grenzen eines als Funktionseinheit modellierten Proben-Transporters durch Dekomposition

und Verteilung von Verantwortlichkeiten auf mehrere (Sub-) Funktionseinheiten zu überwinden.

Dabei wird deutlich, daß dieser klassische Ansatz zur Reduktion von Komplexität auf technischer Ebene zwar eine deutliche Verbesserung gegenüber dem Konzept eines Proben-Transporters darstellt, auf konzeptioneller Ebene aber nach wie vor Grenzen aufweist. Diese bieten allerdings Ansatzpunkte, um im nächsten Kapitel vom konkreten Problem zu abstrahieren und die konzeptionellen Unterschiede zwischen der Modellierung interaktiver und der Modellierung technisch eingebetteter Anwendungssysteme herauszuarbeiten.

4.4.1 Lösungsansatz

Ein Ansatz zur Reduktion von Komplexität läßt sich aus dem klassischen Teile-und-Herrsche-Prinzip (*divide et impera*) herleiten. Für Funktionseinheiten in verteilten Anwendungssystemen könnte die Anwendung dieses Prinzips bedeuten, die fachlichen Aufgaben einer komplexen Funktionseinheit auf mehrere, weniger komplexe Funktionseinheiten zu verteilen, die fortan gemeinsam für deren Erbringung verantwortlich sind.

In diesem Sinne soll auch das Konzept eines als Funktionseinheit modellierten Proben-Transporters aufgegeben werden und durch das sogenannte *Transport-Subsystem* ersetzt werden.

Der Begriff des Subsystems wird von Wirfs-Brock et al. wie folgt definiert (vgl. [WWW90], S. 135):

> "Subsystems are groups of classes, or groups of classes and other subsystems, that collaborate among themselves to support a set of contracts. From outside the subsystem, the group of classes can be viewed as working closely together to provide a clearly delimited unit of functionality. From inside, subsystems reveal themselves to have a complex structure."

Entscheidend an Wirfs-Brocks Subsystembegriff ist in diesem Zusammenhang nicht, daß Klassen und nicht Funktionseinheiten bzw. Prozesse als konstituierende Komponenten eines Subsystems bezeichnet werden. Entscheidend ist vielmehr, daß Subsysteme aus mehreren Komponenten gebildet werden, die zur Erbringung einer Menge von Verträgen (contracts) zusammenarbeiten. Der Begriff des Vertrages ist aber eng mit dem Begriff der Verantwortlichkeit (responsibility) verbunden.

Dazu wieder Wirfs-Brock (vgl. [WWW90], S. 117):

> "A contract defines a cohesive set of responsibilities that a client can depend on."

Verantwortlichkeiten können nun auf einer fachlichen Ebene wiederum als Verpflichtung zur Erbringung von Aufgaben interpretiert werden.

Damit wird deutlich, daß sich Funktionseinheiten und Subsysteme aus Perspektive eines verteilten Anwendungssystems gleich verhalten, da beide für die Erbringung eines bestimmten Teils der Aufgaben des Anwendungssystems verantwortlich sind. Anders

ausgedrückt ist es aus Perspektive des verteilten Anwendungssystems ohne weiteres möglich, Funktionseinheiten durch Subsysteme zu ersetzen bzw. umgekehrt Subsysteme zu Funktionseinheiten zusammenzufassen.

Diese Substituierbarkeit soll nun beim Übergang vom Proben-Transporter zum Transport-Subsystem ausgenutzt werden. So bestand die Schnittstelle zwischen der das XLab-Gesamtsystem repräsentierenden Auftragssteuerung und dem Proben-Transporter im Austausch von Transport-Aufträgen. Nach den obigen Ausführungen müßte es also möglich sein, sie in dieser Form auch für das nachfolgend beschriebene Transport-Subsystem zu verwenden.

Das Transport-Subsystem besteht dazu aus den Funktionseinheiten Automatischer-Zuordnungspunkt, Manueller-Zuordnungspunkt, Bandsteuerung und dem sogenannten Transport-Supervisor, die auf technischer Ebene durch eigenständige Prozesse realisiert werden (vgl. Abbildung 4-5).

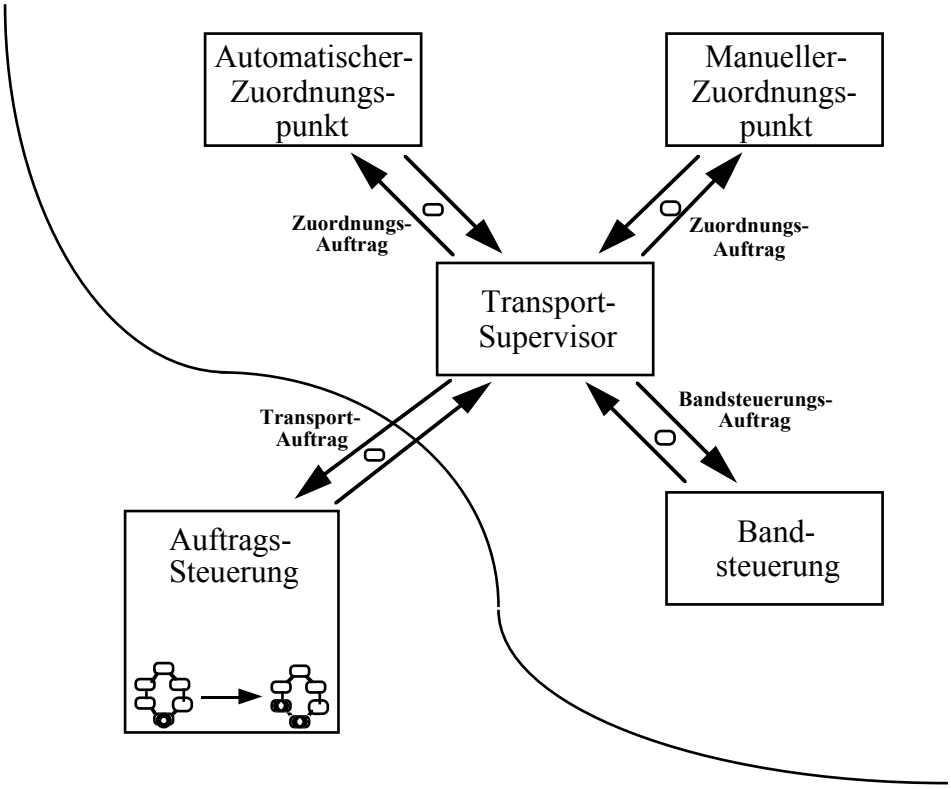

Abb. 4-5: Die Funktionseinheiten des Transport-Subsystems

Die wichtigste Funktionseinheit des Transport-Subsystems ist der *Transport-Supervisor*. Er dient nicht nur als Schnittstelle zwischen Transport-Subsystem und XLab-Gesamtsystem, sondern ist zusätzlich auch für die Koordination der anderen Funktionseinheiten des Transport-Subsystems und die Behandlung von Ausnahmen innerhalb des Subsystems zuständig.

In seiner Rolle als Schnittstelle zwischen Transport-Subsystem und XLab-Gesamtsystem nimmt der Transport-Supervisor die Transport-Aufträge von der Auftragssteuerung entgegen und erzeugt aus diesen spezialisierte *Sub-Aufträge*, die an die entsprechenden

Funktionseinheiten des Transport-Subsystems übergeben werden. Haben diese ihre jeweiligen Sub-Aufträge erfüllt, so kann der Transport-Supervisor auch den ursprünglichen Transport-Auftrag als abgeschlossen betrachten und an die Auftragssteuerung zurückgeben.

Der *Automatische-Zuordnungspunkt* ist innerhalb des Transport-Subsystems für die automatisierte Zuordnung von Probenträger und Probenröhrchen über die am Transportband aufgestellten Barcode-Leseeinheit zuständig. Dazu erhält er vom Transport-Supervisor sogenannte *Zuordnungs-Aufträge* und hat die Aufgabe, dort den Tagcode des Probenträgers und den Barcode des Probenröhrchens zu vermerken. Um die Stabilität und Performanz dieser Komponente des Transport-Subsystems zu gewährleisten, ist der Automatische-Zuordnungspunkt als Hintergrundprozeß implementiert.

Konnte der Barcode nicht erfolgreich gelesen werden, so wird dieser Umstand vom Automatischen-Zuordnungspunkt an den Transport-Supervisor gemeldet. Der Transport-Supervisor kann nun den *Manuellen-Zuordnungspunkt* beauftragen, dem Benutzer die Eingabe des Barcodes "von Hand" zu ermöglichen. Dazu besitzt der Manuelle-Zuordnungspunkt eine Benutzungsschnittstelle und wird vom Transport-Supervisor ebenfalls mit Zuordnungs-Aufträgen versorgt.

Eine weitere Aufgabe des Transport-Supervisors ist die auslastungsabhängige Verteilung der Proben auf Arbeitsplätze bzw. die für die Durchführung der angeforderten Analyse(n) geeigneten Analyseautomaten. Dazu ermittelt der Transport-Supervisor den am wenigsten ausgelasteten Arbeitsplatz oder Analyseautomaten und trägt diesen in sogenannte *Bandsteuerungs-Aufträge* ein. Diese werden an die Bandsteuerung übergeben.

Die *Bandsteuerung* ist fortan lediglich für die Berechnung des jeweils günstigsten Weges und die konkrete Ansteuerung der Weichen zuständig. Analog zu den in Abschnitt 3.3 beschriebenen Transport-Aufträgen des ursprünglichen Modells werden Bandsteuerungs-Aufträge von der Bandsteuerung mit dem Aussteuern des Probenträgers in die Aussonderungsspur des Gerätes als erledigt angesehen und an den Transport-Supervisor zurückgegeben. Damit konnte die Bandsteuerung gegenüber dem Proben-Transporter des ursprünglichen Modells auf einen unabhängigen Hintergrundprozeß mit verringerter Funktionalität, dafür aber höherer Stabilität und Performanz reduziert werden.

Interessantester Aspekt des Transport-Subsystems ist jedoch die Rolle des Transport-Supervisors bei der Behandlung sogenannter übergeordneter Ausnahmen. Der Begriff der übergeordneten Ausnahme innerhalb eines verteilten Anwendungssystems ist von entscheidender Bedeutung für die weitere Argumentation und soll daher wie folgt präzisiert werden:

Begriff 4-2: Übergeordnete Ausnahme

Als übergeordnete Ausnahmen werden diejenigen Ausnahmen und Fehlerbedingungen innerhalb eines verteilten Anwendungssystems verstanden, die nicht innerhalb einer einzelnen Funktionseinheiten, etwa anhand des Rückgabewertes einer fehlgeschlagenen Operation, erkannt und behandelt werden können, sondern deren Erkennung und Behandlung die Zusammenarbeit mehrerer Funktionseinheiten des verteilten Anwendungssystems erfordert.

Der Transport-Supervisor ist also, neben seinen eher koordinativen Aufgaben, dafür verantwortlich, die übrigen Funktionseinheiten des Transport-Subsystems von der Behandlung übergeordneter Ausnahmen zu entlasten. Um die dazu notwendige Stabilität und Performanz zu gewährleisten, ist auch der Transport-Supervisor als Hintergrundprozeß implementiert.

Wichtigste Aufgabe des Transport-Supervisors ist es, die erfolgreiche Durchführung von Bandsteuerungs-Aufträgen, das Einlegen von "Ehrenrunden" und die Betätigung des Bypass-Knopfes zu registrieren und adäquat zu behandeln. Dazu nutzt der Transport-Supervisor nicht nur die Informationen über alle sich derzeit im System befindlichen Transport-Aufträge, sondern schließt, anhand der von den Zuordnungspunkten und der Bandsteuerung eintreffenden Nachrichten, auch auf den externen Kontext des Transport-Subsystems, etwa auf das Vorhandensein mechanischer Unterbrechungen.

Meldet beispielsweise der Automatische-Zuordnungspunkt mehrfach die Zuordnung eines bestimmten Probenträgers zu einem bestimmten Probenröhrchen und ist der entsprechende Bandsteuerungs-Auftrag nicht von der Bandsteuerung als erfolgreich durchgeführt gemeldet worden, so läßt diese Konstellation auf eine durch das verzögerte Schalten von Weichen verursachte "Ehrenrunde" des Probenträgers schließen. In diesem Fall ist keine Neuverteilung des Probenträgers auf ein anderes Analysegerät erforderlich.

Findet dagegen eine solche Mehrfachzuordnung statt, nachdem die Bandsteuerung den entsprechenden Bandsteuerungs-Auftrag als erledigt gemeldet hat, so ist auf eine Betätigung des Bypass-Knopfes zu schließen und eine entsprechende Neuverteilung der Probe vorzunehmen. Diese ist mit einer Erweiterung des Auftragsnetzes um einen erneuten Bandsteuerungs-Auftrag verbunden, da der "alte" Auftrag von der Bandsteuerung als erledigt betrachtet wird.

Das Zusammenspiel von Auftragssteuerung, Transport-Supervisor, Automatischem-Zuordnungspunkt und Bandsteuerung bei der Behandlung übergeordneter Ausnahmen soll abschließend durch Abbildung 4-6 verdeutlicht werden. Die Darstellung ist dabei aus Gründen der Nachvollziehbarkeit stark vereinfacht und deckt nicht alle Aspekte der Zusammenarbeit innerhalb des Transport-Subsystems ab. Als Technik zur Darstellung des Nachrichtenaustauschs zwischen den beteiligten Komponenten dient ein sogenanntes Prozeß-Interaktions-Diagramm (PID)[11].

[11]Eine Erläuterung der Semantik von Prozeß-Interaktions-Diagrammen findet sich im Anhang dieser Arbeit.

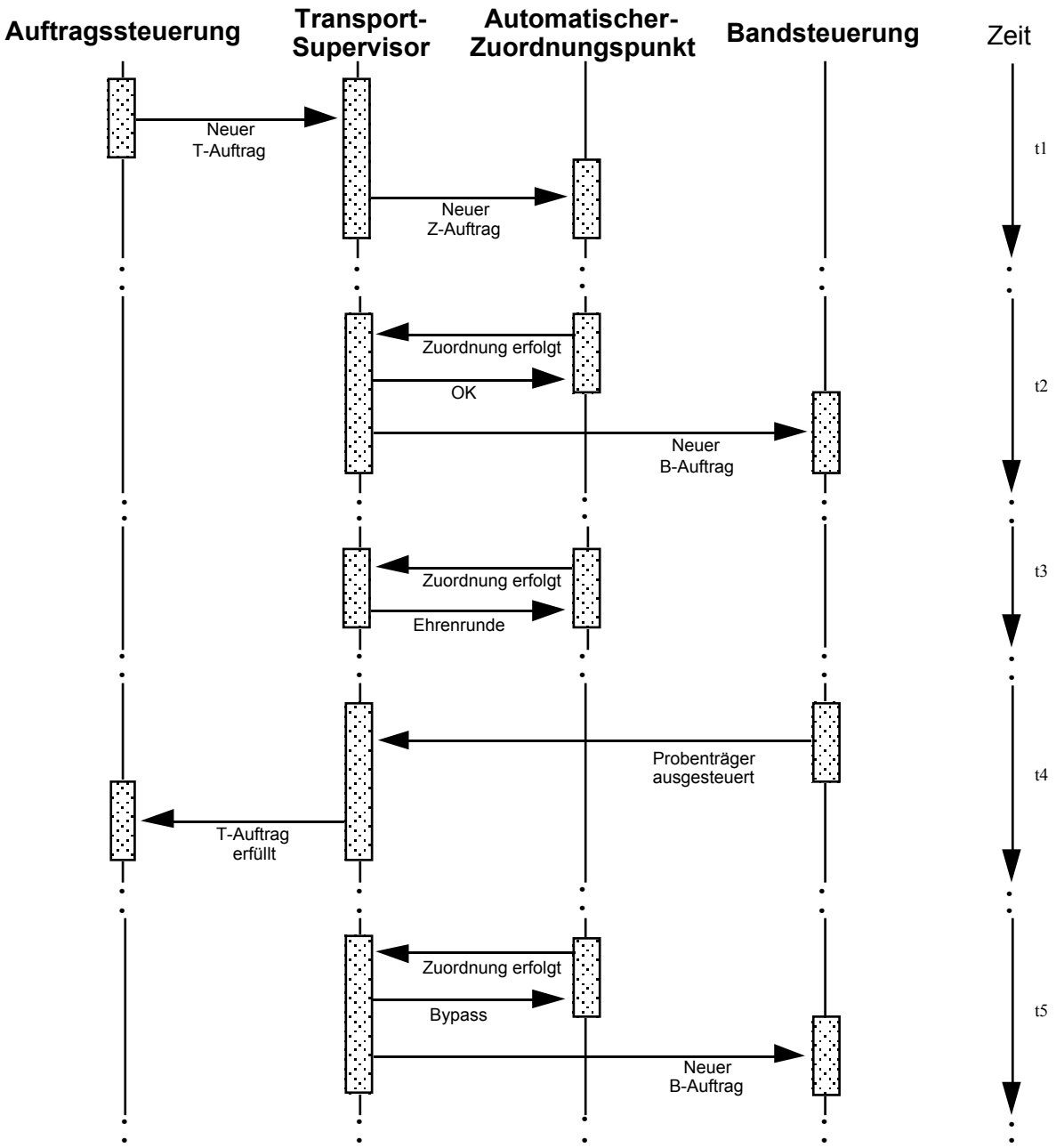

Abb. 4-6: PID für das Zusammenspiel von Auftragssteuerung und Transport-Subsystem

In den beiden ersten Zeitintervallen des Diagramms (t1, t2) ist die zeitliche Abfolge des Nachrichtenaustauschs im Falle eines "normalen" Ablaufs von Zuordnung und Transport aufgetragen. Der Transport-Supervisor in seiner Funktion als Schnittstelle zwischen Transport-Subsystem und XLab-Gesamtsystem erhält hier (t1) zunächst von der Auftragssteuerung die Nachricht, daß ein neuer Transport-Auftrag für ein bestimmtes Probenröhrchen vorliegt ("*Neuer T-Auftrag*"). Als Folge davon erzeugt er einen neuen Zuordnungs-Auftrag und meldet diesen an den Automatischen Zuordnungspunkt ("*Neuer Z-Auftrag*"). Dieser rekonstruiert den gemeldeten Zuordnungs-Auftrag aus der Datenbank und fügt diesen seinem Auftragsbestand hinzu.

In einem späteren Zeitintervall (t2) passiert das betreffende Probenröhrchen auf einem beliebigen Probenträger die Barcode-Leseeinheit des Automatischen-Zuordnungspunktes. Barcode und Tagcode werden gleichzeitig gelesen, so daß eine erfolgreiche Zuordnung durchgeführt werden kann. Diese wird an den Transport-Supervisor gemeldet ("*Zuordnung erfolgt*"). Der Transport-Supervisor kann keine Ausnahmebedingung erkennen und teilt diesen Umstand dem Automatischen-Zuordnungspunkt mit ("OK"). Dieser kann nun seinen Auftragsbestand entsprechend aktualisieren. Anschließend erzeugt der Transport-Supervisor einen neuen Bandsteuerungs-Auftrag und meldet diesen an die Bandsteuerung ("*Neuer B-Auftrag*"). Die Bandsteuerung ist damit für die weitere Abwicklung des Probentransports, also die Ansteuerung der Weichen bis hin zur vorbestimmten Aussonderungsspur, verantwortlich.

In einem noch späteren Zeitintervall (t3) passiert das Probenröhrchen auf dem gleichen Probenträger erneut die Barcode-Leseeinheit des Automatischen-Zuordnungspunktes. Wieder wird eine erfolgreiche Zuordnung an den Transport-Supervisor gemeldet ("*Zuordnung erfolgt*"). Da zu diesem Zeitpunkt noch keine positive Nachricht über das erfolgreiche Aussteuern des Probenträgers von der Bandsteuerung eingetroffen ist, schließt der Transport-Supervisor auf eine "Ehrenrunde". Dieses wird dem Zuordnungspunkt mitgeteilt ("*Ehrenrunde*"), eine Neuverteilung des Probenträgers auf eine andere Aussonderungsspur ist in diesem Falle jedoch nicht erforderlich.

Konnte die Bandsteuerung den Probenträger bei einem erneuten Versuch aussteuern (t4), so teilt sie dieses dem Transport-Supervisor durch eine entsprechende Nachricht ("*Probenträger ausgesteuert*") mit. Der Transport-Supervisor reagiert darauf, indem er den dazugehörigen Transport-Auftrag als erfolgreich abgeschlossen betrachtet und an die Auftragssteuerung zurückgibt ("*T-Auftrag erledigt*").

Das unterste Intervall des Diagramms (t5) verdeutlicht nun, wie eine nachfolgend vom Laborpersonal vorgenommene mechanische Unterbrechung (Bypass) vom Transport-Supervisor entdeckt und adäquat behandelt wird. Diese Ausnahmebehandlung wird wiederum vom Zuordnungspunkt angestoßen, der eine erneute Zuordnung des betrachteten Probenträgers zu seinem Probenröhrchen meldet ("*Zuordnung erfolgt*"). Anders als im oben beschriebenen "Normalfall", kann der Transport-Supervisor nun anhand der ihm vorliegenden Historie der bereits erfolgten Zuordnungen und Transporte feststellen, daß es sich nicht nur um eine erneute Zuordnung eines bereits zugeordneten Probenträgers zu einem Probenröhrchen handelt, sondern daß auch der nachfolgende Bandsteuerungs-Auftrag zu einem früheren Zeitpunkt von der Bandsteuerung als erledigt gemeldet worden ist.

Anhand dieser Informationen schließt der Transport-Supervisor auf das Vorliegen einer mechanischen Unterbrechung. Er teilt diese Schlußfolgerung dem Automatischen-Zuordnungspunkt durch eine entsprechende Nachricht ("*Bypass*") mit. Anschließend nimmt er eine Neuverteilung des Probenträgers auf eine andere Aussonderungsspur vor und vergibt einen entsprechenden Auftrag an die Bandsteuerung ("*Neuer B-Auftrag*").

Damit ist die Ausnahmebehandlung abgeschlossen und der normale Ablauf kann wieder aufgenommen werden.

4.4.2 Grenzen

Innerhalb des Transport-Subsystems ist der Transport-Supervisor für die Entdeckung und Behandlung übergeordneter Ausnahmen verantwortlich. Dazu nutzt er nicht nur die ihm vorliegenden Informationen über alle sich derzeit im System befindlichen Transport-Aufträge, sondern schließt, anhand der von den Zuordnungspunkten und der Bandsteuerung eintreffenden Nachrichten, auch auf den externen Kontext des Transport-Subsystems, etwa auf das Vorhandensein mechanischer Unterbrechungen.

Nachfolgend sei nun noch einmal das Zusammenspiel der Funktionseinheiten des Transport-Subsystems bei der Entdeckung und Behandlung mechanischer Unterbrechungen, dem sogenannten Bypass, betrachtet (vgl. Abb. 4-6). Dabei fällt auf, daß die Ausnahmebehandlung durch den Automatischen-Zuordnungspunkt mit einer erneuten Zuordnung des vom Bypass betroffenen Probenröhrchens zu seinem Probenträger angestoßen wird. M.a.W. muß der betroffene Probenträger auf dem Transportband den Automatischen-Zuordnungspunkt passiert haben, damit die beschriebene Ausnahmebehandlung eingeleitet werden kann.

Legt man nun das nachfolgend beschriebene Szenario zugrunde, so zeigt es sich, daß die beschriebene Ausnahmebehandlung unter den dort gegebenen Voraussetzungen nicht eingeleitet werden kann. Daraus folgt aber wiederum, daß der Bypass innerhalb des Transport-Subsystems weder entdeckt noch behandelt wird und damit ein eindeutiges Versagen des Transport-Subsystems vorliegt.

Einem solchen Fehler-Szenario könnte dabei folgendes Auftragsnetz zugrundeliegen:

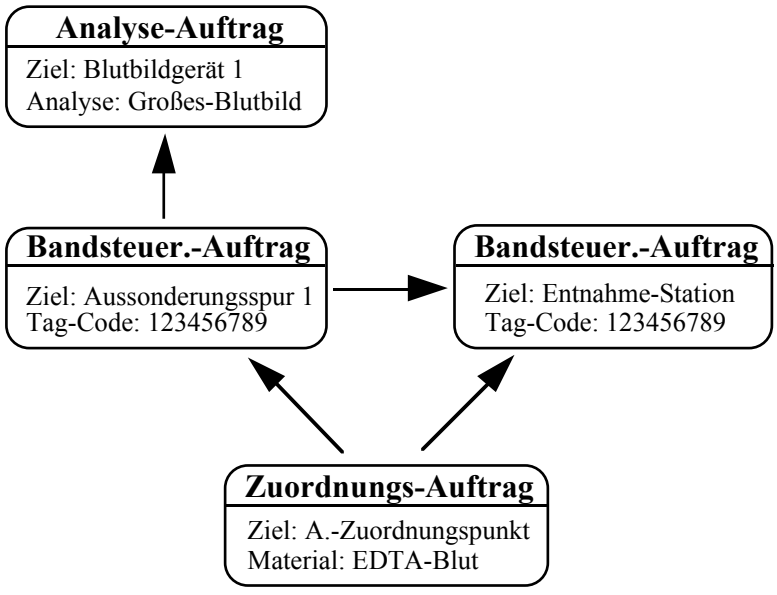

Abb. 4-3: Auftragsnetz für Fehler-Szenario

Zunächst sei davon ausgegangen, daß die Zuordnung von Probenträger zu Probenröhrchen erfolgreich durchgeführt werden konnte. Der Zuordnungs-Auftrag ist damit erfolgreich abgeschlossen.

Die Bandsteuerung erhält nun den ersten Bandsteuerungs-Auftrag für den Probenträger mit dem Tag-Code "123456789". Ziel ist in diesem Falle die "Aussonderungsspur 1". Konnte der

Transport aus Sicht der Bandsteuerung erfolgreich mit dem Aussteuern des Probenträgers auf die Aussonderungsspur abgeschlossen werden, so wird der zugehörige Bandsteuerungs-Auftrag als erledigt betrachtet und von der Bandsteuerung "vergessen". Nachfolgend erhält die Bandsteuerung von der Auftragssteuerung den nachfolgenden Bandsteuerungs-Auftrag. Dessen Ziel ist nunmehr die "Entnahme-Station".

Jetzt sei angenommen, daß noch bevor der dem Bandsteuerungs-Auftrag nachfolgende Analyse-Auftrag vom Blutbildgerät abgearbeitet werden konnte, eine mechanische Unterbrechung des automatisierten Ablaufes erfolgt. Dieses kann etwa durch eine Betätigung des Bypass-Knopfes seitens des Laborpersonals erfolgen. Als Folge davon werden alle sich bereits auf der Aussonderungsspur befindlichen Probenträger, also auch derjenige mit dem Tag-Code "123456789", ohne weitere Misch- oder Analysevorgänge durchzuführen direkt auf den eigentlichen Ring des Transportbandes zurückgesteuert.

Der Bandsteuerung liegt zu diesem Zeitpunkt aber nur noch der Bandsteuerungs-Auftrag mit dem Ziel "Entnahme-Station" vor. Daher wird sie folgerichtig den Probenträger mit dem Tag-Code "123456789" beim Wiedereinsteuern auf den Ring reidentifizieren, an der Entnahme-Station aussteuern und damit dem Ring des Transportbandes entnehmen.

Ein Blick auf Abbildung 4-3 zeigt nun aber, daß der Probenträger zum Zeitpunkt der Entnahme vom Transportband den Automatischen-Zuordnungspunkt noch nicht wieder passiert hat. M.a.W. wird die notwendige Vorbedingung für die Entdeckung und Behandlung der übergeordneten Ausnahme durch den Supervisor nicht erfüllt und das Transport-Subsystem versagt.

Dieses Fehler-Szenario ist kein konstruierter Extremfall. Ein ähnliches Szenario läßt sich beispielsweise problemlos unter der Annahme erstellen, daß mehrere Analysen mit derselben Probe nacheinander am Transportband durchgeführt werden sollen und zusätzlich kausale Abhängigkeiten zwischen diesen Analysen bestehen. Auch in einem solchen Fall würde das Vorliegen einer übergeordneten Ausnahme nicht rechtzeitig vor dem Aussteuern des Probenträgers auf die nächste Aussonderungsspur entdeckt und behandelt werden können, da die Vorbedingung der Ausnahmebehandlung, das Passieren des Automatischen-Zuordnungspunktes, nicht gegeben wäre.

Als Ergebnis dieses Abschnittes soll daher festgehalten werden:

- Das Transport-Subsystem stellt auf technischer Ebene gegenüber dem Konzept des Proben-Transporters eine deutliche Verbesserung dar. Insbesondere konnten Probleme der Stabilität und Performanz durch eine Aufteilung der Aufgaben des Proben-Transporters auf mehrere Funktionseinheiten behoben werden, die zudem teilweise als Hintergrundprozesse implementiert sind.
- Auch das Transport-Subsystem weist konzeptionelle Grenzen auf. Diese liegen, wie beim Proben-Transporter, in der adäquaten Behandlung übergeordneter Ausnahmen begründet. So läßt sich beispielsweise zeigen, daß die Betätigung des Bypass-Knopfes unter bestimmten Voraussetzungen zu einem Versagen des Transport-Subsystems führen kann.

5 Modellierung verteilter, technisch eingebetteter Anwendungssysteme

Im vorangegangenen Kapitel wurde mit dem automatisierten Transport von Proben zum Arbeitsplatz oder Analyseautomaten ein ausgewähltes Problem innerhalb des Laborprojektes herausgegriffen und mehrere Lösungsansätze vorgestellt. Dabei konnte auch für die zuletzt vorgestellte Lösung, das sogenannte Transport-Subsystem, gezeigt werden, daß diese zwar technische Schwächen behebt, die konzeptionellen Probleme aber nach wie vor bestehen bleiben. Das Transport-Subsystem bietet allerdings Ansatzpunkte, um vom konkreten Problem zu abstrahieren und die konzeptionellen Unterschiede zwischen der Modellierung interaktiver und der Modellierung technisch eingebetteter Anwendungssysteme herauszuarbeiten.

In diesem Zusammenhang wurde bereits zu Beginn dieser Arbeit die These formuliert, daß die in der Umgebung eines technisch eingebetteten Anwendungssystems stattfindenden Arbeitsprozesse und Ereignisse, im Vergleich zur Modellierung interaktiver Anwendungssysteme, die Bildung zusätzlicher Modelle bei Entwurf und Konstruktion eines solchen Systems erfordern. Anliegen dieses Kapitels ist es daher, basierend auf einer Analyse des Fehler-Szenarios aus Abschnitt 4.4.2, diese zusätzlichen Modelle argumentativ zu entwickeln und damit die vorgebrachte These zu bestätigen.

5.1 Analyse des Fehler-Szenarios

Das im vorangegangenen Kapitel beschriebene Fehler-Szenario mag auf den ersten Blick konstruiert erscheinen und ganz offensichtlich in den konkreten Gegebenheiten der im Anwendungsbereich von XLab vorgefundenen Hardwarekomponenten begründet liegen. Anliegen dieses Abschnittes ist es zu zeigen, daß das hier aufgetretene Problem grundlegender Natur ist. Um diesen Gedanken zu motivieren, wird zunächst unter Rückgriff auf das zugrundeliegende Transport-Subsystem vom konkreten Fehlerfall abstrahiert.

Das Fehler-Szenario basierte auf der Beobachtung, daß die für den Fall eines Bypass vorgesehene Ausnahmebehandlung eine erneute Zuordnung des vom Bypass betroffenen Probenröhrchens zu seinem Probenträger am Automatischen-Zuordnungspunkt voraussetzt. Diese Voraussetzung kann aber im Falle eines Bypass mit nachfolgender Entnahme des Probenröhrchens an der Entnahme-Station niemals erfüllt sein. Der Transport-Supervisor muß daher von einem korrekten Verhalten des Transport-Subsystems ausgehen und wird folgerichtig die Ausnahme nicht behandeln.

Das Versagen des verteilten Transport-Subsystems resultiert also aus dem Umstand, daß die in der Supervisor-Komponente implementierte Ausnahmebehandlung niemals bzw. zu spät angestoßen wird. Als Folge wird die Supervisor-Komponente nicht rechtzeitig mit hinreichenden Informationen versorgt, um das Auftreten einer übergeordneten Ausnahme zu erkennen.

Grund für diesen Mangel an zeitnahen Informationen ist, daß das steuernde Anwendungssystem nicht unmittelbar nach der Betätigung des Bypass-Knopfes vom Vorliegen einer mechanischen Unterbrechung unterrichtet wird. Dieses wiederum resultiert

aus den konkreten Gegebenheiten der im Anwendungsbereich von XLab vorgefundenen Hardwareumgebung.

Abstrahiert man von der konkreten Hardwareumgebung bzw. dem konkreten Transport-Subsystem auf technisch eingebettete Anwendungssysteme im Allgemeinen, so kann der oben aufgezeigte Mangel an zeitnahen Informationen auch als Synchronisationsproblem zwischen den beiden nachfolgend beschriebenen Zustandsmodellen interpretiert werden.

Da ist auf der einen Seite das sogenannte *Anwendungsmodell*. Es umfaßt den durch Werkzeuge, Automaten und Materialien geprägten Zustand der einzelnen Funktionseinheiten eines nach WAM entworfenen, verteilten Anwendungssystems. Dieser Zustand stellt in zweierlei Hinsicht eine Abstraktion dar. In fachlicher Hinsicht determiniert der Zustand von Werkzeugen, Automaten und Materialien die Menge der möglichen Umgangsformen mit dem Anwendungssystem. In technischer Hinsicht stellt dieser Zustand eine Abstraktion von der konkreten softwaretechnischen Repräsentation der Werkzeuge, Automaten und Materialien der Anwendung zur Laufzeit dar.

Wichtig in diesem Zusammenhang ist es festzuhalten, daß ein solchermaßen definiertes Anwendungsmodell keine spezielle Eigenschaft technisch eingebetteter Anwendungssysteme darstellt, sondern sich gleichfalls auch in nach WAM entworfenen, interaktiven Anwendungssystemen wiederfindet.

Das zweite Zustandsmodell, das beim Entwurf technisch eingebetteter Anwendungssysteme eine Rolle spielt, wurde bislang nur implizit verwendet. Es wird durch die fachlich abstrahierende Sicht des Softwareentwicklers auf die technische Konfiguration der einbettenden Hardware gebildet und beschreibt damit, auf der fachlichen Ebene, den Zustand der Umgebung des Anwendungssystems. Anders ausgedrückt umfaßt es genau diejenigen Aspekte der technischen Umgebung, die dem Softwareentwickler für Entwurf und Konstruktion des Anwendungssystems relevant erscheinen. Auf technischer Ebene handelt es sich dabei vor allem um die verfügbaren Schnittstellen zwischen Hard- und Software, sowie um die über diese Schnittstellen registrierbaren Ereignisse und Zustandsveränderungen der technischen Umgebung. Abstrahiert wird i.d.R. von den technischen Einzelheiten der einbettenden Hardware, etwa deren elektromechanischer Realisierung.

In Abschnitt 2.4 wurde bei der Unterscheidung der Anwendungsbereiche interaktiver und technisch eingebetteter Anwendungssysteme festgehalten, daß letztere dem Betrieb und der Überwachung technischer Anlagen dienen. Damit wird deutlich, daß das oben beschriebene Zustandsmodell der technischen Umgebung genau den Zustand einer solchen Anlage abbildet. Es soll daher im weiteren als *Anlagenmodell* bezeichnet werden.

Anlagen- und Anwendungsmodell existieren nicht unabhängig voneinander, sondern bedingen sich zur Laufzeit wechselseitig. So können innerhalb des Anlagenmodells stattfindende Ereignisse dazu führen, daß geeignete Reaktionen innerhalb des Anwendungsmodells ausgelöst werden müssen. Umgekehrt können Veränderungen innerhalb des Anwendungsmodells entsprechende Rückwirkungen auf das Anlagenmodell des Anwendungssystems ausüben. Ein Beispiel aus dem Kontext des Transport-Subsystems soll diesen Zusammenhang verdeutlichen:

> Passiert ein Probenträger den Automatischen-Zuordnungspunkt, so ist dieses Ereignis als Veränderung innerhalb des Anlagenmodells interpretierbar. Als

Konsequenz wird die innerhalb des Anwendungsmodells implementierte Reaktion auf eine solche Zuordnung ausgelöst. Diese Reaktion beinhaltet mit der auslastungsabhängigen Verteilung der Probe auf einen Arbeitsplatz oder ein Analysegerät eine Veränderung des Anwendungsmodells, die letzlich die Aussteuerung des Probenträgers an einer Weiche zur Folge hat. Eine solche Aussteuerung stellt aber wiederum ein Ereignis innerhalb des Anlagenmodells dar, so daß eine wechselseitige Abhängigkeit beider Modelle gegeben ist.

Festzuhalten bleibt also, daß die beiden Zustandsmodelle nicht unabhängig voneinander betrachtet werden können, sondern sich statt dessen zur Laufzeit mit wechselseitigen Abhängigkeiten bedingen. Daraus wiederum folgt, daß geeignete Mechanismen vorgesehen werden müssen, um die beiden Modelle miteinander zu synchronisieren.

Sind solche Mechanismen nicht in ausreichender Menge vorhanden, so können sich die beiden Modelle zur Laufzeit auseinander entwickeln. Damit aber ist die Gefahr verbunden, daß sich das scheinbar korrekte Verhalten des Anwendungsmodells, bezogen auf den tatsächlichen Zustand des Anlagenmodells, als fehlerhaft erweist. Eine solche Inkonsistenz zwischen Anlagen- und Anwendungsmodell könnte beispielsweise entstehen, wenn das Anwendungsmodell relevante Zustandsveränderungen des Anlagenmodells nicht nachvollziehen bzw. das Anwendungsmodell eigene Zustandsveränderungen nicht rechtzeitig an das Anlagenmodell weitergeben kann.

Mit Hilfe der in diesem Abschnitt vorgenommenen Abstraktion über Anlagen- und Anwendungsmodell technisch eingebetteter Anwendungssysteme läßt sich nun auch das eingangs beschriebene Versagen des Transport-Subsystems als Synchronisationsproblem zwischen den beiden Modellen erklären:

Die Betätigung des Bypass-Knopfes ist als mechanische Unterbrechung eine Veränderung innerhalb des Anlagenmodells. Der Transport-Supervisor wird als Teil des Anwendungsmodells nicht unmittelbar von dieser Veränderung unterrichtet und leitet folgerichtig die vorgesehene Ausnahmebehandlung nicht ein. Damit ist das aus Anlagen- und Anwendungsmodell gebildete Gesamtmodell des Transport-Subsystems vom Zeitpunkt der Betätigung des Bypass-Knopfes bis zu einer erneuten Resynchronisation am Automatischen-Zuordnungspunkt nicht mehr konsistent, da eine relevante Veränderung des Anlagenmodells innerhalb des Anwendungsmodells nicht nachvollzogen werden konnte. Diese Inkonsistenz hat zur Konsequenz, daß ein für sich genommen korrektes Verhalten des Anwendungsmodells (keine Ausnahmebehandlung eingeleitet) im Zusammenspiel mit bestimmten Zuständen seines Anlagenmodells (Bypass mit nachfolgender Probenentnahme) zum Fehlverhalten des Gesamtmodells führen kann.

5.2 Anlagen- und Anwendungsmodell

Offen geblieben ist bei der Analyse des Fehler-Szenarios die Frage, ob und wie Inkonsistenzen zwischen Anlagen- und Anwendungsmodell technisch eingebetteter Anwendungssysteme vermieden oder zumindest softwaretechnisch behandelt werden können.

Als Einstieg in die Beantwortung der Frage sollen zunächst die im vorangegangenen Abschnitt eingeführten Begriffe Anlagen- bzw. Anwendungsmodell präzisiert werden.

Motiviert wurde das Anlagenmodell mit der Erkenntnis, daß beim Entwurf technisch eingebetteter Anwendungssysteme auch die einbettende Umgebung, respektive das Verhalten der zu betreibenden Anlage, modelliert werden muß. Dieses zusätzliche Modell bzw. die damit verbundene Modellierung zusätzlicher Ereignisquellen unterscheidet technisch eingebettete von interaktiven Anwendungssystemen.

Damit soll allerdings nicht impliziert werden, daß interaktive Anwendungssysteme kein Modell der einbettenden Umgebung besitzen. Im Unterschied zum Anlagenmodell technisch eingebetteter Anwendungssysteme ist es aber auf die Modellierung einer einzigen Ereignisquelle, nämlich die der Benutzungsschnittstelle des Anwendungssystems, beschränkt. Diese Beschränkung zeigt sich darin, daß weder Zustandsveränderungen der einbettenden Umgebung modelliert, noch entsprechenden Reaktionen innerhalb des interaktiven Anwendungssystems implementiert werden. Statt dessen werden Zustandsveränderungen durch den Benutzer beobachtet, fachlich interpretiert und erst dann über die Benutzungsschnittstelle an das interaktive Anwendungssystem weitergegeben.

Bei technisch eingebetteten Anwendungssystemen ist es dagegen weder möglich noch sinnvoll, alle in der zu betreibenden Anlage stattfindenden Ereignisse bzw. die dadurch ausgelösten Zustandsübergänge durch einen Benutzer interpretieren und über die Benutzungsschnittstelle an das Anwendungssystem weitergeben zu lassen. Grund dafür ist zum einen die große Anzahl an Ereignissen, die gleichzeitig und auf unterschiedlichen technischen Ebenen innerhalb des Systems stattfinden und damit das menschliche Wahrnehmungs- und Reaktionsvermögen weit übersteigen. Zum anderen ist es ja gerade die Grundidee technisch eingebetteter Anwendungssysteme, den Menschen von automatisierbaren Arbeitsabläufen, also etwa dem Betrieb von Anlagen, zu entlasten.

In diesem Sinne wäre es beispielsweise schwer vorstellbar, das in dieser Arbeit thematisierte Transportband interaktiv zu betreiben. Anderenfalls müßte der menschliche Bediener des Transportbandes in die Lage versetzt werden, permanent die Position aller Probenträger auf dem Transportband zu beobachten, diese bezüglich des noch zurückzulegenden Weges zu interpretieren und die entsprechenden Weichenstellungen an das interaktive Transportsystem weiterzugeben. Angesichts der großen Anzahl von Probenträgern würde dieses nicht nur die kognitiven Fähigkeiten des Menschen weit übersteigen, sondern überdies die Verwendung technischer Anlagen zum automatisierten Transport von Proben grundsätzlich in Frage stellen.

Damit erweist es sich für technisch eingebettete Anwendungssysteme als notwendig, innerhalb der zu betreibenden Anlage stattfindende Ereignisse bzw. Zustandsübergänge vom Anwendungssystem selbst behandeln zu lassen. Diese Behandlung erfolgt, indem die Ereignisse vom Anwendungssystem registriert, interpretiert und in implementierte Aktionen umgesetzt werden. Voraussetzung einer solchen implementierten Reaktion ist allerdings, daß zuvor überhaupt ein Modell davon existiert, welche Ereignisse und Zustandsübergänge innerhalb der Anlage auftreten können und welche davon für das Anwendungssystem relevant sind. Genau dieses Modell wurde im vorangegangenen Abschnitt als das Anlagenmodell technisch eingebetteter Anwendungssysteme bezeichnet.

Es ist von entscheidender Bedeutung für diese Arbeit und soll daher noch einmal anhand von Begriff 5-1 festgemacht werden:

Begriff 5-1: Das Anlagenmodell technisch eingebetteter Anwendungssysteme

Das Anlagenmodell technisch eingebetteter Anwendungssysteme wird durch die abstrahierende Sicht des Softwareentwicklers auf die Konfiguration der einbettenden Hardware gebildet. In der fachlichen Interpretation bildet das Anlagenmodell den Zustand der einbettenden technischen Umgebung des Anwendungssystems, also den Zustand der zu betreibenden Anlage, ab. In der technischen Interpretation modelliert es die verfügbaren Schnittstellen zwischen Hard- und Software, sowie die über diese Schnittstellen registrierbaren Ereignisse und Zustandsveränderungen der Anlage. Das Anlagenmodell in seiner Funktion als Umgebungsmodell technisch eingebetteter Anwendungssysteme ist komplexer als das Umgebungsmodell interaktiver Anwendungssysteme, weil es innerhalb der einbettenden technischen Umgebung stattfindende Ereignisse und Zustandsveränderungen abbilden muß.

Die Existenz eines Anlagenmodells stellt also offensichtlich eine spezielle Eigenschaft technisch eingebetteter Anwendungssysteme dar. Interessant ist es nun zu untersuchen, welchen Einfluß die Existenz eines Anlagenmodells auf die Komplexität des zugehörigen Anwendungsmodells hat. Für interaktive Anwendungssysteme, also Systeme mit einem beschränkten Umgebungsmodell, gilt in diesem Zusammenhang, daß das zugehörige Anwendungsmodell "lediglich" Reaktionen auf Eingaben des Benutzers implementiert.

Für das Anwendungsmodell technisch eingebetteter Anwendungssysteme läßt sich dieses "einfache" Benutzungsmodell nicht aufrecht erhalten. Dort sind, neben den Eingaben des Benutzers, auch die innerhalb des Anlagenmodells abgebildeten Zustände, Zustandsübergänge und Ereignisse zu betrachten. Diese müssen vom Anwendungsmodell unmittelbar, d.h. ohne den Umweg über die Benutzungsschnittstelle, registriert, interpretiert und in implementierte Aktionen umgesetzt werden. In der Konsequenz erhöht sich dadurch die Komplexität des Anwendungsmodells, da nicht nur zusätzliche Ereignisquellen selbst, sondern auch die innerhalb des Anlagenmodells modellierten Zustände der Anlage in Software abgebildet werden müssen.

Die hier dargelegten, speziellen Eigenschaften des Anwendungsmodells technisch eingebetteter Anwendungssysteme werden nachfolgend anhand von Begriff 5-2 festgehalten. Im weiteren Verlauf dieser Arbeit wird sich allerdings zeigen, daß Synchronisationsprobleme zwischen Anlagen- und Anwendungsmodell eine erweiterte Interpretation des Anwendungsmodells erfordern. Aus diesem Grunde handelt es sich bei Begriff 5-2 nur um eine vorläufige Fassung.

Begriff 5-2: Das Anwendungsmodell technisch eingebetteter Anwendungssysteme (vorläufige Fassung)

Das Anwendungsmodell technisch eingebetteter Anwendungs-Systeme wird von den aus dem Entwurf und der Konstruktion interaktiver Anwendungssysteme

bekannten Software-Arbeitsmitteln und -gegenständen gebildet. Es modelliert den Zustand der Werkzeuge, Automaten und Materialien einer Anwendung zur Laufzeit. In der fachlichen Interpretation determiniert dieses Zustandsmodell die Menge der möglichen Umgangsformen mit dem Anwendungssystem. In der technischen Interpretation erlaubt es, von der konkreten softwaretechnischen Repräsentation der Werkzeuge, Automaten und Materialien zur Laufzeit zu abstrahieren. Das Anwendungsmodell technisch eingebetteter Anwendungssysteme ist komplexer als dasjenige interaktiver Anwendungssysteme, weil die innerhalb seines Anlagenmodells abgebildeten Zustände, Zustandsübergänge und Ereignisse unmittelbar und ohne den Umweg über die Benutzungsschnittstelle registriert, interpretiert und in implementierte Aktionen umgesetzt werden müssen.

5.3 Sensoren und Aktuatoren

Mit der Definition der Begriffe 5-1 und 5-2 sind im vorangegangenen Abschnitt die Rollen geklärt worden, die Anlage- und Anwendungsmodell innerhalb technisch eingebetteter Anwendungssysteme spielen. Offen geblieben ist dabei die Frage, wie das Zusammenspiel der beiden Modelle zur Laufzeit konkret aussehen kann bzw. welchen Einfluß die jeweilige Anlage auf die Gestaltung des Zusammenspiels hat.

Vom Anwendungsmodell wurde in diesem Zusammenhang lediglich gefordert, die innerhalb des Anlagenmodells abgebildeten Zustände, Zustandsübergänge und Ereignisse unmittelbar und ohne den Umweg über die Benutzungsschnittstelle zu registrieren, zu interpretieren und nachfolgend in implementierte Aktionen umzusetzen. Voraussetzung dafür ist natürlich in erster Linie, daß entsprechende Schnittstellen zwischen der Hardware, die die zu betreibende Anlage des Anwendungssystems bildet, und der Software des Anwendungssystems vorhanden sind.

Schnittstellen zwischen Hard- und Software lassen sich grundsätzlich in zwei Kategorien aufteilen. *Sensoren* dienen dazu, in der Anlage des Anwendungssystems stattfindende Ereignisse zu registrieren und an die Software weiterzugeben. Umgekehrt ermöglichen *Aktuatoren* es dem Anwendungssystem, seinerseits steuernd auf die Hardware einzuwirken und damit Veränderungen am Zustand der Anlage zu veranlassen. Sensoren und Aktuatoren selbst können auf konzeptioneller Ebene wiederum durch jeweils zwei Komponenten modelliert werden (vgl. Abbildung 5-1).

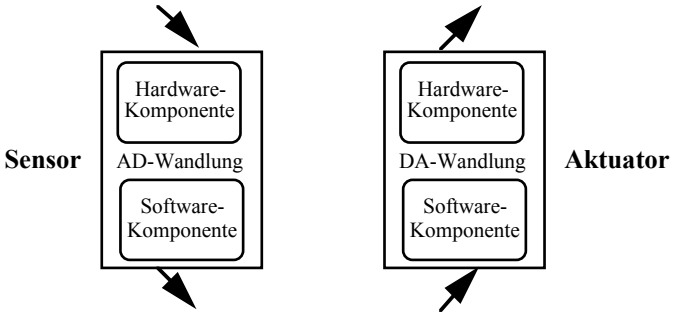

Abb. 5-1: Modellierung von Sensoren und Aktuatoren

Die *Hardwarekomponente eines Sensors* registriert Ereignisse, die innerhalb zu betreibenden Anlage des Anwendungssystems stattfinden. Dabei handelt es sich i.d.R. um analoge Eingangsgrößen wie Temperatur, elektromagnetische Veränderungen oder elektrische Spannung. Diese werden in digitale Ausgangssignale umgewandelt (A/D-Wandlung) und der *Softwarekomponente eines Sensors* als Programmereignis zur Verfügung gestellt. Auf technischer Ebene wird dieser Mechanismus meist über sogenannte *Betriebssystem-Interrupts* realisiert, die über entsprechende *Interrupt-Handler* in der Anwendungssoftware ausgewertet werden können.

Umgekehrt führt der Aufruf der *Softwarekomponente eines Aktuators* aus dem Programm heraus dazu, daß der zugehörigen *Hardwarekomponente eines Aktuators* ein digitales Eingangssignal übermittelt wird. Dieses Signal wird, je nach Aktuatortyp, in entsprechende analoge Ausgangsgrößen umgewandelt (D/A-Wandlung) und kann auf diese Weise den Zustand der einbettenden Anlage verändern.

Beispiele für Sensoren und Aktuatoren lassen sich innerhalb des Transport-Subsystems leicht finden. So stellt die Barcode-Leseeinheit des Automatischen-Zuordnungspunktes einen Sensor dar, der die beim Scannen eines Barcodes gewonnenen, analogen Eingangsgrößen in die digitale Repräsentation eines Barcodes umwandelt und der Software zur Verfügung stellt. Auch die bereits beschriebenen Tagcode-Leser sind Beispiele für Sensoren. Sie messen die elektromagnetischen Veränderungen, die ein am Tagcode-Leser vorbeifahrender Probenträger bzw. dessen Tagcode-Pille verursacht, und wandeln diese in die digitale Repräsentation des Tagcodes um. Diese ist dann wiederum von der Software ausgewertbar.

Die Weichen des Transportbandes sind dagegen Beispiele für Aktuatoren. Die Bandsteuerung des Transport-Subsystems berechnet zunächst für jeden Probenträger die notwendigen Weichenstellungen auf dem Weg zum ausgewählten Arbeitsplatz oder Analyseautomaten. Die berechneten Weichenstellungen werden dann über entsprechende Methodenaufrufe an die Softwarekomponenten der Weichen, sogenannte *Weichen-Adapter*, übermittelt. Die Weichen-Adapter wandeln die Aufrufe in digitale Eingangssignale für sogenannte *Weichen-Controller* um, die auf dem hier verwendeten Abstraktionsniveau als Teil der Hardwarekomponente des Aktuators interpretiert werden können[12]. Die Weichen-Controller wiederum sind nun für die Umwandlung der digitalen Eingangssignale in die analoge Ausgangsgröße der Weiche, nämlich die mechanische Aussteuerung eines Probenträgers, zuständig.

Die konzeptionelle Aufteilung von Sensoren und Aktuatoren in jeweils eine Hard- und eine Softwarekomponente ermöglicht es, diese Komponenten nahtlos den beiden im vorangegangenen Abschnitt thematisierten Modellen technisch eingebetteter Anwendungssysteme zuzuordnen (vgl. Abbildung 5-2).

[12]Weichen-Controller können für sich genommen ebenfalls als technisch eingebettetes Anwendungssystem betrachtet werden, da auch sie Software beinhalten, die in eine technische Umgebung eingebettet wird. Die Entwicklung von Software für hardwarenahe Controller erfolgt allerdings i.d.R. auf einem ganz anderen Abstraktionsniveau als das hier thematisierte LIS und soll daher im Rahmen dieser Arbeit nicht weiter diskutiert werden.

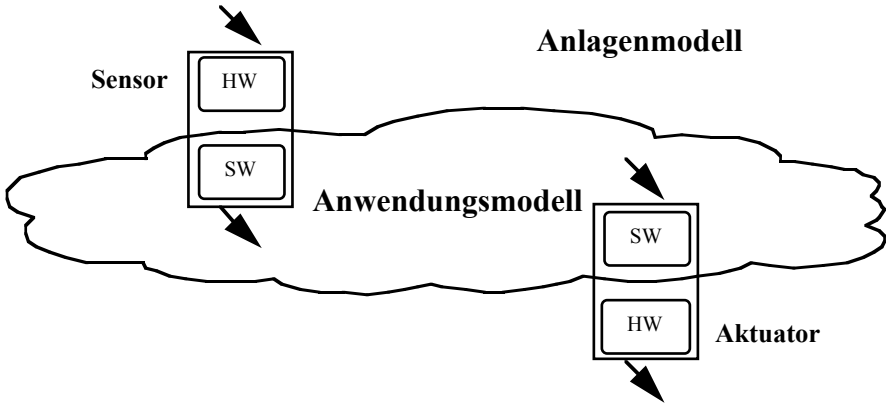

Abb. 5-2: Das Zusammenspiel von Anlagen- und Anwendungsmodell über Sensoren und Aktuatoren

Die jeweilige Hardware-Komponente ist dabei immer Teil des Anlagenmodells. Bei der Hardware-Komponente von Sensoren oder Aktuatoren handelt es sich in der Tat um die abstrahierende Sicht des Softwareentwicklers auf die Konfiguration der einbettenden Hardware (vgl. Begriff 5-1). So wird zum Beispiel bei der bereits erwähnten Barcode-Leseeinheit des Automatischen-Zuordnungspunktes vollständig davon abstrahiert, daß eine ganze Anzahl von Eingangsgrößen zur vollständigen Erkennung eines Barcodes ausgewertet und interpretiert werden müssen. Statt dessen ist für den Softwareentwickler lediglich das Gesamtergebnis des Lesevorgangs, also der "Wert" des gelesenen Barcodes, von Interesse.

Die Software-Komponente eines Sensors oder Aktuators ist Teil des Anwendungsmodells. Der Umgang mit der Software-Komponente eines Aktuators unterscheidet sich dabei nicht wesentlich vom Umgang mit anderen Komponenten des Anwendungssystems, da er i.d.R. auf einfache Methodenaufrufe beschränkt werden kann. Komplizierter ist dagegen der Umgang mit der Software-Komponente eines Sensors, da in der zu betreibenden Anlage stattfindende Ereignisse in den Kontrollfluß des Anwendungsmodells integriert werden müssen. Hier muß auf Ebene des Anwendungsmodells entschieden werden, ob ein solches Ereignis den "normalen" Kontrollfluß der Anwendung priorisiert unterbrechen darf (*Interrupt-Ansatz*), oder ob ein solches Ereignis in den allgemeinen Ereignis-Mechanismus reaktiver Anwendungssysteme integriert werden soll (*Event-Loop-Ansatz*).

Die in diesem Abschnitt herausgearbeiteten Eigenschaften von Sensoren und Aktuatoren sollen abschließend anhand der Begriffe 5-3 und 5-4 festgehalten werden. Auch bei Sensoren und Aktuatoren wird die weitere Argumentation dieser Arbeit eine erweiterte Interpretation notwendig machen. Daher haben auch die Begriffe 5-3 und 5-4 vorläufigen Charakter.

Begriff 5-3: Sensor (vorläufige Fassung)

Ein Sensor ist eine Schnittstelle zwischen der Hardware und der Software eines technisch eingebetteten Anwendungssystems. Er dient dazu, in der vom Anwendungssystem zu betreibenden Anlage stattfindende Ereignisse zu registrieren und an die Software weiterzugeben. Sensoren können auf konzeptioneller Ebene durch zwei Komponenten modelliert werden. Die

Hardware-Komponente eines Sensors registriert Ereignisse, die innerhalb der Anlage stattfinden. Diese lassen sich als analoge Eingangsgrößen interpretieren, die in digitale Ausgangssignale umgewandelt (A/D-Wandlung) und der zugehörigen Software-Komponente des Sensors als Programmereignis zur Verfügung gestellt werden. Die Hardware-Komponente eines Sensors ist Teil des Anlagenmodells, die Software-Komponente dagegen Teil des Anwendungsmodells technisch eingebetteter Anwendungssysteme.

Begriff 5-4: Aktuator (vorläufige Fassung)

Ein Aktuator ist eine Schnittstelle zwischen der Hardware und der Software eines technisch eingebetteten Anwendungssystems. Er ermöglicht es dem Anwendungssystem, steuernd auf die Hardware einzuwirken und damit Veränderungen am Zustand der zu betreibenden Anlage zu veranlassen. Aktuatoren können auf konzeptioneller Ebene durch zwei Komponenten modelliert werden. Die Software-Komponente eines Aktuators wird aus dem Programm heraus aufgerufen und übermittelt der zugehörigen Hardware-Komponente des Aktuators ein digitales Eingangssignal. Dieses wird von der Hardware-Komponente in entsprechende analoge Ausgangsgrößen umgewandelt (D/A-Wandlung) und kann auf diese Weise den Zustand der einbettenden Anlage verändern. Die Hardware-Komponente eines Aktuators ist Teil des Anlagenmodells, die Software-Komponente dagegen Teil des Anwendungsmodells technisch eingebetteter Anwendungssysteme.

5.4 Steuerungsmodell

Im vorangegangenen Abschnitt wurde gezeigt, wie das Zusammenspiel von Anlagen- und Anwendungsmodell technisch eingebetteter Anwendungssysteme konzeptionell über Sensoren bzw. Aktuatoren modelliert werden kann. Dieser Abschnitt wird sich mit der Frage beschäftigen, ob und wie Inkonsistenzen zwischen Anlagen- und Anwendungsmodell vermieden oder zumindest softwaretechnisch behandelt werden können.

Aus der Analyse des zugrundegelegten Fehler-Szenarios ging hervor, daß das Versagen des Transport-Subsystems als Synchronisationsproblem zwischen Anlagen- und Anwendungsmodell verstanden werden kann (vgl. Abschnitt 5.1). Als zentrales Problem erweist sich dabei, daß das aus Anlagen- und Anwendungsmodell gebildete Gesamtmodell des Transport-Subsystems vom Zeitpunkt der Betätigung des Bypass-Knopfes bis zu einer erneuten Resynchronisation am Automatischen-Zuordnungspunkt nicht mehr konsistent ist.

Anders ausgedrückt kann eine relevante Veränderung des Anlagenmodells, hier also die Betätigung des Bypass-Knopfes, vom Anwendungsmodell bis zur Resynchronisation am Automatischen-Zuordnungspunkt nicht nachvollzogen werden. Als Folge dieser Inkonsistenz lassen sich Fälle konstruieren, bei denen ein an sich korrektes Verhalten des Anwendungsmodells im Zusammenspiel mit bestimmten Zuständen seines Anlagenmodells in einem Fehlverhalten des Gesamtmodells resultiert.

Entscheidende Ursache für dieses Fehlverhalten ist nicht nur der Umstand, daß überhaupt eine Inkonsistenz zwischen den beiden Modellen vorhanden ist. Ebenso wichtig ist auch, daß das Anwendungsmodell über keinerlei Mittel verfügt, den Grad an Synchronizität zwischen dem angenommenen und dem tatsächlichen Zustand des Anlagenmodells abzubilden und adäquat zu behandeln. Es ist also nicht in der Lage zu modellieren, daß die ihm zur Verfügung stehenden Informationen über den Zustand der zu betreibenden Anlage unter Umständen bereits "veraltet" sind. Statt dessen wird vorausgesetzt, daß alle relevanten Veränderungen innerhalb des Anlagenmodells unmittelbar und ohne weitere Verzögerungen an das Anwendungsmodell weitergegeben werden. Anders ausgedrückt muß das Anwendungsmodell des Transport-Subsystems mangels anderer Möglichkeiten voraussetzen, daß ihm ein *synchrones Modell* der Anlage zur Verfügung steht.

Die Annahme eines synchronen Anlagenmodells ist aber für ein LIS aus einer Vielzahl von Gründen nicht zu realisieren.

So läßt sich argumentieren, daß ein wirklich synchrones Anlagenmodell schon aufgrund quantenmechanischer Gesetzmäßigkeiten niemals realisierbar ist. In diesem Zusammenhang besagt etwa die Heisenberg'sche Umschärferelation, daß die Durchführung einer Messung Rückwirkungen auf das Gemessene hat und damit wiederum das Ergebnis der Messung verfälscht (vgl. [HW90], S. 37-40).

Auch extrem feingranulare Programmiermodelle, beispielsweise das sogenannte *perfect synchrony*-Modell (vgl. [Hal93]), die die Annahme eines synchronen Anlagenmodells über garantierte Taktfrequenzen der Sensoren und die Auszählung von Maschinenzyklen realisieren, kommen für die Entwicklung "großer" Softwaresysteme nicht in Frage. Sie sind statt dessen eher zur Entwicklung "harter" Echtzeitsysteme, beispielsweise im Bereich der hardwarenahen Controller-Programmierung, geeignet[13].

Neben diesen eher grundsätzlichen Einwänden sprechen auch pragmatische Überlegungen gegen die Annahme eines synchronen Anlagenmodells. So wäre für das in dieser Arbeit thematisierte LIS zu fordern, daß zu jedem Zeitpunkt sowohl die Position jedes Probenröhrchens auf dem Transportband als auch der gesamte Zustandsraum der einbettenden Hardware, determiniert durch den Zustand von Transportband, Weichen, Mischern und Analyseautomaten, durch entsprechende Sensoren synchron an das Anwendungsmodell übermittelt werden müßte.

Gegen diese Erweiterung lassen sich mehrere Argumente anführen. So verursachen zusätzliche Sensoren aus Perspektive des Hardwareentwicklers auch zusätzlichen Bedarf an Platz und Gewicht, sowie einen nicht zu unterschätzenden Aufwand an Verkabelung. Ferner sind sie für einen nicht unerheblichen Anteil der Kosten einer Hardwarekomponente verantwortlich. Aus Sicht des Softwareentwicklers implizieren zusätzliche Sensoren eine höhere "Taktrate" von Informationen, die entsprechend zu verarbeiten sind und daher wiederum in erhöhtem Aufwand für die Software resultieren.

---

[13]Eine tiefergehende Diskussion der Argumente, die gegen Modelle wie *perfect synchrony* bei Entwicklung und Betrieb eines LIS sprechen, ist nicht Gegenstand dieser Arbeit. Auf konzeptioneller Ebene sind es vor allem ungeklärte Fragen der Berechenbarkeit und der Nebenläufigkeit, die derartige Ansätze noch für absehbare Zeit unlösbar erscheinen lassen. Auf technischer Ebene kommen Aspekte wie das *Multitasking* bzw. *Multithreading* moderner Betriebssysteme, sowie das *Caching* der heutigen Prozessoren erschwerend hinzu.

Festzuhalten bleibt also, daß im Anwendungsbereich eines LIS mit einer begrenzten Anzahl an Sensoren und Aktuatoren auszukommen sein wird. Daraus läßt sich nun aber unmittelbar folgern, daß das Anwendungsmodell technisch eingebetteter Anwendungssysteme lediglich über ein *asynchrones Modell* der zu betreibenden Anlage verfügen kann. Die oben postulierte Voraussetzung des Anwendungsmodells, ein synchrones Modell der Anlage, kann also nicht gewährleistet werden.

Für die Modellierung technisch eingebetteter Anwendungssysteme stellt sich nun die Frage, an welcher Stelle man ansetzen kann, um die offenbar zu streng gefaßte Voraussetzung des Anwendungsmodells aufzuweichen bzw. zu umgehen. Zwei Alternativen sollen an dieser Stelle diskutiert werden:

- Das Anwendungsmodell wird dahingehend erweitert, daß der Grad an Synchronizität mit dem Anlagenmodell modellierbar ist. Im Anwendungsmodell muß abgebildet werden, daß Informationen über den Zustand des Anlagenmodells entweder hinreichend "aktuell" oder bereits "veraltet" sind. Damit entfällt die Notwendigkeit eines synchronen Anlagenmodells.
- Alle Aspekte des Anwendungsmodells, die die zu betreibende Anlage sowie damit verbundene Synchronizitätsgesichtspunkte betreffen, werden in ein eigenes Modell ausgelagert. Dieses neue Modell wird "zwischen" Anlagen- und Anwendungsmodell angeordnet und hat die Aufgabe, dem Anwendungsmodell ein *synchronisiertes Bild* des Anlagenmodells zu präsentieren. Als synchronisiertes Bild eines Modells soll in diesem Zusammenhang verstanden werden, wenn sich das Modell aus der Perspektive des Betrachters scheinbar synchron verhält, dieses aber in Wirklichkeit nicht ist. Gelingt es dem neuen Modell ein solches synchronisiertes Bild zu präsentieren, so kann aus Perspektive des Anwendungsmodells die Voraussetzung eines synchronen Anlagenmodells aufrechterhalten werden.

Für das konkrete Problem des Transport-Subsystems wäre auf Basis der ersten Alternative die folgende Lösung denkbar:

> Das Anwendungsmodell des Transport-Subsystems modelliert explizit, daß eine Entnahme des Probenröhrchens vom Transportband nur zu bestimmten Zeitpunkten erfolgen darf. Liegen hinreichend "aktuelle" Informationen darüber vor, ob der Probentransport erfolgreich durchgeführt werden konnte oder ob dieser zuvor durch eine Betätigung des Bypass-Knopfes unterbrochen worden ist, so ist die Entnahme erlaubt. Diese Information steht bei der dem Szenario zugrundegelegten Hardwarekonfiguration allerdings nur am Automatischen-Zuordnungspunkt zur Verfügung, da dieser den einzigen Resynchronisationspunkt innerhalb des Transport-Subsystems darstellt. Die hier vorgestellte Modellierungsalternative würde daher in der Praxis dazu führen, daß das Anwendungsmodell des Transport-Subsystems die endgültige Entnahme des Probenröhrchens vom Transportband solange verweigert, bis der zugehörige Probenträger den Automatischen-Zuordnungspunkt erneut passiert hat.

Die erste Alternative stellt eine theoretisch praktikable Lösung des Problems dar. Sie weist allerdings bei näherer Betrachtung einige Schwächen auf, die nachfolgend erörtert werden sollen.

Ungünstig ist, daß alle Probenträger vor der endgültigen Entnahme des Probenröhrchens zwecks Resynchronisation von Anlagen- und Anwendungsmodell noch einmal den Automatischen-Zuordnungspunkt passiert haben müssen. Es wird sich allerdings im weiteren Verlauf dieser Arbeit zeigen, daß es sich hierbei um ein prinzipielles Problem der konkreten Hardwarekonfiguration des Transportbandes handelt, daß nicht durch softwaretechnische Modellierungsalternativen behoben werden kann, sondern statt dessen Veränderungen an der Hardwareumgebung erfordert.

Interessanter, weil auf technisch eingebettete Anwendungssysteme im Allgemeinen übertragbar, ist eine Betrachtung der zusätzlichen Komplexität, die dem Anwendungsmodell aus der ersten Modellierungsalternative erwächst. In diesem Zusammenhang wurde schon anhand von Begriff 5-2 festgehalten, daß das Anwendungsmodell technisch eingebetteter Anwendungssysteme komplexer als dasjenige interaktiver Anwendungssysteme ist, weil die innerhalb seines Anlagenmodells abgebildeten Zustände, Zustandsübergänge und Ereignisse unmittelbar und ohne den Umweg über die Benutzungsschnittstelle registriert, interpretiert und in implementierte Reaktionen umgesetzt werden müssen.

Muß nun zusätzlich auch noch der Grad an Synchronizität mit dem Anlagenmodell abgebildet werden, so beginnt sich die Frage abzuzeichnen, ob ein solches Modell bei hinreichender Komplexität von Anlagenmodell und Anwendungslogik überhaupt noch beherrschbar ist, oder ob nicht statt dessen eine Aufteilung in mehrere überschaubare Teil-Modelle angebracht wäre.

Das hier aufgezeigte Problem eines sehr komplexen Anwendungsmodells bzw. die Frage nach dessen Aufteilung in mehrere Teil-Modelle liefert damit einen ersten Hinweis darauf, daß offenbar schon bei der (vorläufigen) Definition des Anwendungsmodells in Abschnitt 5.2 verschiedene Abstraktionen in einem einzigen Modell vermengt worden sind, die bei näherer Betrachtung unterschiedliche Aspekte der Modellierung technisch eingebetteter Anwendungssysteme betreffen.

Da ist auf der einen Seite das Modell der Software-Arbeitsmittel und -gegenstände. Es modelliert den Zustand der Werkzeuge, Automaten und Materialien einer Anwendung zur Laufzeit. Wichtig für die weitere Argumentation ist, daß beim Entwurf der Software-Arbeitsmittel und -gegenstände eher selten eine isomorphe oder gar mimetische Abbildung stofflicher Arbeitsmittel und Arbeitsgegenstände stattfindet. Statt dessen ist es Aufgabe des fachlichen (Klassen-) Entwurfs festzulegen, wie die Arbeitsmittel- und Arbeitsgegenstände innerhalb des Anwendungssystems als Werkzeuge und Materialien vergegenständlicht werden sollen (vgl. Abschnitt 2.1). Zur Laufzeit des Anwendungssystems wird es der Interpretation des menschlichen Benutzers überlassen, den Bezug zwischen den Werkzeugen und Materialien der Anwendung und dem jeweiligen Arbeitszusammenhang herzustellen.

Besonders augenscheinlich wird dieser interpretative Aspekt des menschlichen Umgangs mit Software, wenn dadurch neue Formen der Handhabung von Arbeitsmitteln oder Arbeitsgegenständen ermöglicht werden. Modelliert man beispielsweise einen *Ordner* als Software-Material, so könnte dessen Inhalt durch ein *Auflister-Werkzeug* nach verschiedenen Kriterien sortiert und angezeigt werden. Dieser Umgang ist mit einem stofflichen Ordner so ohne weiteres nicht möglich. Es bleibt also dem Benutzer überlassen, die neue Handhabungsmöglichkeit eines Software-Ordners zu interpretieren und im jeweiligen Arbeitszusammenhang einzusetzen.

Auf der anderen Seite wurde aber auch das Zustandsmodell der zu betreibenden Anlage mit in das Anwendungsmodell aufgenommen. Motiviert wurde diese Erweiterung damit, daß Reaktionen auf Zustandsveränderungen der Anlage innerhalb des Anwendungsmodells technisch eingebetteter Anwendungssysteme implementiert werden müssen. Dazu ist zunächst eine mimetische oder zumindest isomorphe Abbildung der stofflich verfügbaren Schnittstellen zwischen Hard- und Software notwendig, die über das im vorangegangenen Abschnitt vorgestellte Konzept von Sensoren und Aktuatoren realisiert werden kann.

Vergleicht man diese Abbildung mit der oben beschriebenen Abbildung von Software-Arbeitsmitteln und -gegenständen, so fällt auf, daß der Zustand der Anlage bzw. das Verhalten der Sensoren und Aktuatoren nicht vom Benutzer des Softwaresystems interpretiert und in einen Arbeitszusammenhang gestellt wird, sondern von der Software selbst. Damit aber wird die Software in die Lage versetzt, durch die Implementation geeigneter Reaktionen auf Zustandsveränderungen der Anlage, ihrerseits steuernd und regelnd auf diese einzuwirken.

Zusammmenfassend läßt sich nunmehr feststellen, daß die erste der beiden vorgeschlagenen Modellierungsalternativen zwei konzeptionelle Schwächen aufweist:

- Die Komplexität des Anwendungsmodells wird durch die oben beschriebene Notwendigkeit, den Grad an Synchronizität mit dem Anlagenmodell abzubilden, wesentlich erhöht. Fraglich ist dabei vor allem, ob ein Modell dieser Komplexität überhaupt noch beherrschbar ist.
- Innerhalb des Anwendungsmodells werden zwei Abstraktionen miteinander vermengt, die zur Laufzeit auf unterschiedliche Weise interpretiert und umgesetzt werden. Das Modell der Software-Arbeitsmittel und -gegenstände wird durch den Benutzer umgesetzt, indem die zur Verfügung gestellten Software-Arbeitsmittel und -gegenstände interpretiert und in einen Arbeitszusammenhang gestellt werden. Das Modell der zu betreibenden Anlage wird dagegen von der Software selbst interpretiert und in implementierte Aktionen umgesetzt, die wiederum den Zustand der Anlage verändern oder sondieren. Eine solche Vermengung unterschiedlicher Abstraktionen in einem Modell kann erfahrungsgemäß zu Problemen bei dessen Entwurf und Konstruktion führen.

Beide Schwächen lassen darauf schließen, daß eine Aufteilung des bislang angenommenen Anwendungsmodells in zwei Teil-Modelle eine bessere Lösung des Problems darstellen könnte. Genau dieser Ansatz aber wurde mit der zweiten Modellierungsalternative verfolgt. Bei dieser wurde vorgeschlagen, alle Aspekte des Anwendungsmodells, die die zu betreibende Anlage sowie damit verbundene Synchronizitätsgesichtspunkte betreffen, in ein eigenes Modell auszulagern.

Damit wäre die folgende Lösung des Problems möglich:

> Zunächst werden die Softwarekomponenten der Sensoren und Aktuatoren des Transport-Subsystems in ein eigenes Modell ausgelagert. Dazu zählen neben den Softwarekomponenten von Barcode-Leseeinheit und Tagcode-Lesern auch die Weichen-Adapter. Dann werden alle Komponenten des Anwendungsmodells, die den Zustand der zu beschreibenden Anlage abbilden, in das neue Modell verschoben. Dabei handelt es sich u.a. um diejenigen Bestandteile des Transport-Supervisors, die für die Entdeckung von Bypass-Bedingungen verantwortlich sind

und dazu ein Modell des Zustands von Transportband und Probenröhrchen verwalten. Aufgabe des neuen Modells ist es, dem nunmehr reduzierten Anwendungsmodell ein synchronisiertes Bild des Anlagenmodells zu präsentieren. Dazu werden nur diejenigen Ereignisse und Zustandsänderungen des Anlagenmodells, die auf synchronen bzw. resynchronisierten Informationen beruhen, an das Anwendungsmodell weitergegeben. Im Falle des Transport-Subsystems bewirkt eine solche "Filterung" von Informationen beispielsweise, daß die erfolgreiche Durchführung eines Transport-Auftrages erst dann an das eigentliche Anwendungsmodell weitergegeben wird, wenn diese Information aufgrund einer Resynchronisation mit dem Anlagenmodell als bestätigt erachtet werden kann.

In Abschnitt 2.4 wurde bei der Unterscheidung der Anwendungsbereiche interaktiver und technisch eingebetteter Anwendungssysteme festgehalten, daß letztere dem Betrieb technischer Anlagen dienen. Weiter wurde festgestellt, daß es zum Betrieb einer technischen Anlage notwendig ist, diese steuern zu können.

Betrachtet man in diesem Zusammenhang die Rolle, die das hier vorgeschlagene Modell innerhalb des Gesamtmodells technisch eingebetteter Anwendungssysteme spielt, so wird deutlich, daß genau die Abbildung von Steuerungsmechanismen Gegenstand des Modells ist. Es soll daher im weiteren als das *Steuerungsmodell* technisch eingebetteter Anwendungssysteme bezeichnet werden.

Das Steuerungsmodell ermöglicht es damit, die oben postulierte Voraussetzung des Anwendungsmodells, ein synchrones Modell der Anlage, aufrechtzuerhalten. Zudem vermeidet es die konzeptionellen Schwächen der ersten Modellierungsalternative, da nicht nur die Komplexität des Anwendungsmodells reduziert, sondern auch eine modelltechnisch saubere Trennung von Aspekten der zu betreibenden Anlage und der eigentlichen Anwendungslogik vollzogen wird.

Das Konzept eines Steuerungsmodells ist von großer Bedeutung für die weitere Argumentation dieser Arbeit. Es soll daher anhand von Begriff 5-5 noch einmal auf den Punkt gebracht werden:

Begriff 5-5: Das Steuerungsmodell technisch eingebetteter Anwendungssysteme

Das Steuerungsmodell technisch eingebetteter Anwendungssysteme wird dazu verwendet, die Steuerung der zu betreibenden bzw. zu überwachenden Anlage abzubilden. In der fachlichen Interpretation modelliert es den Zustand der zu betreibenden Anlage, sowie die zur Verfügung stehenden Schnittstellen zwischen Hard- und Software. Es ermöglicht damit, Aspekte der einbettenden Umgebung von der eigentlichen, im Anwendungsmodell abgebildeten, Anwendungslogik zu separieren. In der technischen Interpretation hat das Steuerungsmodell die Aufgabe, dem Anwendungsmodell ein synchronisiertes Bild des Anlagenmodells zu präsentieren. Dazu wird es "zwischen" Anlagen- und Anwendungsmodell angeordnet und gibt nur diejenigen Ereignisse und Zustandsänderungen des Anlagenmodells, die auf synchronen bzw. resynchronisierten Informationen

beruhen, an das Anwendungsmodell weiter. Um seine Aufgabe erfüllen zu können, muß das Steuerungsmodell Zustandsveränderungen des Anlagenmodells registrieren, interpretieren und in implementierte Aktionen umsetzen. Das Steuerungsmodell kann teilweise durch mimetische Abbildung der verfügbaren Schnittstellen zwischen Hard- und Software gewonnen werden, indem man diese als Sensoren bzw. Aktuatoren modelliert.

Zusammenfassend betrachtet steht mit dem Steuerungsmodell ein Konzept für die Modellierung des Zusammenspiels von Anlagen- und Anwendungsmodell zur Verfügung. Es komplettiert damit den in dieser Arbeit vertretenen Ansatz zur Modellierung technisch eingebetteter Anwendungssysteme. Gleichzeitig bestätigt es die eingangs vorgebrachte These, daß die in der Umgebung eines technisch eingebetteten Anwendungssystems stattfindenden Arbeitsprozesse und Ereignisse, im Vergleich zur Modellierung interaktiver Anwendungssysteme, die Bildung zusätzlicher Modelle bei Entwurf und Konstruktion eines solchen Systems erfordern.

Bevor dieser Ansatz bzw. die zusätzlichen Modelle im nächsten Kapitel auf konstruktiver Ebene umgesetzt werden können, ist allerdings noch eine Überarbeitung der Begriffe Anwendungsmodell, Sensor und Aktuator erforderlich, da diese in den Abschnitten 5.2 und 5.3 lediglich in einer vorläufigen Fassung eingeführt worden waren. Ihre endgültige Fassung soll nun auch das Konzept des Steuerungsmodells wiederspiegeln.

Für das Anwendungsmodell technisch eingebetteter Anwendungssysteme gilt in diesem Zusammmenhang, daß es nicht, wie ursprünglich angenommen, grundsätzlich komplexer als dasjenige interaktiver Anwendungssysteme sein muß. Ausschlaggebend dafür ist, daß zusätzliche Komplexität, die durch die zu betreibende Anlage bzw. damit verbundene Synchronisationsgesichtspunkte verursacht wird, in das Steuerungsmodell verlagert wird. Als Folge kann das Anwendungsmodell technisch eingebetteter Anwendungssysteme, analog zu dem interaktiver Anwendungssysteme, unter der Annahme eines synchronen Anlagenmodells betrieben werden und sich auf die Implementation der eigentlichen Anwendungslogik beschränken.

Dieser Umstand wird durch die endgültige Fassung von Begriff 5-2 reflektiert:

Begriff 5-2: Das Anwendungsmodell technisch eingebetteter Anwendungssysteme (endgültige Fassung)

Das Anwendungsmodell technisch eingebetteter Anwendungs-Systeme wird von den aus dem Entwurf und der Konstruktion interaktiver Anwendungssysteme bekannten Software-Arbeitsmitteln und -gegenständen gebildet. Es modelliert den Zustand der Werkzeuge, Automaten und Materialien einer Anwendung zur Laufzeit. In der fachlichen Interpretation determiniert dieses Zustandsmodell die Menge der möglichen Umgangsformen mit dem Anwendungssystem. In der technischen Interpretation erlaubt es, von der konkreten softwaretechnischen Repräsentation der Werkzeuge, Automaten und Materialien zur Laufzeit zu abstrahieren. Das Anwendungsmodell erhält zur Laufzeit mit Hilfe des Steuerungsmodells Informationen über den Zustand des Anlagenmodells und kann umgekehrt über das Steuerungsmodell sein Anlagenmodell beeinflussen.

Dabei geht das Anwendungsmodell davon aus, daß ihm über das Steuerungsmodell ein synchrones Modell der zu betreibenden Anlage präsentiert wird.

Die Softwarekomponenten von Sensoren bzw. Aktuatoren wurden in der vorläufigen Fassung der Begriffe 5-3 bzw. 5-4 als Teil des Anwendungsmodells definiert. Das in diesem Abschnitt vorgestellte Konzept eines Steuerungsmodells erfordert es nun, alle Aspekte des Anwendungsmodells, die die zu betreibende Anlage sowie damit verbundene Synchronizitätsgesichtspunkte betreffen, in das Steuerungsmodell auszulagern. Dazu zählen auch die hier thematisierten Sensoren und Aktuatoren.

Die endgültige Fassung der Begriffe 5-3 und 5-4 macht diesen Zusammenhang deutlich:

Begriff 5-3: Sensor (endgültige Fassung)

Ein Sensor ist eine Schnittstelle zwischen der Hardware und der Software eines technisch eingebetteten Anwendungssystems. Er dient dazu, in der vom Anwendungssystem zu betreibenden Anlage stattfindende Ereignisse zu registrieren und an die Software weiterzugeben. Sensoren können auf konzeptioneller Ebene durch zwei Komponenten modelliert werden. Die Hardware-Komponente eines Sensors registriert Ereignisse, die innerhalb der Anlage stattfinden. Diese lassen sich als analoge Eingangsgrößen interpretieren, die in digitale Ausgangssignale umgewandelt (A/D-Wandlung) und der zugehörigen Software-Komponente des Sensors als Programmereignis zur Verfügung gestellt werden. Die Hardware-Komponente eines Sensors ist Teil des Anlagenmodells, die Software-Komponente dagegen Teil des Steuerungsmodells technisch eingebetteter Anwendungssysteme.

Begriff 5-4: Aktuator (endgültige Fassung)

Ein Aktuator ist eine Schnittstelle zwischen der Hardware und der Software eines technisch eingebetteten Anwendungssystems. Er ermöglicht es dem Anwendungssystem, steuernd auf die Hardware einzuwirken und damit Veränderungen am Zustand der zu betreibenden Anlage zu veranlassen. Aktuatoren können auf konzeptioneller Ebene durch zwei Komponenten modelliert werden. Die Software-Komponente eines Aktuators wird aus dem Programm heraus aufgerufen und übermittelt der zugehörigen Hardware-Komponente des Aktuators ein digitales Eingangssignal. Dieses wird von der Hardware-Komponente in entsprechende analoge Ausgangsgrößen umgewandelt (D/A-Wandlung) und kann auf diese Weise den Zustand der einbettenden Anlage verändern. Die Hardware-Komponente eines Aktuators ist Teil des Anlagenmodells, die Software-Komponente dagegen Teil des Steuerungsmodells technisch eingebetteter Anwendungssysteme.

6 Entwurf und Konstruktion verteilter, technisch eingebetteter Anwendungssysteme

Im vorangegangenen Kapitel wurde mit der Unterscheidung von Anlagen-, Steuerungs- und Anwendungsmodell ein Konzept zur Modellierung technisch eingebetteter Anwendungssysteme vorgestellt. Es gilt nun, diese Modelle auch auf konstruktiver Ebene umzusetzen.

Offen ist dabei zunächst, wie das Zusammenspiel der einzelnen Modelle gestaltet werden kann. In diesem Zusammenhang wurde bereits in der Einleitung dieser Arbeit die These formuliert, daß Anlagen- und Steuerungsmodell technisch eingebetteter Anwendungssysteme zur Laufzeit miteinander synchronisiert werden müssen, um dem Anwendungsmodell ein synchronisiertes Bild des Anlagenmodells präsentieren zu können.

Weiter ist zu klären, wie Anwendungssysteme, die auf Basis der drei Modelle entworfen worden sind, unterteilt werden können. Anders ausgedrückt stellt sich die Frage, welche "vertikalen" Schnitte durch Anlagen- Steuerungs- und Anwendungsmodell dem zukünftigen Anwendungssystem die adäquate Erbringung seiner Aufgaben ermöglichen.

Abschließend bleibt zu zeigen, wie die Konstruktion der oben genannten Schnitte auf softwaretechnischer Ebene durch geeignete Entwurfskomponenten unterstützt werden kann. Hierfür soll das von WAM zur Verfügung gestellte Repertoire an Entwurfsmetaphern verwendet bzw. erweitert werden.

6.1 Synchronisierte Zustände des Steuerungsmodells

Das Versagen des konkreten Transport-Subsystems bei der Behandlung übergeordneter Ausnahmen konnte anhand der in Abschnitt 5.1 vorgenommenen Analyse als Synchronisationsproblem zwischen Anlagen- und Anwendungsmodell erklärt und konzeptionell durch die Einführung eines sogenannten Steuerungsmodells entschärft werden. Dem Steuerungsmodell wurde dabei die Aufgabe zugewiesen, seinem Anwendungsmodell ein synchronisiertes Bild des Anlagenmodells, respektive des Zustandes der zu betreibenden Anlage, zu präsentieren. Zu diskutieren ist nun, welche Probleme sich daraus für das Zusammenspiel der drei Modelle ergeben.

Um die oben genannte Aufgabe erfüllen zu können, muß das Steuerungsmodell zur Laufzeit mit dem Anlagenmodell synchronisiert werden. Synchronisieren bedeutet dabei vor allem, daß der Zustand des Anlagenmodells durch entsprechende Sensoren innerhalb des Steuerungsmodells registriert und abgebildet wird. Umgekehrt bedeutet Synchronisieren aber auch, daß das Steuerungsmodell über die vorhandenen Aktuatoren steuernd auf den Zustand seines Anlagenmodells einwirken kann.

Als zentrales Problem erweist sich dabei, daß relevante Zustände bzw. Zustandsübergänge des Anlagenmodells vom Steuerungsmodell nicht immer synchron, also unmittelbar und ohne relevante zeitliche Verzögerung, nachvollzogen werden können.

In diesem Zusammenhang wurde bereits in Abschnitt 5.4 festgestellt, daß das Anwendungsmodell technisch eingebetteter Anwendungssysteme lediglich über ein

asynchrones Modell der zu betreibenden Anlage verfügen kann. Gleiches gilt auch für das hier thematisierte Steuerungsmodell. Grund für diese Einschränkung ist in beiden Fällen, daß Ereignisse und Zustandsveränderungen des Anlagenmodells nur über die zur Verfügung stehenden Sensoren registriert werden können. Da deren Anzahl aber begrenzt ist, können relevante Ereignisse und Zustandsveränderungen des Anlagenmodells unter Umständen erst "mit Verspätung" registriert werden und damit zu den bereits beschriebenen Inkonsistenzen zwischen den beiden Modellen führen.

Anders ausgedrückt hat das Steuerungsmodell zwar die Aufgabe, dem Anwendungsmodell ein synchronisiertes Bild des Anlagenmodells zu präsentieren, kann selbst aber lediglich über ein asynchrones Modell ebendieser Anlage verfügen.

Nachfolgend sollen zwei denkbare Alternativen als Auswege aus diesem Dilemma diskutiert werden:

- Das Steuerungsmodell versucht, Ereignisse und Zustandsveränderungen des Anlagenmodells zu antizipieren, indem es von synchronen Informationen über den Zustand des Anlagenmodells auf zukünftige Zustandsveränderungen extrapoliert. Für den Fall, daß die antizipierten Zustandsveränderungen tatsächlich eintreten, werden sie zu einem späteren Zeitpunkt über asynchron eintreffende Ereignisse bestätigt. Das Anwendungsmodell kann so jederzeit mit Informationen über den Zustand der zu betreibenden Anlage versorgt werden, die zumindest vermutlich mit dem tatsächlichen Zustand der Anlage übereinstimmen.
- Das Steuerungsmodell modelliert explizit eine Menge von Zuständen, innerhalb derer es aufgrund einer entsprechenden Anzahl von Sensoren eine hinreichende Synchronizität mit dem Anlagenmodell garantieren kann. Das Steuerungsmodell gibt nur dann Informationen über den Zustand der zu betreibenden Anlage an das Anwendungsmodell weiter, wenn es sich in einem solchen *synchronisierten Zustand* befindet.

Für die erste Alternative läßt sich zeigen, daß sie nur dann eine geeignete Lösung des Synchronisationsproblems darstellt, wenn Abweichungen vom extrapolierten Zustand des Anlagenmodells grundsätzlich auch im nachhinein behandelbar sind. Anders ausgedrückt dürfen asynchron eintreffende Ereignisse, die Abweichungen vom extrapolierten Zustand des Anlagenmodells melden, nicht zu einem Fehlverhalten des Anwendungssystems führen. Statt dessen muß nach der nun notwendigen Resynchronisation des Steuerungsmodells ein erfolgreiches "Wiederaufsetzen" des Anwendungsmodells möglich sein.

Genau dieses Wiederaufsetzen kann aber im Falle des konkreten Transport-Subsystems nicht garantiert werden. Wendet man hier den ersten Ansatz an, so würde nach Eintreffen der Information über die erfolgreiche Aussteuerung des Probenträgers auch auf eine erfolgreiche Durchführung der Analytik extrapoliert. Eine Bestätigung dafür kann allerdings erst asynchron, am Automatischen-Zuordnungspunkt, eingeholt werden. Konnte die Analytik nicht erfolgreich durchgeführt werden und wurde der Probenträger inzwischen vom Band entnommen, so ist diese Abweichung vom extrapolierten Zustand nicht im nachhinein behandelbar und das Transport-Subsystem versagt.

In der zweiten Alternative entfällt die Voraussetzung, daß Abweichungen vom extrapolierten Zustand des Anlagenmodells grundsätzlich auch im nachhinein behandelbar sein müssen. Grund dafür ist, daß ohnehin nur synchrone bzw. resynchronisierte Informationen an das

Anwendungsmodell weitergegeben werden und somit keine im nachhinein zu korrigierenden Abweichungen auftreten können. Sie ist daher gegenüber der ersten Alternative zu bevorzugen.

Die zweite Alternative impliziert, daß überhaupt synchronisierte Zustände des Steuerungsmodells modellierbar sind. Die erste Frage in diesem Zusammenhang muß also lauten, was genau einen synchronisierten Zustand ausmacht bzw. wie man einen solchen Zustand findet. Im nächsten Abschnitt wird dann zu diskutieren sein, ob Anzahl und Anordnung der synchronisierten Zustände des Steuerungsmodells geeignet sind, um dem Anwendungsmodell ein synchronisiertes Bild des Anlagenmodells präsentieren zu können.

Eine Antwort auf die erste Frage liefert Begriff 6-1:

Begriff 6-1: Synchronisierter Zustand des Steuerungsmodells

> Das Steuerungsmodell befindet sich in einem synchronisierten Zustand, wenn es dem Anwendungsmodell aufgrund einer entsprechenden Anzahl an Sensoren eine hinreichende Form von Synchronizität mit dem Anlagenmodell garantieren kann. Hinreichend bedeutet in diesem Zusammenhang, daß keine aus Synchronisationproblemen resultierenden Fehler auftreten können. Für das Steuerungsmodell gilt, daß nur dann Informationen über das Anlagenmodell an das Anwendungsmodell weitergegeben werden dürfen, wenn es sich in einem solchen synchronisierten Zustand befindet.

In diesem Sinne ist ein synchronisierter Zustand zunächst einmal ein technisch motiviertes Konstrukt mit dessen Hilfe sich ausdrücken läßt, wann das Anwendungsmodell hinreichend synchron mit dem Anlagenmodell ist. Fragt man nun aber, wie ein solcher Zustand gefunden werden kann, so wird der fachliche Aspekt synchronisierter Zustände ersichtlich.

So läßt sich kein allgemeiner Algorithmus angeben, mit dessen Hilfe synchronisierte Zustände des Steuerungsmodells gefunden werden können. Statt dessen ist es lediglich möglich unter Berücksichtigung der fachlichen Aufgaben des Anwendungssystems zu entscheiden, ob ein gegebener Zustand des Steuerungsmodells synchronisiert ist oder nicht:

> Wenn die bis zur nächstmöglichen Zustandsänderung des Steuerungsmodells stattfindenden Zustandsänderungen des Anlagenmodells in fachlicher Hinsicht für das Anwendungsmodell nicht relevant sind, so befindet sich das Steuerungsmodell in einem synchronisierten Zustand.

Ein Beispiel soll die Anwendung dieses Entscheidungsalgorithmus verdeutlichen:

> Das Steuerungsmodell des Transport-Subsystems wird mit dem Anlagenmodell bezüglich eines Probenträgers synchronisiert, wenn dieser den Automatischen-Zuordnungspunkt passiert. Es befindet sich also zumindest zu diesem Zeitpunkt in einem synchronisierten Zustand und darf Informationen über die Position des Probenträgers auf dem Transportband an das Anwendungsmodell weitergeben. Der Zustand des Steuerungsmodells kann aus Sicht des Anlagenmodells nur dann verändert werden, wenn über einen Sensor ein entsprechendes Ereignis an das Steuerungsmodell gemeldet wird. Im Falle des betrachteten Probenträgers kann

die nächstmögliche Zustandsänderung des Steuerungsmodells also frühestens mit dem Erreichen der nächsten Weiche bzw. des dazugehörigen Tagcode-Lesers veranlaßt werden. Bis dahin besteht für das Anwendungsmodell kein fachlich relevanter Anlaß, die genaue Position des Probenträgers auf dem Transportband zu kennen oder gar verändern zu wollen. Aus diesem Grunde befindet sich das Steuerungsmodell des Transport-Subsystems bezüglich des betrachteten Probenträgers solange in einem synchronisierten Zustand, bis dieser die Weiche erreicht hat und eine anwendungsrelevante Veränderung des Anlagenmodells zumindest möglich wird.

Auf Basis der bisherigen Argumentation läßt sich also feststellen, daß das Konzept synchronisierter Zustände des Steuerungsmodells einen praktikablen Ansatz zur Gestaltung des Zusammenspiels zwischen den drei Modellen darstellt. Dabei ist es wichtig festzuhalten, daß beim Entwurf synchronisierter Zustände auch die vom Anwendungssystem zu erbringenden Aufgaben berücksichtigt werden müssen. Anders ausgedrückt kann der Entwurf des Steuerungsmodells keinesfalls unabhängig vom konkreten Anlagen- bzw. Anwendungsmodell erfolgen.

6.2 Aufgabenbezogene Schnitte

Auch die im vorangegangenen Abschnitt aufgeworfene Frage nach Anzahl und Anordnung der synchronisierten Zustände des Steuerungsmodells läßt sich nur auf fachlicher Ebene und unter Rückgriff auf die zu erbringende (Gesamt-) Aufgabe des Anwendungssystems beantworten.

Als Aufgabe des Steuerungsmodells wurde bislang "lediglich" definiert, dem Anwendungsmodell ein synchronisiertes Bild des Anlagenmodells zu präsentieren. Dazu waren Inkonsistenzen zwischen Anlagen- und Anwendungsmodell so gering zu halten, daß keine aus Synchronisationsproblemen resultierenden Fehler auftreten können.

Bei näherer Betrachtung ist aber nicht nur die Vermeidung von Fehlern für den Entwurf des Steuerungsmodells entscheidend. Ebenso wichtig ist, daß dem Anwendungssystem als Ganzes die adäquate Erbringung seiner fachlichen Aufgaben ermöglicht wird. Anders ausgedrückt darf die Existenz das Steuerungsmodells nicht dazu führen, daß zwar Synchronisationsprobleme vermieden, gleichzeitig aber die Funktion des Gesamtsystems hinsichtlich der zu erbringenden fachlichen Aufgaben beeinträchtigt wird.

Das folgende Beispiel verdeutlicht diese Problematik:

In Abschnitt 5.4 wurde auf konzeptioneller Ebene gezeigt, wie das Steuerungsmodell zur Lösung von Synchronisationsproblemen bei der Modellierung des Transport-Subsystems eingesetzt werden kann. Dazu sollte das Steuerungsmodell die erfolgreiche Durchführung eines Transport-Auftrages erst dann an das eigentliche Anwendungsmodell weitergeben, wenn diese Information aufgrund einer Resynchronisation mit dem Anlagenmodell bestätigt werden konnte. Ferner wurde darauf hingewiesen, daß eine solche Resynchronisation aufgrund der vorhandenen Hardware erst am Automatischen-Zuordnungspunkt erfolgen kann. Die Folge ist nun, daß alle Probenträger vor der endgültigen

Entnahme vom Transportband noch einmal den Automatischen-Zuordnungspunkt passiert haben müssen. Davon sind aber auch Probenträger mit erfolgreich durchgeführtem Transport-Auftrag betroffen, die nun sogenannte "Ehrenrunden" einlegen müssen. "Ehrenrunden" wiederum bedeuten für das Transportband eine erhebliche Belastung, da die Anzahl der verfügbaren bzw. gleichzeitig transportierbaren Probenträger beschränkt ist und damit letztlich der Durchsatz des Transport-Subsystems sinkt.

Das Beispiel zeigt, daß ein solchermaßen konzipiertes Steuerungsmodell seine technisch motivierte Aufgabe, nämlich dem Anwendungsmodell ein synchronisiertes Bild des Anlagenmodells zu präsentieren, erfüllt. Es zeigt aber auch, daß das Steuerungsmodell die Funktion des Transport-Subsystems hinsichtlich der adäquaten Erbringung fachlicher Aufgaben beeinträchtigt, da zur Vermeidung von Synchronisationsproblemen die beschriebenen "Ehrenrunden" in Kauf genommen werden.

Analysiert man die Ursachen dieser Beeinträchtigung, so liegen diese letzlich in Anzahl und Anordnung der innerhalb des Steuerungsmodells modellierbaren synchronisierten Zustände begründet.

So ist das Steuerungsmodell gezwungen, die Weitergabe von "aktuelleren" Informationen über die Position eines Probenträgers bzw. die Durchführung eines Transport-Auftrages zu verzögern ("*Probenträger in Aussonderungsspur ausgesteuert*"), um dem Anwendungsmodell ein synchronisiertes Bild des Anlagenmodells präsentieren zu können. Anders ausgedrückt wird ein synchronisierter Zustand des Steuerungsmodells, der bezüglich eines Probenträgers mit dem Passieren des Automatischen-Zuordnungspunktes erreicht worden ist ("*Probenträger zugeordnet und auf dem Transport*"), so lange "künstlich" aufrecht erhalten, bis eine Resynchronisation mit dem Anlagenmodell an ebendiesem Automatischen-Zuordnungspunkt stattfinden kann ("*Transport erfolgreich durchgeführt*" bzw. "*Transport gescheitert*").

Damit aber wird der fachliche Aspekt des Konzepts synchronisierter Zustände ignoriert, da das Steuerungsmodell die Weitergabe fachlich relevanter Zustandsveränderungen des Anlagenmodells, beispielsweise also die erfolgreiche Durchführung eines Transport-Auftrages, aus technisch motivierten Synchonisationsgesichtspunkten verzögert.

Es scheint also offenbar notwendig zu sein, Anzahl und Anordnung der synchronisierten Zustände des Steuerungsmodells so zu entwerfen, daß nicht nur dem technischen Aspekt der Fehlervermeidung genügt werden kann, sondern auch dem fachlichen Aspekt der Aufgabenerbringung. Zudem wurde hinsichtlich des Entwurfs einzelner synchronisierter Zustände festgehalten, daß auch dabei die Gegebenheiten der zu betreibenden Anlage und die im Anwendungsmodell implementierte Anwendungslogik berücksichtigt werden müssen.

In der Konsequenz ist damit aber auch die Anzahl und Anordnung der synchronisierten Zustände des Steuerungsmodells abhängig vom Anlagen- bzw. Anwendungsmodell.

Anders ausgedrückt stellt sich nicht nur die Frage nach Anzahl und Anordnung synchronisierter Zustände des Steuerungsmodells, sondern vor allem auch die, wie Anlagen-, Steuerungs- und Anwendungsmodell "vertikal" geschnitten sein müssen, um die adäquate Erbringung der Aufgaben des zukünftigen Anwendungssystems, die Vermeidung von Synchronisationsfehlern und einen nahtlosen Übergang zur Konstruktion geeigneter Softwarekomponenten ermöglichen.

Da diese vertikalen Schnitte durch die drei Modelle eines technisch eingebetteten Anwendungssystems vor allem anhand der zu erbringenden Aufgaben vollführt werden, sollen sie im weiteren als *aufgabenbezogene Schnitte* bezeichnet und anhand von Abbildung 6-1 verdeutlicht werden.

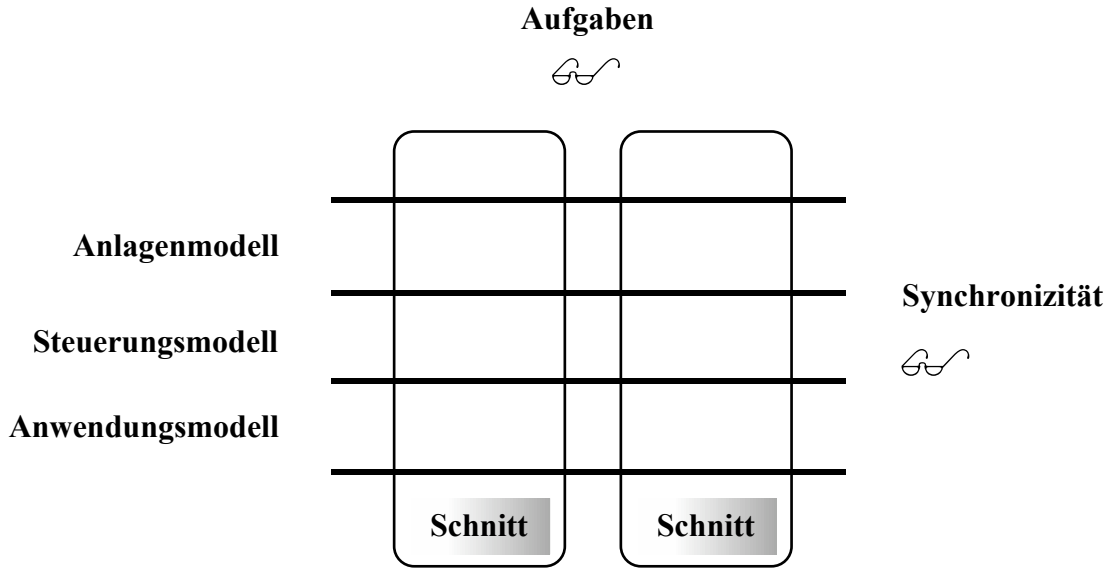

Abb. 6-1: Aufgabenbezogene Schnitte durch das Anlagen-, Steuerungs- und Anwendungsmodell technisch eingebetteter Anwendungssysteme

Um die Anwendbarkeit des Konzepts aufgabenbezogener Schnitte zu demonstrieren, sollen nachfolgend exemplarisch gezeigt werden, wie aufgabenorientierte Schnitte des Transport-Subsystems zu konzipieren wären.

> Die drei Modelle des Transport-Subsystems werden hinsichtlich der Aufgaben der einzelnen Aussonderungsspuren des Transportbandes geschnitten. Dazu wird für jede Aussonderungsspur eine zusammengehörige Menge von Hard- und Softwarekomponenten modelliert, die insgesamt die Aufgaben einer Aussonderungsspur erbringen. Das Steuerungsmodell eines solchen Schnittes modelliert synchronisierte Zustände, indem es für jeden sich im Bereich der Aussonderungsspur befindlichen Probenträger mit Hilfe einer geeigneten Anzahl und Anordnung von Sensoren die geforderte hinreichende Synchronizität mit dem Anlagenmodell herzustellen versucht. Gelingt dieses, so befindet sich das Steuerungsmodell hinsichtlich des betrachteten Probenträgers in einem synchronisierten Zustand und darf entsprechende Informationen unmittelbar an das Anwendungsmodell weitergeben. Dieses "Recht" erlischt, wenn der betreffende Probenträger die Aussonderungsspur verläßt und damit eine fachlich relevante Veränderung des Anlagenmodells verursacht. Innerhalb des Steuerungsmodells wird diese Veränderung abgebildet, indem es den synchronisierten Zustand hinsichtlich des betrachteten Probenträgers verläßt und keine entsprechenden Informationen mehr an das Anwendungsmodell weitergibt.

Entscheidend für das Transport-Subsystem ist an diesem Beispiel, daß ein solchermaßen konzipierter Schnitt von Anlagen-, Steuerungs- und Anwendungsmodell nicht nur die adäquate Erbringung der "normalen" Aufgaben des Transport-Subsystems unterstützt, sondern auch die rechtzeitige Entdeckung und Behandlung von Ausnahmebedingungen ermöglicht.

So können beispielsweise Informationen über die erfolgreiche Durchführung eines Transport-Auftrages oder das Auftreten eines Bypass vom Steuerungsmodell bezogen werden, noch während sich der Probenträger im Bereich der Aussonderungsspur befindet. Damit aber entfällt einerseits die Notwendigkeit, alle Probenträger, auch bei erfolgreicher Durchführung des Transport-Auftrages, zu Resynchronisationszwecken eine "Ehrenrunde" drehen zu lassen. Andererseits erhält das Anwendungsmodell die Möglichkeit, noch bevor der Probenträger die Entnahmestation erreicht, auf eine eventuelle Ausnahmebedingung zu reagieren und beispielsweise eine Entnahme der Probe vom Transportband zu verhindern.

Festzuhalten bleibt damit, daß das Konzept synchronisierter Zustände des Steuerungsmodells einen praktikablen Ansatz zur Gestaltung des Zusammenspiels zwischen den drei Modellen technisch eingebetteter Anwendungssysteme zu liefern vermag. Anzahl und Anordnung dieser synchronisierten Zustände werden durch aufgabenbezogene Schnitte des zukünftigen Anwendungssystems determiniert, die jeweils aus einem Teil des (Gesamt-) Anlagen-, Steuerungs- und Anwendungsmodells bestehen. Diese Schnitte gewährleisten eine adäquate Erbringung der Aufgaben des zukünftigen Anwendungssystems und dienen zur Vermeidung von Synchronisationsfehlern. Im nächsten Abschnitt wird sich zeigen, daß sie auch den nahtlosen Übergang zur softwaretechnischen Konstruktion entsprechender Softwarekomponenten ermöglichen.

Festgestellt werden muß allerdings auch, daß das hier konzipierte Steuerungsmodell des Transport-Subsystems mit der im Anwendungsbereich vorgefundenen Hardwareumgebung nicht realisiert werden kann. Ausschlaggebender Faktor ist dabei, daß mit dem Automatischen-Zuordnungspunkt nur ein einziger (Re-) Synchronisationspunkt innerhalb des Transport-Subsystems zur Verfügung steht. Als Folge davon ist es für das Steuerungsmodell nicht möglich, die geforderte hinreichende Synchronizität mit dem Anlagenmodell auf Ebene von Aussonderungsspuren herstellen. Anders ausgedrückt erlaubt es das zugrundegelegte Anlagenmodell dem Steuerungsmodell nicht, die zur Erbringung seiner Aufgaben notwendigen synchronisierten Zustände zu modellieren.

Feststellung und Lösung bestätigen daher die bereits in der Einleitung dieser Arbeit aufgestellte These hinsichtlich des Hardwareentwurfs. Dort wurde gefordert, daß der Entwurf der einbettenden Hardware, respektive der zu betreibenden Anlage, nicht unabhängig vom (Software-) Entwurf des Steuerungs- bzw. des Anwendungsmodells erfolgen darf.

Ohne tiefergehende Diskussion sei abschließend angemerkt, daß diese selbstverständlich erscheinende Forderung in der Praxis keineswegs unumstritten ist. So führt der "weiche" Charakter von Software (leider) häufig zur Forderung der Hardwareingenieure an die Softwareentwickler, ihre Software den Gegebenheiten der Hardware anzupassen, da diese weitaus aufwendiger zu verändern sei. So schreibt etwa Ellis (vgl [Ell94], S. 214):

> "As the concurrent design of the hardware evolves, software requirements, particularly those related to the hardware interface, will change. Design or

> implementation problems with the hardware often result in reallocation of requirements from hardware to software or introduce new software requirements to correct a troubled hardware implementation."

Reflektiert wird diese Sichtweise durch das klassische Wasserfall-Modell, daß der Entwicklung technisch eingebetteter Anwendungssysteme häufig zugrundegelegt wird. Dieses ordnet Hardwareentwicklung in zeitlicher Hinsicht vor Softwareentwicklung ein und impliziert damit, daß der Entwurf der Hardware unabhängig von der Software erfolgen kann.

Auf der anderen Seite werden unter dem Stichwort Hardware-/Software-Codesign allerdings mittlerweile auch Ansätze diskutiert, die beide Aktivitäten gleichberechtigt und nebenläufig innerhalb des Entwicklungsprozesses technisch eingebetteter Anwendungssysteme einordnen (vgl [Sys96], S. 24). Anhand der Ausführungen dieses Abschnittes sollte deutlich geworden sein, daß sich der in dieser Arbeit vertretene Ansatz zur Modellierung technisch eingebetteter Anwendungssysteme in dieselbe Richtung einordnen läßt.

6.3 Bearbeitungsstationen

Mit Anlagen-, Steuerungs- und Anwendungsmodell konnte im fünften Kapitel auf konzeptioneller Ebene gezeigt werden, welche zusätzlichen Modelle sich für den Entwurf technisch eingebetteter Anwendungssysteme als notwendig erweisen. Nachfolgend wurde das Konzept synchronisierter Zustände des Steuerungsmodells eingeführt, um einen Ansatz zur Gestaltung des Zusammenspiels zwischen den drei Modellen bereitzustellen. Das Anwendungssystem selbst wird in aufgabenbezogene Schnitte unterteilt, die jeweils aus einem Teil des Anlagen-, Steuerungs- und Anwendungsmodells bestehen und damit auch Anzahl, Anordnung und Umfang der synchronisierten Zustände des Steuerungsmodells determinieren. Offen ist dabei vor allem, wie aufgabenbezogene Schnitte in nach WAM entworfenen Anwendungssystemen konstruktiv umgesetzt werden können.

Diesbezüglich wurde in Kapitel 2 mit Werkzeug, Automat, Material und Arbeitsumgebung das von WAM zur Konstruktion interaktiver Anwendungssysteme bereitgestellte Repertoire an Entwurfsmetaphern vorgestellt. Versucht man, für das Konzept aufgabenbezogener Schnitte eine passende Metapher innerhalb des bereits vorhandenen Repertoires zu finden, so zeigt sich schnell, daß hier noch keine geeignete Abstraktion existiert.

Werkzeuge und Materialien, also in Software vergegenständlichte Arbeitsmittel und Arbeitsgegenstände, sind als Entwurfsmetapher für die Konstruktion aufgabenbezogener Schnitte nicht geeignet, da sie offenkundig zu feingranular zur Vergegenständlichung von Schnitten durch mehrere Modelle sind. Gleiches gilt für Automaten in ihrer fachlichen und technischen Interpretation.

Die Entwurfsmetapher Arbeitsumgebung wurde als fachlicher Rahmen für die in der täglichen Arbeit verwendeten Arbeitsmittel und Arbeitsgegenstände eingeführt. Insofern könnte man sie als Schnitt durch ein aus kooperativen Arbeitsumgebungen bestehendes Konglomerat verteilter, interaktiver Anwendungssysteme interpretieren. Die Ausführungen des Abschnittes 2.4 hinsichtlich der Unterschiede zwischen interaktiven und technisch eingebetteten Anwendungssystemen sollten aber deutlich gemacht haben, daß das Konzept der kooperativen Arbeitsumgebung für technisch eingebettete Anwendungssysteme nicht geeignet ist. So würde insbesondere die dann fällige Erweiterung der Arbeitsumgebung um

Anlagen- und Steuerungsmodell bzw. die innerhalb des Steuerungsmodells zu implementierenden Reaktionen auf Veränderungen des Anlagenmodells dem zugrundegelegten Leitbild und Kooperationsmodell von Arbeitsumgebungen widersprechen.

Diese Überlegungen machen deutlich, daß das Repertoire der von WAM zur Verfügung gestellten Entwurfsmetaphern erweitert werden muß, um aufgabenbezogene Schnitte der drei Modelle auch auf konstruktiver Ebene umzusetzen. Als Ansatzpunkt sollen die vom Anwendungssystem als Ganzes zu erbringenden Aufgaben, die seiner Schnitte und die der einzelnen Modelle innerhalb eines solchen Schnittes dienen. Sie werden zu diesem Zweck nachfolgend noch einmal kurz umrissen:

- Aufgabe eines technisch eingebetteten Anwendungssystems als Ganzes ist es, sowohl die adäquate Erbringung von fachlichen Aufgaben als auch die Vermeidung von aus Synchronisationsproblemen resultierenden Fehlern sicherzustellen. Dazu werden die drei Modelle nach aufgabenbezogenen Gesichtspunkten zunächst durch geeignete Schnitte unterteilt. Zur Laufzeit ist das Anwendungssystem dafür verantwortlich, den einzelnen Schnitten geeignete Aufträge zuzuweisen, es bildet in diesem Sinne also deren *Kontext*. Entscheidend für die in dieser Arbeit thematisierten Synchronisationprobleme ist, daß aus Sicht des Anwendungssystems zur gesamten Laufzeit erfragbar sein muß, ob ein bestimmter aufgabenbezogener Schnitt den ihm zugewiesenen Auftrag bereits erfüllt hat oder nicht.
- Aufgabe eines solchen Schnittes ist es, ihm zugewiesene Aufträge selbständig zu erfüllen. Dazu zählt auch, daß der aufgabenbezogene Schnitt keine Synchronisationsfehler innerhalb seines Kontextes verursachen darf, beispielsweise indem er Inkonsistenzen zwischen Anlagen- und Anwendungsmodell zuläßt, die erst außerhalb des "eigenen" Schnittes entdeckt und behandelt werden können. Anders ausgedrückt muß ein aufgabenbezogener Schnitt für seinen Zuständigkeitsbereich eine hinreichende Synchronizität zwischen Anlagen- und Anwendungsmodell gewährleisten, um spätestens beim "Verlassen" seines Zuständigkeitsbereiches aufgrund synchroner Informationen über das Anlagenmodell dem Kontext mitteilen zu können, ob der zugewiesene Auftrag erfüllt werden konnte oder nicht.
- Aufgabe des Anlagenmodells eines aufgabenbezogenen Schnittes ist es, den Zustand der zu betreibenden Anlage, sowie die verfügbaren Sensoren und Aktuatoren abzubilden.
- Aufgabe des Steuerungsmodells eines aufgabenbezogenen Schnittes ist es, dem Anwendungsmodell ein synchronisiertes Bild des Anlagenmodells zu präsentieren. Dazu müssen synchronisierte Zustände modelliert und implementiert werden, die eine hinreichende Form von Synchronizität mit dem Anlagenmodell gewährleisten. Informationen über das Anlagenmodell dürfen zur Laufzeit nur dann an das Anwendungsmodell weitergegeben werden, wenn sich das Steuerungsmodell in einem synchronisierten Zustand befindet.
- Aufgabe des Anwendungsmodells eines aufgabenbezogenen Schnittes ist es, die eigentliche Anwendungslogik zur Erbringung der fachlichen Aufgaben des Schnittes zu implementieren.

Fraglich ist nun, wie diese Aufgaben von den Entwurfskomponenten eines lauffähigen Anwendungssystems erbracht werden können bzw. wie eine geeignete Entwurfsmetapher zur

Vergegenständlichung aufgabenbezogener Schnitte in nach WAM entworfenen Anwendungssystemen aussehen könnte.

Bei der Suche wird deutlich, daß eine solche Entwurfsmetapher nicht auf Ebene einzelner Werkzeuge, Automaten oder Materialien anzusiedeln ist, sondern eher auf Ebene einer Arbeitsumgebung, da sie die zur Erbringung fachlicher Aufgaben notwendige Funktionalität nicht selbst implementiert, sondern vielmehr einen Rahmen für die in den einzelnen Modellen angesiedelten Entwurfskomponenten bildet.

Erschwerend kommt hinzu, daß technisch eingebettete Anwendungssysteme häufig eine verteilte Realisierung erforderlich machen (vgl. Kapitel 4). Als Konsequenz muß die neue Entwurfsmetapher nicht nur Anleitung bei der softwaretechnischen Konstruktion aufgabenbezogener Schnitte geben, sondern zudem auch ein geeignetes Kooperationsmodell, analog zu dem kooperativer Arbeitsumgebungen, bereitstellen.

Die sogenannte *Bearbeitungsstation* scheint für die hier beschriebenen Zwecke eine geeignete Metapher darzustellen:

Begriff 6-2: Bearbeitungsstation

Eine Bearbeitungsstation dient innerhalb technisch eingebetteter Anwendungssysteme als Entwurfsmetapher zur Vergegenständlichung aufgabenbezogener Schnitte von Anlagen-, Steuerungs- und Anwendungsmodell. In der fachlichen Interpretation ist sie dafür verantwortlich, Aufträge, die ihr der jeweilige Kontext erteilt, selbständig und im Zusammenspiel mit "ihrem" Teil der zu betreibenden Anlage abzuarbeiten. Weiter muß sie eine hinreichende Form von Synchronizität zwischen Anlagen- und Anwendungsmodell gewährleisten, um ihrem Kontext zur gesamten Laufzeit synchronisierte Informationen über den Stand der Auftragsabarbeitung liefern zu können. In der technischen Interpretation bildet eine Bearbeitungsstation den Ort für diejenigen Entwurfskomponenten, die als Teile von Steuerungs- und Anwendungsmodell den eigentlichen Zustand der zu betreibenden Anlage, die zur Verfügung stehenden Schnittstellen zwischen Hard- und Software und die eigentliche Anwendungslogik implementieren. Die konzeptionelle Abgeschlossenheit von Bearbeitungsstationen hinsichtlich der erteilten Aufträge erlaubt es, diese auf technischer Ebene als eigenständige Prozesse zu implementieren, die ohne menschlichen Benutzer "im Hintergrund" ablaufen können.

Die Bearbeitungsstation erweitert das von WAM zur Verfügung gestellte Repertoire an Entwurfsmetaphern. Sie unterstützt die objektorientierte Modellierung technisch eingebetteter Anwendungssysteme, indem sie es ermöglicht, die einzelnen Hardwarekomponenten der zu betreibenden Anlage nach aufgabenbezogenen Gesichtspunkten zu kapseln. Diese Herangehensweise reflektiert den ohnehin meist objektorientierten Charakter von Hardware, da Hardware-Baugruppen eine zweckgebundene softwaretechnische Entsprechung in Bearbeitungsstationen finden können.

Das Konzept der Aufträge erlaubt es ferner, daß aus Bearbeitungsstationen aufgebaute Anwendungssysteme ein relativ einfaches, aber dennoch mächtiges Kooperationsmodell implementieren können.

So wurde bereits gezeigt, daß Bearbeitungsstationen dahingehend entworfen werden, ihre Aufträge selbständig und ohne Beteiligung anderer Bearbeitungsstationen abarbeiten zu können. Damit aber entfällt die Notwendigkeit, ein Kooperationsmodell innerhalb von Bearbeitungsstationen implementieren zu müssen, daß über die Entgegennahme und Rückgabe von abgearbeiteten bzw. nicht-abarbeitbaren Aufträgen hinausgeht.

Weiter wurde gefordert, daß Bearbeitungsstationen in der Lage sein müssen, ihrem Kontext zur gesamten Laufzeit synchrone Informationen über den Stand der Auftragsabarbeitung zu liefern. Damit ist es für den jeweiligen Kontext einer Bearbeitungsstation möglich, die zu leistende Koordinationsarbeit auf die Erteilung von Aufträgen zu beschränken, die von der betreffenden Bearbeitungsstation nach erfolgreicher bzw. nicht erfolgreicher Abarbeitung des Auftrages an den Kontext zurückgegeben werden. Anders ausgedrückt ist es also nicht mehr notwendig, komplexe Mechanismen zur Ausnahmebehandlung, etwa das im vierten Kapitel vorgestellte Supervisor-Konzept, vorzusehen.

Am Ende dieses Abschnittes steht mit der Bearbeitungsstation eine Entwurfsmetapher zur Verfügung, die es erlaubt, das Konzept aufgabenbezogener Schnitte in nach WAM entworfenen, technisch eingebetteten Anwendungssystemen zu vergegenständlichen. Im nächsten Abschnitt soll exemplarisch gezeigt werden, wie die neue Entwurfsmetapher verwendet werden kann, um das in dieser Arbeit thematisierte Transport-Subsystem zu konstruieren.

6.4 Bandmodule

Bereits in Abschnitt 6.2 wurde konzeptionell aufgezeigt, wie das Transport-Subsystem aufgabenbezogen geschnitten werden könnte, um nicht nur die adäquate Erbringung seiner "normalen" Aufgaben, sondern auch die Behandlung von Ausnahmen zu ermöglichen. Als Ansatzpunkt für diese Schnitte wurden die einzelnen Aussonderungsspuren des Transportbandes vorgeschlagen.

Verwendet man die Entwurfsmetapher von der Bearbeitungsstation zur Vergegenständlichung dieser Schnitte, so zeigt sich, daß diese innerhalb des Transport-Subsystems durch sogenannte *Bandmodule* konstruktiv umgesetzt werden können. Anders ausgedrückt ist es möglich, Bandmodule innerhalb des nach WAM entworfenen Transport-Subsystems als Bearbeitungsstationen zu konstruieren.

Auf Ebene des Anlagenmodells umfaßt ein solches Bandmodul jeweils eine Aussonderungsspur mit ihren Weichen und Tagcode-Lesern, sowie die sich im Bereich der Aussonderungsspur befindlichen Probenträger.

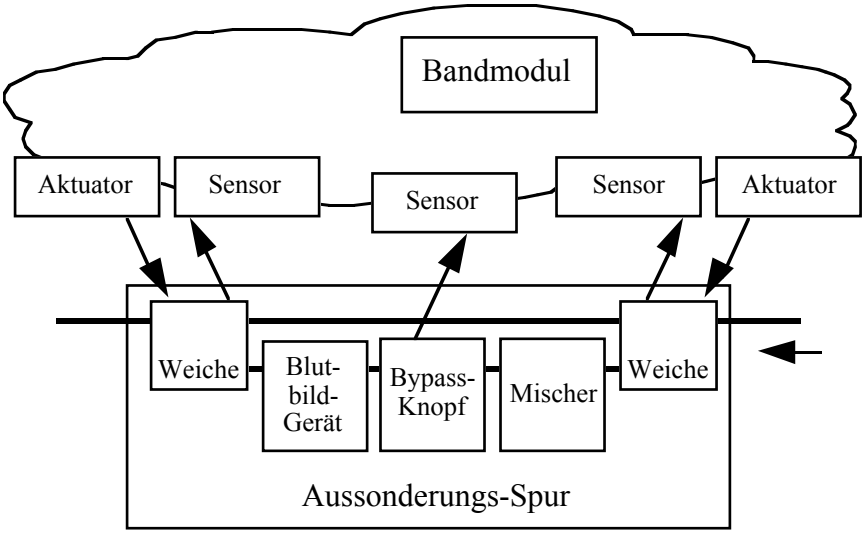

Abb. 6-2: Bandmodul

Das Anwendungsmodell eines Bandmoduls ist dafür verantwortlich, vom Transport-Supervisor erteilte Transport-Aufträge im Zusammenspiel mit den Tagcode-Lesern und Weichen des Anlagenmodells abzuarbeiten, d.h. die im Transport-Auftrag vermerkten Probenträger auszusteuern und nach Abschluß der Analytik auf den Ring des Transportbandes zurückzuführen.

Weiter fordert die Entwurfsmetapher von der Bearbeitungsstation, daß diese ihrem Kontext zur gesamten Laufzeit synchronisierte Informationen über den Stand der Auftragsbearbeitung bereitstellen muß. Das Anwendungsmodell eines Bandmoduls setzt diese Forderung um, indem es, sobald ein Probenträger die Aussonderungsspur verläßt, den Transport-Supervisor unterrichtet, ob der erteilte Auftrag erfüllt werden konnte, oder ob die Auftragsabarbeitung aufgrund einer Ausnahmebedingung (Bypass) abgebrochen werden mußte. Befindet sich der Probenträger dagegen noch im Zuständigkeitsbereich des Bandmoduls, also innerhalb der Aussonderungsspur, so ist der Auftrag aus Sicht des Transport-Supervisors noch nicht erfüllt.

Um diese Aufgabe gegenüber dem Transport-Supervisor erbringen zu können, muß das Anwendungsmodell spätestens wenn der betrachtete Probenträger die Aussonderungsspur verläßt, über hinreichend synchrone Information bezüglich des Auftretens einer Ausnahmebedingung verfügen.

Diese Informationen bereitzustellen ist Aufgabe des Steuerungsmodells. Das Steuerungsmodell eines Bandmoduls implementiert dazu ein synchronisiertes Modell des Zustandes der Aussonderungsspur. Dieses umfaßt zunächst einmal die Stellung der Weichen und die Positionen derjenigen Probenträger, die sich im Bereich der Aussonderungsspur befinden.

Um die geforderten Informationen hinsichtlich der Betätigung des Bypass-Knopfes bereitstellen zu können, müssen Anlagen- und Steuerungsmodell mit einem zusätzlichen Sensor ausgestattet werden, der die Betätigung des Bypass-Knopfes registriert (vgl. Abbildung 6-2). Durch diesen wird das Steuerungsmodell in die Lage versetzt, dem Anwendungsmodell ein hinreichend synchronisiertes Bild des Anlagenmodells zu präsentieren. Dieses nutzt wiederum das Anwendungsmodell, um die oben geforderten

Informationen über den Stand der Auftragsabarbeitung "rechtzeitig" an den Transport-Supervisor zu liefern.

Noch einmal wird an dieser Stelle also deutlich, daß das hier konstruierte Steuerungsmodell mit der im Anwendungsbereich vorgefundenen Hardwareumgebung nicht realisiert werden kann, sondern statt dessen Anpassungen an der Hardware, respektive dem Anlagenmodell, erforderlich macht.

Damit kann aber auch festgehalten werden, daß die Entwurfsmetapher von der Bearbeitungsstation einen nahtlosen Übergang zur Konstruktion von Bandmodulen innerhalb des nach WAM entworfenen Transport-Subsystems unterstützt. Zu zeigen bleibt nun, wie sich als Bearbeitungsstationen entworfene Bandmodule in den Kontext des Transport-Subsystems einbetten lassen.

In diesem Zusammenhang wird schnell deutlich, daß auch die Einbettung der einzelnen Bandmodule in den Kontext des Transport-Subsystems nahtlos über das im vorangegangenen Abschnitt beschriebene Kooperationsmodell von Bearbeitungsstationen realisierbar ist.

So ist es für den Transport-Supervisor als koordinierende Funktionseinheit des Transport-Subsystems möglich, die zu leistende Koordinationsarbeit auf die auslastungsabhängige Verteilung von Probenträgern auf die einzelnen Bandmodule, sowie die Erteilung und Rücknahme von entsprechenden Aufträgen zu beschränken. Da die einzelnen Bandmodule synchronisierte Informationen über den Stand der Auftragsabarbeitung liefern, entfällt die in Abschnitt 4.4 beschriebene Notwendigkeit, diese Information im nachhinein und auf Kosten möglicherweise aufwendiger Fehlerbehandlung am Automatischen-Zuordnungspunkt zu rekonstruieren.

Abbildung 6-3 verdeutlicht noch einmal das hier vorgestelle Konzept eines aus Bandmodulen aufgebauten Transport-Subsystems:

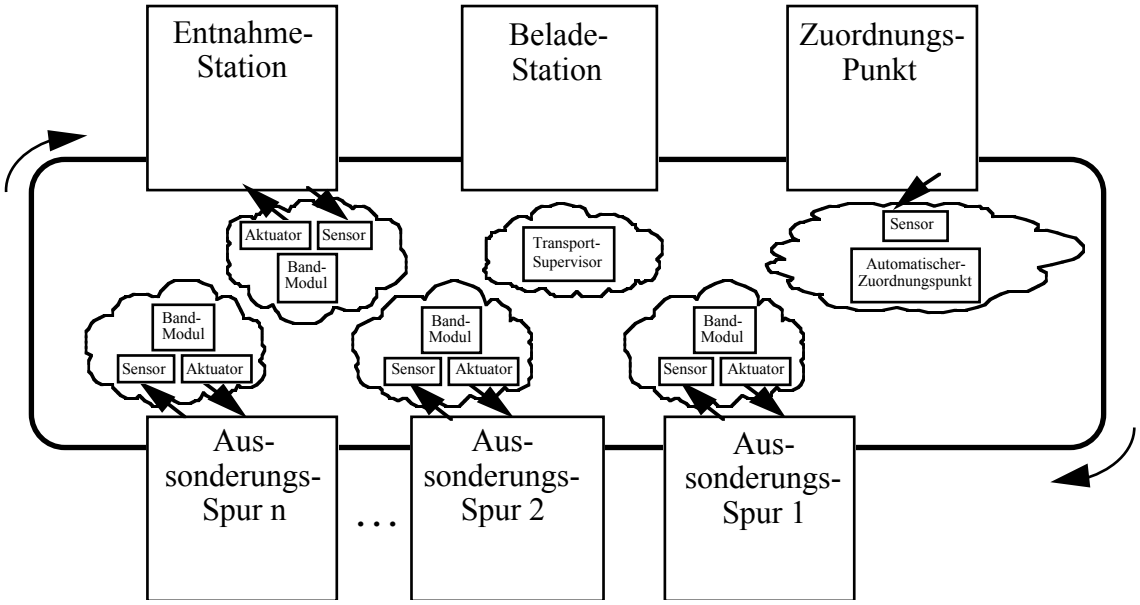

Abb. 6-3: Transport-Subsystem mit Bandmodulen

Auch die in Kapitel 4 vorgestellte Architektur des XLab-Systems läßt sich als System von Bearbeitungsstationen interpretieren. So bildet die Auftragssteuerung den Kontext für die einzelnen Funktionseinheiten des XLab-Systems, die sich aus Perspektive der Auftragssteuerung als Bearbeitungsstationen verhalten und von dieser über Aufträge koordiniert werden.

Funktionseinheiten wiederum können als Subsysteme modelliert werden, die, wie am Beispiel des Transport-Subsystems gezeigt, selbst wieder aus Funktionseinheiten, respektive Bearbeitungsstationen, bestehen und über (Sub-) Aufträge koordiniert werden.

Damit wird am Ende dieses Kapitels nicht nur die rekursive Verwendbarkeit des Konzepts von Bearbeitungsstationen deutlich, sondern auch, daß diese ein tragfähiges Konzept für den Entwurf und die Konstruktion verteilter, technisch eingebetteter Anwendungssysteme "im Großen" darstellen.

7 Zusammenfassung und Ausblick

Dieses Kapitel komplettiert die vorliegende Arbeit, indem die wesentlichen Argumentationsschritte noch einmal nachvollzogen, Ergebnisse zusammengefaßt, sowie offene Punkte und Themen für weiterführende Arbeiten aufgezeigt werden.

Anliegen dieser Arbeit war es, den Anwendungsbereich der Softwareentwicklungsmethode WAM um Konzepte zur Modellierung verteilter, technisch eingebetteter Anwendungssysteme zu erweitern. Den praktischen Erfahrungshintergrund bildete dabei ein Projekt zur Entwicklung eines Laborinformations- und Steuerungssystems, bei dem die Methode WAM zum Einsatz kam und in dessen Kontext die Konzepte dieser Arbeit problematisiert und erprobt werden konnten.

Zu Beginn dieser Arbeit wurden eine Reihe von Thesen aufgestellt, die vor allem auf die konzeptionellen Unterschiede zwischen der Modellierung interaktiver und der Modellierung technisch eingebetteter Anwendungssysteme abzielten. Insbesondere wurde postuliert, dem aus der Modellierung interaktiver Anwendungssysteme gewohnten Anwendungsmodell zwei zusätzliche Modelle zur Seite zu stellen. Diese beiden Modelle, Anlagen- und Steuerungsmodell, sollten die speziellen Belange technisch eingebetteter Anwendungssyteme berücksichtigen. Desweiteren wurde das entscheidende Problem technisch eingebetteter Anwendungssysteme, die Synchronisation von Hard- und Software, problematisiert und als Konsequenz eine gleichwertige Behandlung von Hard- und Softwareentwurf bei der Entwicklung technisch eingebetter Anwendungssysteme gefordert. Für die innerhalb des Laborprojekts eingesetzte Softwareentwicklungsmethode WAM wurde zudem vorgeschlagen, die Existenz der zusätzlichen Modelle durch eine neue Entwurfsmetapher zu vergegenständlichen.

Das weitere Vorgehen bestand nun darin, die aufgestellten Thesen anhand einer konkreten Problemstellung aus dem Kontext des Laborprojektes zu motivieren. Dazu wurde zunächst die Architektur des Laborinformations- und Steuerungssystems XLab überblicksartig vorgestellt und nachfolgend ein spezielles Teilproblem, der automatisierte Transport von Proben zum Arbeitsplatz oder Analysegerät, thematisiert.

Im Anschluß daran wurden zwei Lösungen vorgestellt und die Grenzen des jeweiligen Ansatzes aufgezeigt. Insbesondere konnte gezeigt werden, daß sich auch für die komplexere der beiden Lösungen, das sogenannte Transport-Subsystem, Fehler-Szenarien konstruieren lassen, unter denen das Subsystem trotz seiner ausgefeilten Ausnahmebehandlung versagt.

Eine in diesem Zusammenhang durchgeführte Analyse der aufgetretenen Fehler ergab, daß sich das Versagen des Transport-Subsystems abstrahiert als Synchronisationsproblem zwischen dem Modell der einbettenden Anlage (Transportband) und dem steuernden Anwendungsmodell (Transport-Subsystem) erklären läßt, bei dem sich das scheinbar korrekte Verhalten des Anwendungsmodells, bezogen auf den tatsächlichen Zustand des Anlagenmodells, als fehlerhaft erweist.

Als Konsequenz wurde ein zusätzliches (Software-) Modell, das sogenannte Steuerungsmodell eingeführt. Dieses bildet den Zustand der zu betreibenden Anlage, sowie die zur Verfügung stehenden Schnittstellen zwischen Hard- und Software, sogenannte

Sensoren und Aktuatoren, ab. Es wird "zwischen" Anlagen- und Anwendungsmodell angeordnet und hat zur Laufzeit die Aufgabe, dem Anwendungsmodell ein synchronisiertes Bild des Anlagenmodells zu präsentieren.

Um diese Aufgabe erfüllen zu können, muß das Steuerungsmodell zur Laufzeit mit dem Anwendungsmodell synchronisiert werden. Dazu wurde das Konzept der synchronisierten Zustände des Steuerungsmodells eingeführt. Befindet sich das Steuerungsmodell in einem solchen Zustand, so kann es eine hinreichende Synchronizität mit dem Anlagenmodell garantieren und darf daher (synchronisierte) Informationen über den Zustand der zu betreibenden Anlage an das Anwendungsmodell weitergeben.

Allerdings wurde bei der Suche nach einem Algorithmus zum "Finden" der synchronisierten Zustände des Steuerungsmodells deutlich, daß die vom Anwendungssystem zu erbringenden Aufgaben beim Entwurf dieser Zustände berücksichtigt werden müssen. Auch bei der Beantwortung der sich anschließenden Frage nach Anzahl und Anordnung der synchronisierten Zustände des Steuerungsmodells konnte die entscheidende Rolle fachlicher Aufgaben und damit auch eine Abhängigkeit der synchronisierten Zustände des Steuerungsmodells vom jeweiligen Anlagen- und Anwendungsmodell des Anwendungssystems nachgewiesen werden.

Im folgenden wurde deshalb das Konzept aufgabenbezogener Schnitte motiviert, daß die adäquate Erbringung der Aufgaben des zukünftigen Anwendungssystems und die Vermeidung von Synchronisationsfehlern durch "vertikale" Schnitte von Anlagen-, Steuerungs- und Anwendungsmodell ermöglicht.

Beim Übergang vom Entwurf dieser Schnitte zu deren softwaretechnischer Konstruktion mußte allerdings festgestellt werden, daß das bislang von WAM zur Verfügung gestellte Repertoire an Entwurfsmetaphern nicht ausreicht, um aufgabenbezogene Schnitte zu vergegenständlichen. Aus diesem Grunde wurde das WAM-Repertoire um die sogenannte Bearbeitungsstation erweitert und ein Kooperationsmodell für Bearbeitungsstationen in verteilten Anwendungssystemen vorgeschlagen.

Die neue Entwurfsmetapher von der Bearbeitungsstation war abschließend exemplarisch zur Konstruktion des Transport-Subsystems einzusetzen. Dazu wurde zunächst auf Ebene der einzelnen Bearbeitungsstation gezeigt, wie sogenannte Bandmodule als Bearbeitungsstationen zu konstruieren sind. Nachfolgend konnte demonstriert werden, daß auch die Einbettung der einzelnen Bandmodule in den Kontext des Transport-Subsystems nahtlos über das Kooperationsmodell von Bearbeitungsstationen realisierbar ist.

Auf Basis dieser Argumentationsschritte können die wesentlichen Ergebnisse der vorliegenden Arbeit nun wie folgt stichpunktartig zusammengefaßt werden:

- Die eingangs vorgebrachten Thesen bezüglich der zusätzlichen Modelle technisch eingebetteter Anwendungssysteme wurden durch Abstraktion eines konkreten Fehler-Szenarios aus dem Kontext des Laborprojektes motiviert. Sie erfuhren ihre argumentative Bestätigung als gezeigt werden konnte, wie sich die problematisierten Synchronisationsfehler des konkreten Transport-Subsystems durch aufgabenbezogene Schnitte der drei Modelle vermeiden lassen.
- Das Konzept aufgabenbezogener Schnitte von Anlagen-, Steuerungs- und Anwendungsmodell bestätigt die These, daß der Entwurf der einbettenden Hardware,

respektive der zu betreibenden Anlage, nicht unabhängig vom (Software-) Entwurf des Steuerungs- bzw. des Anwendungsmodells erfolgen darf. So konnten aufgabenbezogene Schnitte der drei Modelle des konkreten Transport-Subsystems erst dann vorgenommen werden, als auch der Entwurf der Hardware zur Disposition gestellt wurde.

- Das Repertoire der von WAM zur Verfügung gestellten Entwurfsmetaphern muß zur Vergegenständlichung aufgabenbezogener Schnitte um die sogenannte Bearbeitungsstation erweitert werden, da die bereits vorhandenen Metaphern Werkzeug, Material, Automat und Arbeitsumgebung keine geeignete Abstraktion darstellen.
- Der Einsatz objektorientierter Konzepte und Methoden zur Modellierung technisch eingebetteter Anwendungssysteme ist möglich und hat sich, speziell in der Ausprägung der Softwareentwicklungsmethode WAM, für die Konzeption und Realisierung eines konkreten Laborinformations- und Steuerungssystems bewährt.

Auch wenn das Anliegen der vorliegenden Arbeit, eine Erweiterung des Anwendungsbereichs der Methode WAM, mit diesen Ergebnissen im Wesentlichen als erfüllt betrachtet werden kann, so bleiben doch einige Punkte offen.

Da ist zum einen das WAM zugrundegelegte Leitbild vom Arbeitsplatz für qualifizierte menschliche Tätigkeit. Dieses muß im Zusammenhang mit Arbeitsplätzen in verteilten, technisch eingebetteten Anwendungssystemen überdacht und diskutiert werden, um auch dort Gestaltungsziele bei der Softwareentwicklung beschreiben zu können. Auf Basis einer solchen Diskussion müßte es ferner möglich sein, den in dieser Arbeit nur angerissenen Begriff des Funktionsarbeitsplatzes zu präzisieren und in den Kontext der Entwurfsmetaphern Arbeitsumgebung und Bearbeitungsstation zu stellen. Dabei stellt sich vor allem die Frage, ob Funktionsarbeitsplätze im Lichte des (neuen) Leitbildes als Bearbeitungsstationen interpretierbar sind.

Die in dieser Arbeit neu eingeführte Entwurfsmetapher von der Bearbeitungsstation wurde in ihrer technischen Interpretation als Ort für diejenigen Entwurfskomponenten definiert, die als Teile von Steuerungs- und Anwendungsmodell den eigentlichen Zustand der zu betreibenden Anlage, die zur Verfügung stehenden Schnittstellen zwischen Hard- und Software und die eigentliche Anwendungslogik implementieren. Offen geblieben ist dabei, wie eine solche Bearbeitungsstationen auf Ebene des technischen Klassenentwurfs umgesetzt werden kann. Innerhalb des Laborprojektes wurden diesbezüglich eine Reihe von Ansätzen entwickelt und erprobt, die aber vor allem aufgrund zeitlicher Restriktionen im Rahmen dieser Arbeit nicht weiter diskutiert werden können.

Als Ansatzpunkte bzw. Themen für weiterführende Arbeiten sollen daher an dieser Stelle nur die folgenden Stichpunkte genannt werden:

- Entwurfsmuster, z.B. *State* für synchronisierte Zustände des Steuerungsmodells, *Observer* für die Kopplung von Steuerungs- und Anwendungsmodell (vgl. [GHJ+95])
- Fernsteuerung und Unterbrechbarkeit von Bearbeitungsstationen ("Not-Aus"-Schalter)
- Frameworks für Sensoren, Aktuatoren und Gerätetreiber

Literaturverzeichnis

[Abb83] R.J. Abbot: *Program Design by Informal English Descriptions*. In: Communications of the ACM, Vol 26, Nr. 11, 1993.

[BCS92] R. Budde, M.-L. Christ-Neumann, K.-H. Sylla: *Tools and Materials, an Analysis and Design Metaphor*. In: Proceedings of the TOOLS 7, New York, London, Prentice Hall, 1992.

[BGZ95a] D. Bäumer, G. Gryczan, H. Züllighoven: *Objektorientierte Softwareentwicklung für Banken - Methodik und Erfahrungen aus einer mehrjährigen Projektpraxis*. OBJEKTSpektrum, (1995) 3, S. 45-53.

[BGZ95b] U. Bürkle, G. Gryczan, H. Züllighoven: *Object-oriented System Development in a Banking Project - Methodology, Experience and Conclusions*. Human-Computer-Interaction, 10 (1995) 2 & 3, S. 293-336.

[BKK+92] R. Budde, K. Kautz, K. Kuhlenkamp, H. Züllighoven: *Prototyping*. Berlin, Heidelberg, Springer, 1992.

[Boo91] G. Booch: *Object-Oriented Design*. Benjamin/Cummings, Redwood City, CA, 1991.

[Bro86] Brockhaus-Enzyklopädie in 24 Bänden - 19. Auflage, Bd. 1. Mannheim, 1986.

[BS95a] R. Budde, K-H. Sylla: *Objektorientierte Echtzeit Anwendungen*. OBJEKTSpektrum, (1995) 2, S. 54-60.

[BS95b] R. Budde, K-H. Sylla: *Eingebettete Echtzeitsysteme*. OBJEKTSpektrum, (1995) 4, S. 10-16.

[BZ90] R. Budde, H. Züllighoven: *Software-Werkzeuge in einer Programmierwerkstatt*. Berichte der Gesellschaft für Mathematik und Datenverarbeitung, Nr. 182, Oldenbourg, München, Wien, 1990.

[BZ92] R. Budde, H. Züllighoven: *Software Tools in a Programming Workshop.* In: C. Floyd, H. Züllighoven, R. Budde, R. Keil-Slawik (Hrsg.). Software Development and Reality Construction. Springer, Berlin, Heidelberg, New York, 1992, S. 252-268.

[CAP88] College of American Pathologists: *Standards for Laboratory Accreditation - 1998 Edition*. Northfield, IL, 1998.

[CY91] P. Coad, E. Yourdon: *Object-Oriented Analysis - Second Edition*. Yourdon Press, Englewood Cliff, 1991.

[DIN90] Deutsches Institut für Normung: *Allgemeine Kriterien zum Betreiben von Prüflaboratorien*. Deutsche und Europäische Norm DIN EN 45 001, Berlin, 1990.

[Ell94] J.R. Ellis: *Objectifying Real-Time Systems*. SIGS Books, New York, 1994.

[Flo87] C. Floyd: *Outline of a Paradigm Change in Software Engineering*. In: G. Bjerknes et al. (Hrsg.). Computers and Democracy - A Scandinavian Challenge, Avebury, 1987.

[GHJ+95] E. Gamma, R. Helm, R. Johnson, J. Vlissides: *Design Patterns - Elements of Reusable Object-Oriented Software*. Reading, MA, Addison-Wesley, 1995.

[GLP91] *Grundsätze der Guten Laborpraxis* - Anhang 1 zu § 19a, Absatz 1 des Chemikaliengesetzes vom März 1990 in der Fassung vom Juni 1991.

[Gry96] G. Gryczan: *Prozeßmuster zur Unterstützung kooperativer Tätigkeit*. Deutscher Universitäts Verlag, Wiesbaden, 1996.

[GZ92] G. Gryczan, H. Züllighoven: *Objektorientierte Systementwicklung - Leitbild und Entwicklungsdokumente*. In: Informatik-Spektrum , 15 (5), S. 264-272

[Hal93] N. Halbwachs: *Synchronous Programming of Reactive Systems*. Kluwer Academic Publishers, 1993.

[HS95] L. Heuser, A. Schill: *Verteiltes objektorientiertes Programmieren mit C++ - Das Programmsystem DC++*. International Thomson Publishing, Bonn, 1995.

[HW90] T. Hey, P. Waltung: *Quantenuniversum - Die Welt der Wellen und Teilchen.* Spektrum der Wissenschaft Verlagsgesellschaft, Heidelberg, 1990.

[Jäg87] B.Jäger: *Regelungstechnik*. In W. Beitz, K.-H. Küttner (Hrsg.). Dubbel - Handbuch für den Maschinenbau, 16. Auflage. Springer, Berlin, Heidelberg, New York, London, Paris, Tokyo 1987.

[Jac92] I. Jacobson: *Object-Oriented Software Engineering - A Use Case Driven Approach*. Reading, MA, Addison-Wesley, 1992.

[JV87] E. Jessen, R. Valk: *Rechensysteme - Grundlagen der Modellbildung*. Springer, Berlin, Heidelberg, 1987.

[KGZ+94] K. Kilberth, G. Gryczan, H. Züllighoven, D. Bäumer, R. Budde, K. Hasbron-Blume, K.-H. Sylla, V. Weimer: *Objektorientierte Anwendungsentwicklung*. 2. Auflage. Vieweg, 1994.

[Maa94] S. Maaß: *Maschine, Partner, Medium, Welt ... Eine Leitbildgeschichte der Software-Ergonomie*. Erscheint in: Hellige, H.D. (Hrsg.). Leitbilder der Informatik- und Computer-Entwicklung. Tagung der GI-Fachgruppe 'Historische Aspekte von Informatik und Gesellschaft' und des Deutschen Museums, München, Vordruck des Tagungsbandes, Bremen, 1994.

[Mey88] B. Meyer: *Object-Oriented Software Construction*. Prentice Hall, New York, 1988. Deutsch: Objektorientierte Softwareentwicklung. Hanser, 1990.

[Mey92] B. Meyer: *Eiffel: the language*. Prentice Hall International (UK) Ltd., 1992.

[MO92] S. Maaß, H. Oberquelle: *Perspectives and Metaphores for Human-Computer Interaction*. In: C. Floyd, H. Züllighoven, R. Budde, R. Keil-Slawik (Hrsg.). Software Development and Reality Construction. Springer, Berlin, Heidelberg, New York, 1992, S. 233-251.

[ND78] K. Nygard, O.-J. Dahl: *The Development of the Simula Language.* In: ACM SIGPLAN Notices 17:8, 1978.

[RBP+91] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, W. Lorensen: *Object-Oriented Modeling and Design*. Prentice-Hall, London, 1991.

[Rum95] J. Rumbaugh: *OMT - The object model*. In: Journal of Object-Oriented Programming Vol. 7, No. 8 (January 1995).

[SM88] S. Shlaer, S. J. Mellor: *Object-Oriented Systems Analysis: Modeling the World in Data*. Prentice Hall, Englewood Cliffs, NJ, 1988.

[SM92] S. Shlaer, S. J. Mellor: *Object Lifecycles: Modeling the World in States*. Prentice Hall, Englewood Cliffs, NJ, 1992.

[Str91] B. Stroustrup: *The C++ Programming Language*. Addison-Wesley, Reading, MA, 1991. Second Edition.

[Sys96] *Hardware- und Software-Co-Design in der Praxis*. In: Systeme - Zeitschrift für Elektronikentwickler und Systemintegratoren, (Juli 1996) 7, S. 24-28.

[Tay93] D. Taylor: *How To Find Good Objects*. Object Magazine 3 (3) (September/Oktober 1993).

[VS95] S. Vinoski, D. Schmidt: *Comparing Alternative Distributed Programming Techniques*. C++-Report, 7 (1995) 4, S. 44-53.

[Weg90] P. Wegner: *Concepts and Paradigms of Object-Oriented Programming*. In: OOPS Messenger 1, 1990, S. 8-87.

[Wul95] M. Wulf: *Konzeption und Realisierung einer Umgebung zur Koordination rechnergestützer Taetigkeiten in kooperativen Arbeitsprozessen*. Diplomarbeit, Fachbereich Informatik, Universität Hamburg, 1995.

[WW94] D. Weske, M. Wulf: *Konzepte zur Materialversorgung verteilter Werkzeugumgebung am Beispiel der Anbindung einer objektorientierten Datenbank*. Studienarbeit, Universität Hamburg, Fachbereich Informatik, März 1994.

[WWW90] R.J. Wirfs-Brok, B. Wilkerson, L. Wiener: *Designing Object-Oriented Software*. Prentice Hall, New York, 1990.

[Zül92] H. Züllighoven: *Umgang mit Software oder Software als Werkzeug und Material*. In: W. Coy et al. (Hrsg.). Sichtweisen der Informatik, Vieweg, Braunschweig, 1992.

Anhang: Semantik von Prozeß-Interaktions-Diagrammen

Die hier verwendeten Prozeß-Interaktions-Diagramme (PID) basieren sogenannten Objekt-Interaktions-Diagrammen (vgl. [Jac92]). Sie werden normalerweise zur Beschreibung der Dynamik von Objekten innerhalb eines Prozesses verwendet.

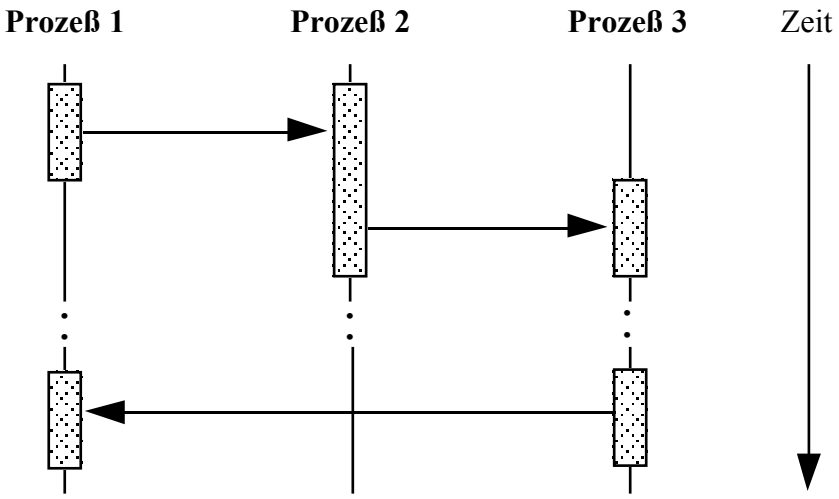

Abb. A-1: Objekt- bzw. Prozeß-Interaktions-Diagramm

Objekt-Interaktions-Diagramme (OID) bestehen im wesentlichen aus drei Komponenten. Vertikale Linien in Richtung der Zeitachse modellieren die Lebensdauer eines Objektes. Rechtecke bezeichen Zeitabschnitte innerhalb der Lebensdauer, in denen das Objekt aktiv ist, d.h. Zeitabschnitte, in denen der Kontrollfluß der Anwendung durch eine Methode dieses Objekts verläuft. Horizontale Pfeile kennzeichen Methodenaufrufe, d.h. die Weitergabe des Kontrollflusses aus dem Methodenrumpf des rufenden Objektes an eine Methode des gerufenen Objekts.

Interpretiert man die Objekte eines OID als unabhängig voneinander ablaufende Prozesse und betrachtet man die Weitergabe des Kontrollflusses nicht als Methodenaufruf innerhalb eines Prozesses, sondern als Interprozeßkommunikation, so lassen sich die solchermaßen entstandenen PIDs zur Darstellung der Dynamik verteilter Systeme verwenden. Die vertikalen Linien modellieren dabei die Lebensdauer eines Prozesses, die Rechtecke entlang der vertikalen Linien die Zeitabschnitte, in denen der betreffende Prozeß aktiv ist.