Modellierung einer wissensbasierten, interaktiven Prozeßsteuerung

Volker Dötsch
Universität Leipzig
Institut für Informatik
Postfach 920
04009 Leipzig

volkerd@informatik.uni-leipzig.de
http://www.informatik.uni-leipzig.de/~volkerd/

Juni 1998

Inhaltsverzeichnis

1 Einleitung, Motivation

Der zunehmende Bedarf, wissensbasierte Systeme für realistische Anwendungen einzusetzen, stellt eine Herausforderung an die Modellierung der zugrundeliegenden Domäne dar. Einführungen zu Expertensystemen, Wissensrepräsentation sowie Wissensverarbeitung findet man beispielsweise in [Pup91], [Str91], [Bib93], [Hen91], [Pup90], [BJT96b], [Sch98] und [Min74]. Einige kritische Betrachtungen zu Fragen der Wissensverarbeitung findet man in [BHJR94] oder auch in [GS96]. Speziell auf die Problematik der Wissensrepräsentation in komplexen Domänen wird in [Hel96] eingegangen. Um neben dem theoretischen Wissen auch praktische Erfahrungen bei der Modellierung von Anwendungsfällen zu erlangen, wurde mit einigen Studenten des Studiengangs Informatik ein entsprechendes Praxisseminar durchgeführt. In diesem sollten sie ihr bereits in den entsprechenden Lehrveranstaltungen erworbenes theoretisches Wissen zu Expertensystemen, Wissensrepräsentation, Wissensverarbeitung und Planung vertiefen und praktisch anwenden.

Wir wählten die wissensbasierte Steuerung technischer Prozesse als Anwendungsgebiet aus. Modelliert werden sollte der technische Prozeß selbst, eine autonome Prozeßsteuerungskomponente sowie Einflußmöglichkeiten des Nutzers auf die Steuerung. Außerdem sollten mögliche Konflikte zwischen automatischer Steuerung, Nutzervorgaben und den physikalischen Grenzen der technischen Anlage vom System selbst erkannt und möglichst „günstig“ beseitigt werden. Wir werden darauf im Kapitel 2.2.4 genauer eingehen.

Realisiert werden sollte dieses Vorhaben von den Studenten im Hauptstudium ohne Vorkenntnisse über das zu verwendende Werkzeug innerhalb eines einsemestrigen Praxisseminars. Aus diesem Grund war es nötig, von einer realistischen Prozeßsteuerungsaufgabe zu abstrahieren. Im Seminar wurde deshalb bei der Erörterung und Diskussion der zugrundeliegenden Domäne (wissensbasierte, interaktive Steuerung eines technischen Prozesses) auf wesentlich mehr Prozeßgrößen, Problematiken und Zusammenhänge eingegangen, als dann aus Zeitgründen mit dem wissensbasierten Werkzeug modelliert werden konnte.

Nachfolgend werden die im Praxisseminar umgesetzten Modellierungsaufgaben sowie die erstellte Spezifikation vorgestellt. Dabei auftretende Besonderheiten sowie der Lösungsansatz werden diskutiert.

2 Die Modellierungsaufgabe

2.1 Die verfahrenstechnische Anlage im Überblick

Modelliert wird eine vereinfachte Prozeßsteuerung aus der chemischen Industrie. Eine Steuerungskomponente sowie der Nutzer steuern den im Reaktor ablaufenden chemischen Prozeß. Es gibt mehrere Prozeßparameter, über die der Zustand im Reaktor beinflußt werden kann. Beispielsweise wird durch Erhöhen der Heizung die Temperatur im Reaktor vergrößert. Der aktuelle Zustand im Reaktor, zum Beispiel die aktuelle Temperatur, wird mit Hilfe von Meßsensoren ermittelt, deren Werte als Beobachtungsergebnisse an einer zentralen Meßwertsammelstelle zur Verfügung stehen. Beim Überschreiten festgelegter Grenzwerte der einzelnen Prozeßgrößen werden automatisch entsprechende Alarme aktiviert. Wir unterscheiden folgende Alarmstufen: (1) gelber Alarm – der Zustand im Reaktor ist kritisch, besondere Vorsicht ist geboten und (2) roter Alarm – Lebensgefahr. Werden die Grenzwerte im Reaktor durch geeignete Steuerung des Prozesses wieder unterschritten, so werden auch die Alarme deaktiviert. Eine gute Beschreibung verfahrenstechnischer Steuerungen (wissensbasierte Prozeßführung), deren Problematiken, zu berücksichtigenden Größen sowie auch besonders die Berücksichtigung von Zeit findet man unter anderem in [Arn96].

Das Ziel der Prozeßsteuerung ist es, den Prozeß so zu steuern, daß gerade noch kein kritischer Zustand erreicht wird (maximale Auslastung der Anlage). Nicht modelliert wurden zufällige Störgrößen und Defekte der technischen Anlage. Eine Übersicht über die Modellierungsaufgabe gibt Abbildung 1.

Der chemische Prozeß im Reaktor wird vorrangig vom Nutzer gesteuert. Zu seiner Information müssen die aktuellen Meßwerte und Alarme auf dem Bildschirm angezeigt werden. Die autonome Prozeßsteuerungskomponente generiert, basierend auf den aktuellen Meßergebnissen, Vorschläge für eine möglichst günstige Steuerung (einen Aktionsplan). Diese Vorschläge werden dem Nutzer angezeigt und können von ihm bestätigt oder überschrieben werden. Somit steuert der Nutzer den ablaufenden Prozeß, allerdings nur solange, wie dadurch kein kritischer Zustand im Reaktor erreicht wird (Details folgen in Kapitel 2.2.4).

Die Auswirkungen der vom Nutzer gewählten Steuerungsaktionen werden mittels einer Simulationskomponente prognostiziert (Plansimulation). Will der Nutzer den Prozeß in einen „kritischen Bereich“ (aktiver gelber oder roter Alarm) steuern, soll ihm dies vom Simulationssystem mitgeteilt werden. Außerdem darf der entsprechende Plan (die geplanten Aktionen) nicht ausgelöst werden. Stattdessen sind die Auswirkungen der durch die autonome Prozeßsteuerungskomponente generierten Vorschläge vorherzusagen (wieder mittels Simulation). Sollten diese vorgeschlagenen Aktionen nicht in einen kritischen Bereich führen, werden sie ausgeführt. Andernfalls, d.h. haben weder Nutzer noch automatische

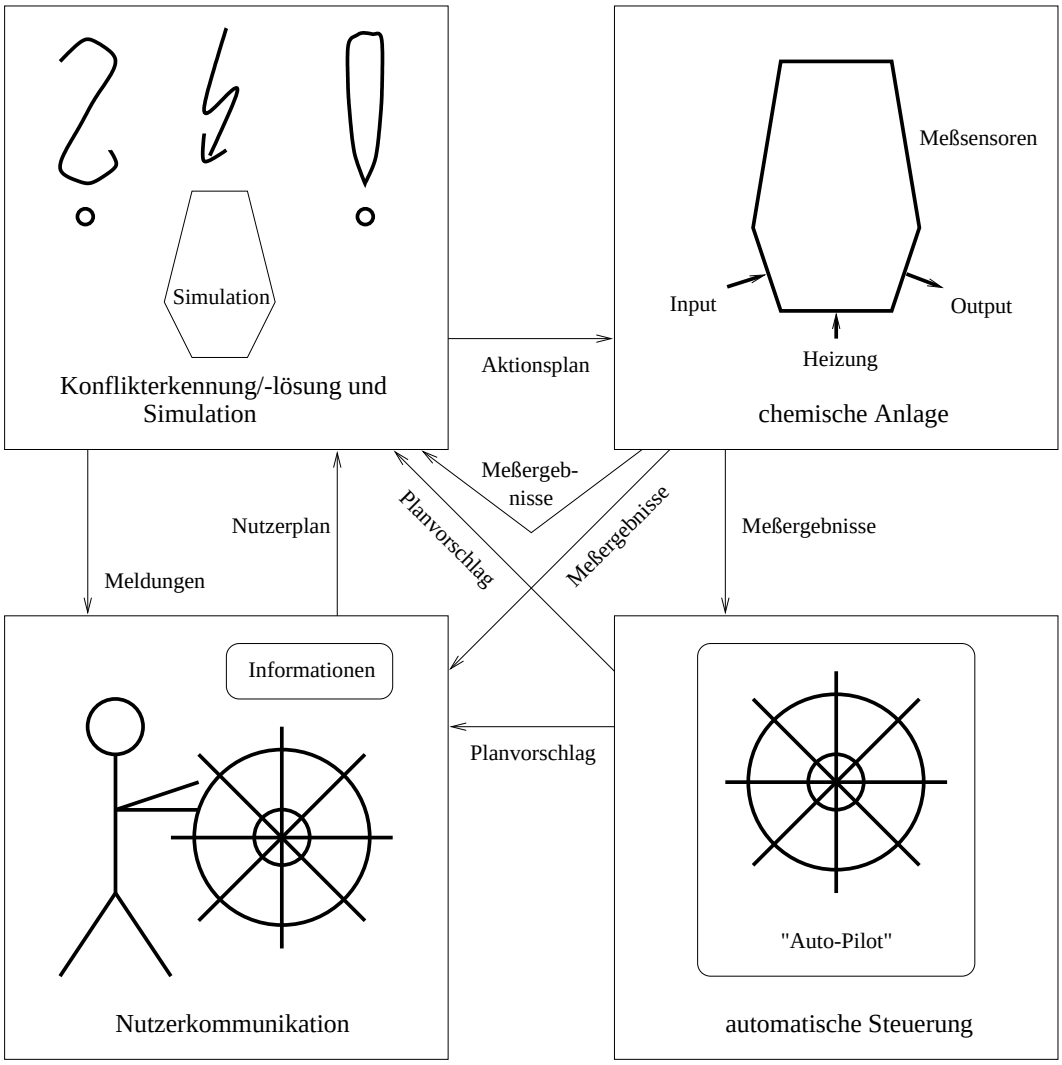

Abbildung 1: Übersicht über die Modellierungsaufgabe

Prozeßsteuerungskomponente den Prozeß „unter Kontrolle“, ist ein Standardplan auszuführen. Durch die Aktionen dieses Standardplans wird der chemische Prozeß entweder in einen sicheren Zustand zurückgeführt oder die Anlage „heruntergefahren“. Anschließend wird die Prozeßführung wieder vom Nutzer bzw. der automatischen Prozeßsteuerungskomponente übernommen.

2.2 Details der Modellierungsaufgabe

2.2.1 Die chemische Anlage

Da die Zeit innerhalb des Praxisseminars nicht ausreichte, um eine wirklich realitätsnahe Steuerung einer chemischen Anlage zu modellieren, wurden die zu-

grunde liegenden Modelle von Reaktor, Steuerung, etc. stark vereinfacht. Wir konzentrierten uns deshalb beispielsweise nur auf einige Prozeßparameter und Steuerungsmöglichkeiten, verzichteten auf die Berücksichtigung von Zeit und ließen mögliche Störgrößen außer acht.

Gemessen wird im Reaktor nur die Temperatur und der Druck. Weitere Parameter wie z.B. Füllstand werden nicht berücksichtigt. Druck und Temperatur können jeweils nur die Werte *low*, *medium* und *high* annehmen. Der Zustand im Reaktor wird nur über die drei Prozeßparameter *Heizung*, *Massezuführung* (In Mass) und *Masseentnahme* (Out Mass) beeinflußt. Diese drei Parameter lassen sich jeweils nur in Schritten von einer Einheit nach oben (durch die Aktion *inc*) oder unten (durch die Aktion *dec*) verändern. Die Meßergebnisse für die Reaktortemperatur und den Druck stehen an der zentralen Meßwertsammellstelle am Reaktor zur Verfügung.

Übersteigen Temperatur und Druck bestimmte Grenzwerte (der Prozeß kommt in einen kritischen Bereich; Grenzwerte siehe im Anhang, Tabellen 2, 5 oder 8), so wird „gelber“ bzw. „roter Alarm“ signalisiert. Hat sich der Zustand im Reaktor wieder normalisiert, werden die Alarme abgeschaltet. Die aktuellen Daten für den Druck, die Temperatur und eventuell gaschaltete Alarme sind ständig für den Nutzer sichtbar.

Der aktuelle Druck im Reaktor hängt von dem bisherigen Druck und der durch die Aktionen verursachten Druckänderung ab, analoges gilt für die Temperatur. Für die Zusammenhänge von Heizung, Massezuführung, Masseentnahme, Druckänderung und Temperaturänderung verwenden wir drei verschiedene Modelle, auf die in den nachfolgenden Abschnitten näher eingegangen wird.

Da das zur Modellierung verwendete Werkzeug die Möglichkeit bietet, eine lauffähige Implementation zu erzeugen, müssen auch geeignete Maßnahmen (Bedingungen, Aktionen, Steuerungen) bei der Modellierung vorgesehen werden, die zur Initialisierung bei Start der Simulation nötig sind. Beim „Anfahren“ einer chemischen Anlage herrscht schließlich auch keine Arbeitstemperatur im Reaktor, sondern diese muß erst durch eine geeignete Prozeßsteuerung erreicht werden.

Reaktormodell 1

Temperatur und Druck im Reaktor sind voneinander unabhängig. Die Druckänderung hängt nur von den Aktionen für In Mass bzw. Out Mass, und die Teperaturänderung nur von der Heizung ab. Die Größe der jeweiligen Änderung bezüglich der zur Steuerung ausgelösten Aktionen sowie die Grenzwerte für die Alarme werden im Anhang A, Tabellen 2, 3 und 4 beschrieben.

Hier im Reaktormodell 1 kann die Temperatur in einem Schritt (beim einmaligen Auslösen von Aktionen) nur um höchstens eine Einheit fallen (high nach medium oder medium nach low) bzw. steigen (low nach medium oder medium

nach high). Hingegen kann der Druck um maximal zwei Einheiten fallen bzw. steigen.

Reaktormodell 2

Auch im Reaktormodell 2 kann die Temperatur beim einmaligen Auslösen von Aktionen höchstens um eine Einheit fallen bzw. steigen, der Druck hingegen wieder um maximal zwei Einheiten.

Die aktuelle Temperaturänderung wird nur durch die Aktionen für die Heizung beeinflußt. Die Größe der Druckänderung ist nun aber abhängig von den Aktionen für In Mass und Out Mass sowie zusätzlich von der aktuellen Temperaturänderung. Details sowie die Grenzwerte für die Alarme werden im Anhang B, Tabellen 5, 6 und 7 beschrieben.

Reaktormodell 3

Temperatur und Druck im Reaktor lassen sich weiterhin nur über die drei Prozeßparameter Heizung, In Mass und Out Mass beeinflussen. Die Größe der Druckänderung ist wie bei Reaktormodell 2 abhängig von den Aktionen für In Mass und Out Mass sowie der aktuellen Temperaturänderung. Nun ist aber auch die Größe der aktuellen Temperaturänderung von den Aktionen für die Heizung und der aktuellen Druckänderung abhängig. Das bedeutet, selbst bei konstanter Heizung kann die Temperatur durch große Druckänderungen beeinflußt werden. Damit wäre es sogar möglich, die Temperatur im Reaktor zu steuern, ohne den Parameter Heizung zu benutzen. Den konkreten Zusammenhang finden Sie in Anhang C, Tabellen 8 und 9.

2.2.2 Die automatische Steuerung

Wir wollen Möglichkeiten des verwendeten Werkzeuges DESIRE (siehe Kapitel 3) zum Entwurf von Multiagentensystemen nutzen, um mehrere Agenten so zu steuern, daß sie gemeinsam an der Steuerungsaufgabe beteiligt sind. Einführungen in Multiagentensysteme findet man z.B. in [Rie94], [Pet97] und [Sch98]. Vorgehensweisen beim Entwurf von intelligenten Multiagentensystemen werden in [BJT96a] beschrieben.

Eine der Möglichkeiten eines Agenten ist es, auf Einflüsse seiner Umwelt zu reagieren. In [Mül96] werden diese Agenten *behaviour_based* und in [WJ95] und [BJT96a] *reactive* genannt. Sie agieren oft aufgrund geringer Information und einfacher Aktionsregeln. Ihre Informationen erhalten sie von anderen Agenten oder durch „Beobachten“ ihrer Umwelt. Sie reagieren auf Änderungen in „ihrer Welt“ und können durch Aktionen selbst aktiv auf diese Veränderungen Einfluß

nehmen. Wir werden solche *behaviour_based Agents* verwenden, um die automatische Prozeßsteuerung zu modellieren.

Für die automatische Prozeßsteuerung sollten zwei Agenten verwendet werden. Da aber aus Zeitgründen auf die Modellierung von Interaktionen der Agenten untereinander verzichtet werden mußte, wurden nur bei Verwendung von Reaktormodell 1 zwei Agenten benutzt. Das war möglich, da bei Reaktormodell 1 Druck und Temperatur unabhängig voneinander sind. Somit konnte der eine Agent die Temperatur- und der andere die Drucksteuerung übernehmen. Bei den Reaktormodellen 2 und 3 wäre durch die gegenseitige Beeinflussung von Temperatur- und Druckänderung eine Kommunikation zwischen beiden Agenten unvermeidbar. Deshalb steuert dann ein Agent Druck und Temperatur.

Die Steuerungskomponente (ein Agent bzw. mehrere Agenten) muß die jeweils aktuellen Temperatur- und Druckdaten des Reaktors erhalten. Mit Hilfe einer Wissensbasis kann nun der jeweilige Agent Steuerungsaktionen planen, die dann dem Nutzer vorgeschlagen werden. Ein solcher Aktionsplan besteht aus höchstens drei Aktionen – je eine für die Parameter Heizung, In Mass und Out Mass. Sollte der Nutzer dem Plan zustimmen, wird er, falls die Aktionen nicht in einen kritischen Zustand führen, ausgelöst und beeinflußt den Zustand im Reaktor. Es gibt für die Parameter Heizung, In Mass und Out Mass jeweils die Aktionen *inc* (Erhöhen) und *dec* (Verringern). Wird weder die Aktion inc noch die Aktion dec für einen Parameter ausgelöst, so soll der derzeitige Wert beibehalten werden (no change).

Würde der vorgeschlagene Plan den Prozeß in einen kritischen Zustand führen (gelber oder roter Alarm), verhindert die Konfliktlösungskomponente die Ausführung und startet gegebenenfalls einen Standardplan zum sicheren „Herunterfahren“ der Anlage (siehe Abschnitt 2.2.4).

Die Aktionsplanung ist STRIPS-artig. Es gibt Vorbedingungen für die Aktionen, die erfüllt sein müssen, um die Aktion auszulösen. Durch die festgelegten Nachbedingungen jeder Aktion, läßt sich anhand des aktuellen Reaktorzustandes der neue Zustand nach Ausführung der Aktionen berechnen. Betrachtet wird immer nur der aktuelle Reaktorzustand, der durch die Aktionen verändert wird.

Im allgemeinen versteht man unter Planung die Suche nach Aktionen, die aus einem beschriebenen Ausgangszustand einen gewünschten Zielzustand erzeugen. Das Ergebnis der Planung ist eine Folge von Aktionen (ein Plan). In unserer Anwendung beispielsweise die Folge der drei Aktionen *inc(Heizung), dec(In Mass), inc(Out Mass)*. Durch die Ausführung dieses Planes würde bei zugrunde gelegtem Reaktormodell 1 die Temperatur ansteigen und der Druck stark sinken.

Die Anfänge des Planens gehen auf die Entwicklung des Situationskalküls von McCarthy und Hayes (vergl. [MH68] oder [MH69]) zurück. Grundelemente dieses Kalküls sind Situationen und Operatoren. Eine Situation stellt einen

Schnappschuß des interessierenden Weltausschnitts dar (in unserem Fall der Zustand im Reaktor), und sie wird beschrieben durch eine Menge gültiger logischer Formeln. Ein Operator entspricht der formalen Beschreibung einer Handlung und überführt eine gegebene Situation in die Nachfolgesituation. Auf dieser Vorgehensweise, Handlungen und die durch sie vollzogenen Veränderungen zu modellieren, basiert das klassische Planen. Grundlegende Arbeiten dazu sind beispielsweise [FN71] (STRIPS Grundlage), [Lif87] (STRIPS Semantik auf Grundlage der Prädikatenlogik 1. Stufe), [Tat77], [Sac77] und [Cha87]. Eine der wichtigsten Eigenschaften dieser Planungsansätze ist neben der Definition von Aktionsoperatoren mit Vor- und Nachbedingung als Menge von Klauseln deren Umgang mit der Zeit, die als Vorher und Nachher von Situationen betrachtet wird. Des weiteren gibt es auch eine große Palette „nichtklassischer“ Planungsansätze, die sich beispielsweise durch ein anderes Zeitmodell ([Ver83], [All83], [All84]), Unvollständigkeit der Information ([DW91]), verteiltes Planen ([BG88], [LB89], [Mar93]) oder fallbasiertes Planen ([KH92]) abheben.

STRIPS-artige Planungsansätze sind relativ eingeschränkt und erlauben nicht die Behandlung bestimmter dynamischer Aspekte. Für komplexe Aufgaben, wie beispielsweise eine wissensbasierte Steuerung eines realen technischen Prozesses, sind flexiblere Ansätze (vgl. [AJ94]) erforderlich, die vermutlich weit über bisherige Ansätze hinaus gehen werden.

Da es uns hier bei der Modellierung aber nicht um die Verwirklichung eines ausgefeilten Ansatzes zur Aktionsplanung geht, sondern um die Anwendung und Übung von Prinzipien der Wissensmodellierung, beschränkten wir uns bei der Aktionsplanung auf einen STRIPS-artigen Ansatz. Wir lassen auch außer acht, daß die Aktionen im allgemeinen eine Ausführungszeit besitzen. Beispielsweise ist sofort nach Erhöhung der Heizung (inc(Heizung)) die Temperatur angestiegen – es dauert also nicht wie in der Realität eine gewisse Zeit, bis die Auswirkungen der Aktionen beobachtbar sind.

2.2.3 Nutzerkommunikation und manuelle Steuerung

Zur Kontrolle des Prozeßablaufes während einer Simulation werden die einzelnen Schritte (Beobachtungen in der Welt, Inferenzergebnisse, benutzte Regeln, getroffene Annahmen, aktive Module, Datenübergaben, usw.) auf dem Bildschirm protokolliert. Außerdem werden die bereits ausgelösten Aktionen zur Steuerung (der Aktionsplan), der aktuelle Zustand der Welt (Reaktor) sowie die Alarmzustände für den Nutzer auf dem Bildschirm angezeigt.

Der durch die automatische Steuerung erstellte Aktionsplan wird dem Nutzer vorgeschlagen. Dieser kann die einzelnen Aktionen jeweils bestätigen oder eine eigene Auswahl treffen. Da die aktuellen Daten für Druck und Temperatur ständig angezeigt werden, sollte eine manuelle Aktionsplanung nicht schwer fallen. Noch

dazu, wenn nur derart kurze Pläne (zum Erreichen der nächsten Situation im Reaktor) erstellt werden und bereits ein Plan vorgeschlagen wird. Wir wollen aber nicht voraussetzen, daß der vorgeschlagene Plan bereits eine optimale Steuerung bewirkt.

Der Nutzer sowie die automatische Steuerungskomponente erstellen ihren Aktionsplan durch Planen einer Folge von Aktionen. Für jede Aktion läßt sich anhand der Gültigkeit der Vorbedingungen testen, ob sie anwendbar ist oder nicht. Die Ausführbarkeit des gesamten Plans wird jedoch erst in der Komponente zur Konflikterkennung überprüft. So könnten beispielsweise die Anwendungsbedingungen für inc(Heizung) und inc(In Mass) im einzelnen erfüllt sein, aber die Ausführung des gesamten Plans führt in einen kritischen Zustand (wegen der Wechselwirkungen von Druck- und Temperaturänderung in Reaktormodell 3).

Da der Nutzer die Auswirkungen seiner geplanten Aktionen in Gedanken „vorausberechnen“ kann, ist es ihm prinzipiell möglich, qualitativ bessere Pläne zu generieren als die automatische Steuerung. Um die Steuerungskomponenten dahingehend nicht zu benachteiligen, müßte man bereits bei der automatischen Planerzeugung eine inkrementelle Planung mit Plansimulation und Bewertung der Aktionen vorsehen. Das hätte aber den Rahmen des durchgeführten Seminars bei weitem gesprengt.

2.2.4 Konflikterkennung und -lösung

In der Konflikterkennung soll die Anwendbarkeit der erstellten Pläne geprüft werden. Dazu muß prognostiziert werden, wie sich der Zustand im Reaktor bei Ausführung der jeweiligen Pläne verändern würde. Mit Hilfe der notwendigen Informationen über den aktuellen Zustand im Reaktor und den beiden Plänen wird deren Ausführung simuliert. Der Plan des Nutzers und der Plan der automatischen Prozeßsteuerungskomponente werden aufgrund der durch die Plansimulation erhaltenen Vorhersagen bewertet.

Ein Plan gilt als ausführbar, wenn er nicht in einen Alarmzustand (gelber oder roter Alarm) führt. Für den Fall, daß beide Pläne nicht ausführbar sind, wurden Standardpläne definiert, die dann ausgeführt werden sollen. Durch den Standardplan I (inc(Out Mass), dec(Heizung)) werden Druck und Temperatur im Reaktor nur leicht verringert, so daß ein kritischer Reaktorzustand verlassen wird. Bei Ausführung von Standardplan II (inc(Out Mass), dec(In Mass), dec(Heizung)) werden Reaktordruck und -temperatur so weit wie möglich abgesenkt.

Unter welchen Bedingungen welcher Plan bevorzugt und somit ausgeführt wird, zeigt Tabelle 1.

Wie man sieht, bevorzugen wir den Plan des Nutzers, falls keiner der beiden Pläne zu einem Alarm führt. Diese Strategie verfolgen wir, weil der Nutzer die Auswirkungen seiner geplanten Aktionen in Gedanken „vorausberechnen“

		Nutzeraktionen führen zu		
		kein Alarm	gelber Alarm	roter Alarm
Agenten aktionen führen zu	kein Alarm	Plan des Nutzers	Plan der Steuerung	Plan der Steuerung
	gelber Alarm	Plan des Nutzers	Standard-plan I	Standard-plan I
	roter Alarm	Plan des Nutzers	Standard-plan I	Standard-plan II

Tabelle 1: Aktionsauswahl durch die Konfliktlösung

kann, er also möglicherweise qualitativ bessere Pläne erstellt, als die automatische Steuerung.

Schließlich soll der Nutzer eine Rückmeldung erhalten, welcher Plan zur Ausführung ausgewählt wird. Wird nicht der Plan des Nutzers ausgeführt, so ist dies durch Angabe des sonst entstehenden Alarmzustandes zu begründen.

Auch für den Nutzer wäre eine interaktive, inkrementelle Planung mit Simulation und Bewertung der erstellten Teilpläne vorteilhaft, konnte aber aus Zeitgründen nicht modelliert werden.

3 Allgemeines zum verwendeten Werkzeug

Wir verwendeten zur Modellierung das wissensbasierte Werkzeug DESIRE (framework for **DE**sign and **S**pecification of **I**nteracting **RE**asoning components), welches nachfolgend kurz vorgestellt wird. Neben dem leider etwas spärlichen User-Manual [Art97] lassen sich [BJT96b] und [BJT96a] sowohl als Nachschlagewerk als auch für eine Einführung zu DESIRE verwenden.

Die Gründe für die Auswahl von DESIRE waren vielfältig. An dieser Stelle seien stellvertretend nur ein paar von ihnen sowie einige „Nebenbedingungen“ organisatorischer Natur genannt:

- sehr komfortabler grafischer Editor,
- ständig erreichbare Online-Hilfe,
- strikte Trennung von Wissen, Metawissen und Steuerungswissen,
- Verwendung einer mehrsortigen, dreiwertigen Prädikatenlogik,
- Möglichkeiten der Vererbung innerhalb der angelegten Modulhierarchie,

- Möglichkeit der automatischen Generierung einer lauffähigen Implementation aus der erstellten Spezifikation,
- gut ausgebauter Debugger zur Fehlersuche bei gestarteter Implementation,
- gute Kontakte zu den Entwicklern von DESIRE,
- kostenlose Bereitstellung des Systems.

Das zur Modellierung verwendete System DESIRE wurde an der Freien Universität Amsterdam entwickelt und wird dort auch ständig weiterentwickelt. Anfangs wurde DESIRE zur Spezifikation beliebiger wissensbasierter Systeme entwickelt und benutzt. Seit einigen Jahren wird DESIRE jedoch speziell für den Entwurf intelligenter Multiagentensysteme weiterentwickelt. Um die Performance der mit DESIRE erzeugten lauffähigen Implementationen weiter zu verbessern, ist eine Portierung auf eine parallele Plattform in Arbeit.

Das System DESIRE wurde auf der Grundlage von XPCE entwickelt. XPCE ist eine Grafik unterstützende Erweiterung von SWI-Prolog. Die Grafikmöglichkeiten von XPCE wurden bei der Implementation von DESIRE benutzt.

DESIRE beinhaltet einen grafischen Editor (genannt *destool*) und einen Generator zur Erzeugung einer Implementation. Eine Implementation ist innerhalb der DESIRE-Entwicklungsumgebung ausführbar, so daß der modellierte Sachverhalt (beispielsweise unsere interaktive Prozeßsteuerung) sofort simuliert werden kann. Dazu verwendet DESIRE XPCE bzw. Prolog. Es besteht jedoch auch die Möglichkeit, externe, ausführbare Programme in den DESIRE-Modulen als Aktionsskript (unter Berücksichtigung einer entsprechenden Parameterübergabe) zu spezifizieren. Diese Möglichkeit benutzen wir beispielsweise zur Nutzerkommunikation, die sonst in DESIRE nur eingeschränkt möglich ist.

3.1 Unterstützung der Wissensspezifikation

DESIRE unterstützt die Spezifikation von Multiagentensystemen durch Modellierung und Wissensspezifikation mittels:

- Aufgabenzerlegung (task (de)composition)
- Informationsaustausch (information excange)
- verteilter Steuerung (control (de)composition)
- Aufgabenzuweisung (task delegation)
- Wissensverteilung/-strukturierung (knowledge (de)composition)

Nachfolgend wollen wir diese 5 Punkte kurz umreißen. Ausführliche Erläuterungen findet man z.B. in [BJT96b], [BJT96a] oder [TW92].

3.1.1 Aufgabenzerlegung (task (de)composition)

Eine der Vorgehensweisen beim Entwurf eines komplexen Systems ist die top-down Strategie. Hierbei wird eine komplexe Aufgabe solange zerlegt, bis man zu kleinsten „unteilbaren“ Teilaufgaben kommt.

In DESIRE wird jede Teilaufgabe durch ein Modul (*component*) repräsentiert. Es gibt zwei Arten von Modulen, komplexe (*composed*) und primitive. Primitive Module entsprechen den unteilbaren Aufgaben. Komplexe Module bestehen wiederum aus komplexen oder auch primitiven Modulen. Für alle Module (komplex bzw. primitiv) müssen Eingaben und Ausgaben spezifiziert werden. Dies kann man durch Definition von *input* und *output information types* des entsprechenden Moduls, seiner Signatur, tun. Zur Definition der Signaturen wird die bereits genannte mehrsortige Logik verwendet.

3.1.2 Informationsaustausch (information excange)

Der Informationsaustausch zwischen den einzelnen Modulen erfolgt über *Links*. Das sind Verbindungen zwischen Aus- und Eingängen von Modulen. Bei der Spezifikation von Links ist zu definieren, welche Art von Information transferiert werden soll. Einerseits sind es Daten, die ausgetauscht werden und andererseits Steuerungsinformationen, die die Zusammenarbeit der verschiedenen Module beeinflussen.

3.1.3 Verteilte Steuerung (task control)

Die Steuerung definiert die zeitlichen Beziehungen zwischen Informationsaustausch und Wissensverarbeitung innerhalb der Module. Es wird gesteuert, welche Module unter welchen Bedingungen aktiviert werden, wann Informationen übertragen werden und welche Ziele bei der Wissensverarbeitung in den Modulen zu verfolgen sind.

3.1.4 Aufgabenzuweisung (task delegation)

Hier kann festgelegt werden, ob das System selbst die Aufgabe lösen soll (z.B. Ableiten oder Widerlegen von Aussagen), oder ob der Nutzer für diese Aufgabe (z.B. Dateneingabe, Annahmen für gültig erklären, Auswahl zu verfolgender Ziele) verantwortlich ist.

3.1.5 Wissensstrukturierung (knowledge (de)composition)

Es muß festgelegt werden, wo welche Arten von Wissen (input/output Information, Information zur Steuerung, ...) benötigt werden. Es kann angegeben werden,

ob innerhalb eines Moduls eine eigene Wissensbasis, ein externes Programm oder ein Berechnungsalgorithmus für den Inferenzprozeß verwendet werden soll.

3.2 Destool – der grafische Editor von DESIRE

In dem grafischen Editor *destool* kann man durch einfaches Klicken mit der Maus Module und Links anlegen oder modifizieren. Die Editoren für Wissensbasis, Task control (Steuerungswissen), Signatur, usw. können durch Klicken auf die entsprechenden Edit-Buttons aktiviert werden. Durch Anklicken von Modulen oder Links mit der *rechten* Maustaste erhält man ein kontextsensitives Menü der zur Verfügung stehenden Optionen. Beispielsweise öffnet man einen *Linkeditor* zu einem bestimmten *Link*, indem man diesen mit der rechten Maustaste anklickt und *edit* wählt.

Die Bedienung der Texteditoren (z.B. für Wissensbasis oder Steuerungsregeln) wurde an die Bedienung des Editors *emacs* angelehnt. Eine Übersicht über die Funktionalität (copy, paste, delete, load, save, ...) der Editoren findet man im User-Manual [Art97] .

3.3 Fehlerwerkzeuge, Online-Hilfe

Mit der DESIRE-internen Funktion *Check* wird der gesamte bisherige Entwurf auf Syntaxfehler und z.B. undefinierte Relationen getestet. Per Doppelklick auf die ausgegebenen Fehlermeldungen wird die als fehlerhaft bezeichnete Definition in dem entsprechenden Editor geöffnet und die betreffende Zeile farblich markiert.

Im Gegensatz zu dieser sehr hilfreichen Unterstützung bei der Fehlersuche und -beseitigung steht der fehlende Test der Wissensbasen auf Inkonsistenzen. Bei großen Wissensbasen könnte dies zu unerwarteten Ergebnissen beim Inferenzprozeß führen. In unserem Anwendungsfall (interaktive Prozeßsteuerung) konnten die verschiedenen Wissensbasen jedoch so klein gehalten werden, daß sie für den Entwickler überschaubar blieben.

Zur Laufzeit (nach Start einer Implementation) kann der bereits genannte Debugger verwendet werden – leider nur im Textmodus. Neben den umfangreichen Debugfunktionen bietet DESIRE auch eine Verknüpfungsmöglichkeit für Debugger und grafischem Editor. Dadurch werden zur Laufzeit die im Debugger gerade aktiven Module/Links im grafischen Editor farblich markiert, hier wäre dann allerdings die Anzeige der gesamten Modulhierarchie wünschenswert. Ein Ablauftrace wird in einem separaten Fenster erzeugt und kann auf Wunsch in einer Datei gespeichert werden.

Eine kurze Onlinehilfe ist fast immer über ein kontextsensitives Menü erreichbar.

4 Der erstellte Entwurf

Es wurde eine kompositionale Spezifikation mit zwei Hierarchieebenen erstellt. Die oberste Ebene (Top Level) zeigt Abbildung 2. Man findet die vier Kompo-

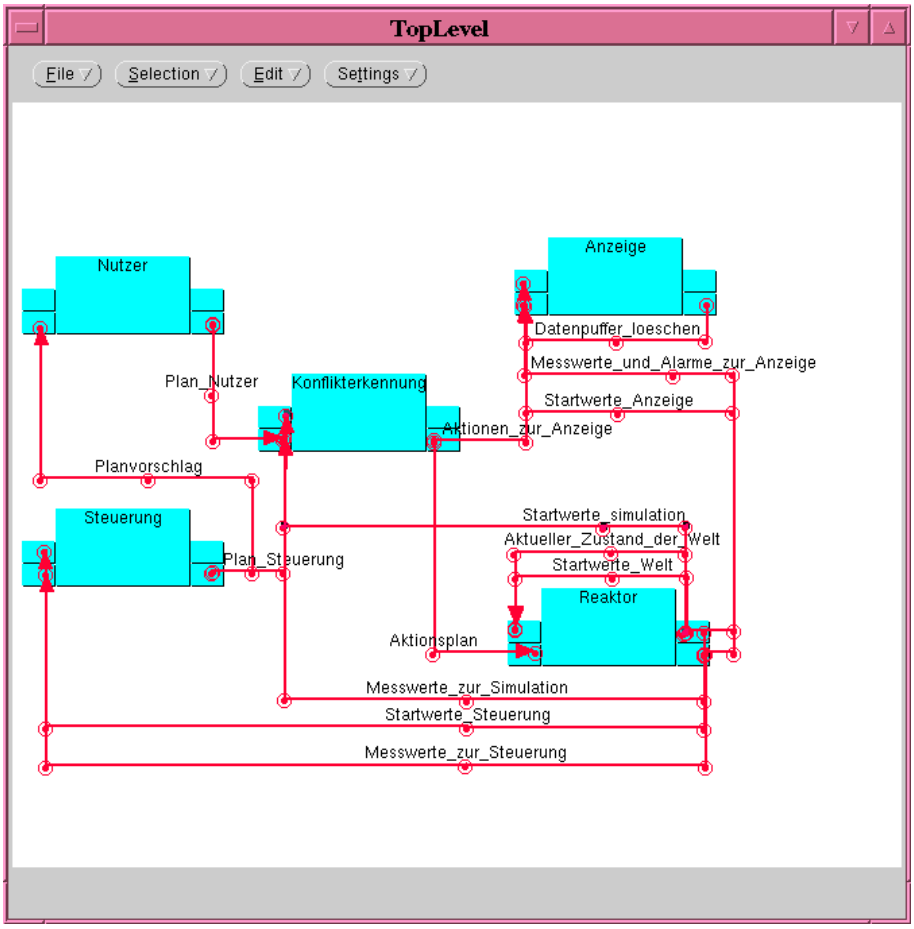

Abbildung 2: Die oberste Ebene des Entwurfes

nenten (Module) der Aufgabenstellung aus Abbildung 1 wieder. Die ständige Anzeige der aktuellen Daten aus dem Reaktor wurde mittels einem zusätzlichen fünften Modul modelliert.

Für den Modul *Reaktor* wurden entsprechend den verschiedenen Reaktormodellen drei separate Wissensbasen erstellt, die je nach zu betrachtendem Modell geladen werden können. Es bot sich an, Regeln zu verwenden, die Variablen enthalten. Diese werden erst im Inferenzprozeß während einer Simulation instantiiert – dadurch blieb die Wissensbasis kompakt. Durch die Links zur Initialisierung werden beim Start einer Simulation die Werte für Druck und Temperatur im Reaktor auf *low* gesetzt. Die Initialwerte sowie die aktuellen Meßwerte für Druck und Temperatur werden systemweit zur Verfügung gestellt.

In dem Modul *Steuerung* verbergen sich auf tieferer Ebene die Agenten zur automatischen Plangenerierung. Sie erstellen den Aktionsplan, der dem Nutzer vorgeschlagen wird. Außerdem wird der Plan dem Konflikterkennungsmodul übergeben, damit die Planausführung simuliert werden kann.

Dem Nutzer wird der vorgeschlagene Plan in einem separaten Fenster auf dem Bildschirm übersichtlich dargestellt (siehe Abbildung 3). Mit Hilfe weniger

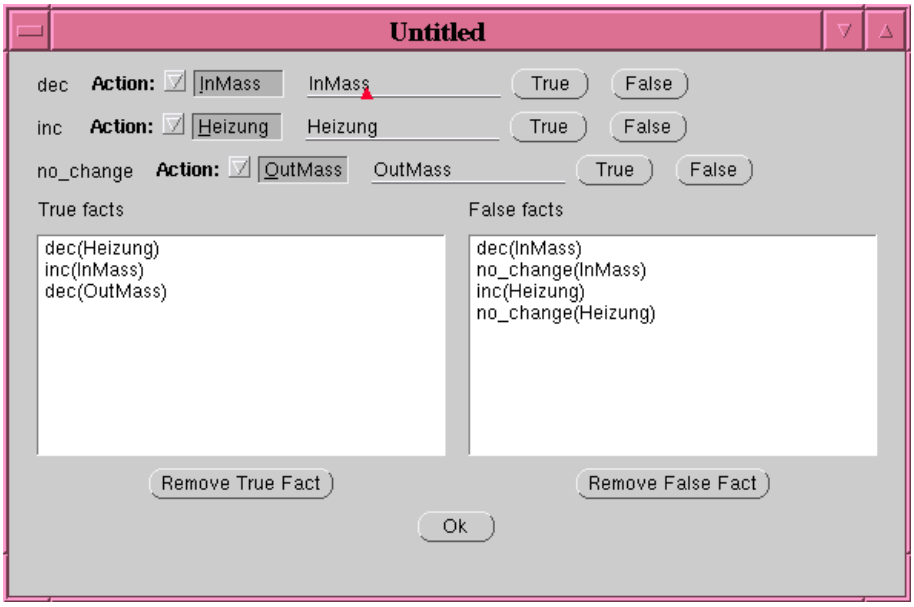

Abbildung 3: Kommunikationsmodul für den Nutzer

Mausklicks kann der Nutzer die vorgeschlagenen Aktionen löschen und eigene Vorgaben in die Aktionsliste eintragen. Diese komfortable Aktionsauswahl wurde mittels eines allgemeinen externen Kommunikationsmoduls realisiert, welches uns für diesen Zweck von den Systementwicklern aus Amsterdam zur Verfügung wurde. Bestätigt der Nutzer den angezeigten Plan mit „ok“, wird auch dieser Plan zur Konflikterkennung weitergeleitet.

Die innere Architektur des Moduls *Konflikterkennung* ist in Abbildung 4 dargestellt. Innerhalb der Konflikterkennung wird auf der zweiten Modellierungsebene die Ausführung der beiden Pläne simuliert (Modul *Simulation*). Die Simulationsergebnisse sowie die zugehörigen Pläne werden dem Modul *Bewertung* übergeben. Hier wird überprüft, ob ein Plan in einen kritischen Zustand führen würde (gelber oder roter Alarm). Entsprechend den Vorgaben durch Tabelle 1 wird der „bessere“ Plan bzw. einer der vordefinierten Standardpläne zur Ausführung ausgewählt. Die Begründung, warum gerade dieser Plan ausgeführt werden soll, wird mittels des Moduls *Ausgabe* auf dem Bildschirm angezeigt.

Der ausgewählte Plan wird nun in die oberste Modellierungsebene zu den

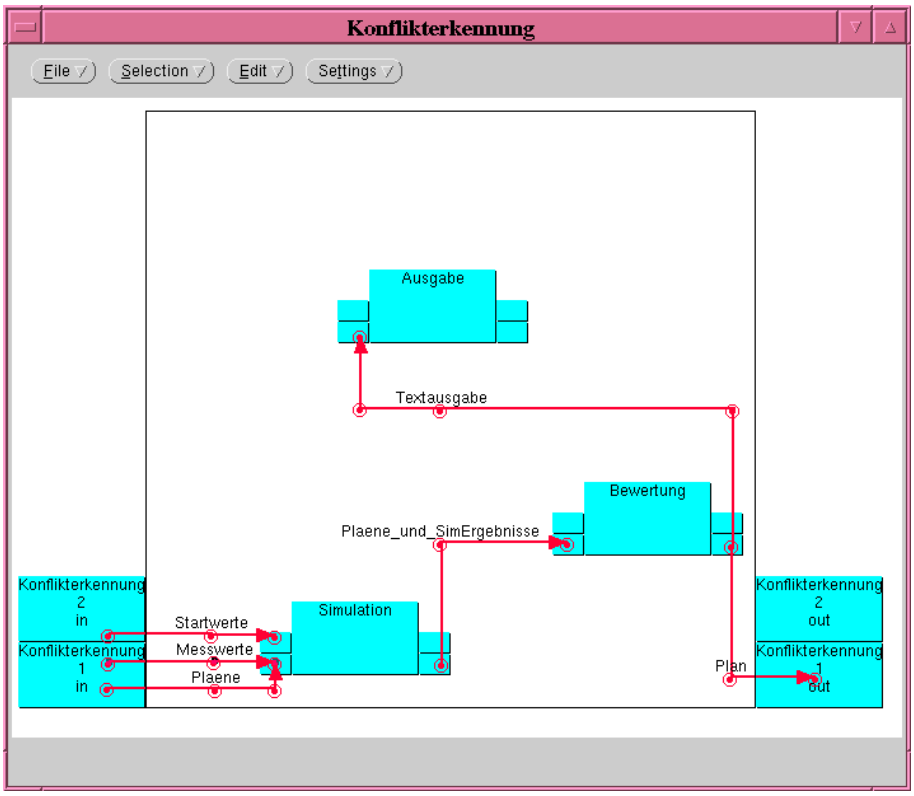

Abbildung 4: Details der Konflikterkennung

Modulen *Reaktor* und *Anzeige* übertragen. Im Anzeigemodul wird dem Nutzer der auszuführende Plan (die Aktionen) angezeigt. Dies ermöglicht ein leichtes Nachvollziehen der Änderungen des Zustandes im Reaktor. Im Modul *Reaktor* wird der Plan ausgeführt. Mögliche Druck- und Temperaturänderungen werden mit Hilfe der Wissensbasis abgeleitet. Anschließend wird der neue Zustand im Reaktor (Druck und Temperatur) „berechnet“, mit den vorgegebenen Grenzwerten für die Alarme (siehe Tabellen 2, 5 und 8 im Anhang) verglichen und diese bei Grenzwertüberschreitung ein- bzw. bei -unterschreitung ausgeschaltet.

Nun werden wieder die aktuellen Meßdaten systemweit an die betreffenden Module verteilt und die Steuerungskomponente erstellt einen Plan, der dem Nutzer vorgeschlagen wird, usw.

Alle Module enthalten eigene Wissensbasen, in denen das zur Lösung der jeweiligen Aufgabe nötige Wissen in Form von Fakten und Regeln enthalten ist. Dabei wird in DESIRE strikt zwischen Domänenwissen und Steuerungswissen getrennt. Die *komplexen* Module (Steuerung, Top-Level, Konflikterkennung) enthalten Wissensbasen zur Ablaufsteuerung, die restlichen Module das Domänenwissen. Zum Domänenwissen gehört beispielsweise, unter welchen Bedingungen und in welchem Maße die Reaktortemperatur bei Planausführung an-

steigt oder abfällt, usw. Die Steuerung betrifft den gesamten zeitlichen Ablauf, also daß nur unmittelbar nach dem Start einer Simulation initialisiert werden muß; oder daß in dem Modul *Steuerung* erst der gesamte Plan generiert wird, bevor er dem Nutzer vorgeschlagen wird; oder daß erst beide Pläne simuliert werden müssen, um sie anschließend zu übertragen und zu vergleichen; usw.

Die Trennung der Wissensarten sowie auch die Möglichkeit eines hierarchischen Modellierung unterstützen in besonderer Weise die Entwicklung eines strukturierten Entwurfes und die Wiederverwendbarkeit von vorhandenen Entwürfen.

5 Ergebnisse und Erfahrungen

Trotz der getroffenen, bereits vorn im Text genannten Einschränkungen war die Modellierung nicht trivial. Die Aufgabenstellung vereinfachte sich aber zumindest so weit, daß die Modellierung sowie die Einarbeitung in DESIRE in der zur Verfügung stehenden kurzen Zeit für die Studenten möglich war.

Es war nicht unser Ziel, eine Anlage aus der chemischen Industrie möglichst wirklichkeitsnah abzubilden – dazu hätten weitere Behälter, Rohrleitungen, Ventile, mehr Prozeßparameter, usw. gehört. Außerdem müßte für eine realistische Steuerung die Zeit sowie äußere Einflüsse (Störgrößen) in geeigneter Weise berücksichtigt werden. Auch sollten nicht besonders ausgefeilte Pläne zur Steuerung erzeugt werden – dazu wären unter anderem eine Plansimulation während der Planerstellung und überhaupt andere Planungsansätze nötig gewesen. Besonders verteiltes Planen erscheint für die Modellierung mit DESIRE gut geeignet, da DESIRE als System zum Entwurf von Multiagentensystemen eine entsprechende Unterstützung bietet. Wir wollten hier vorrangig die prinzipielle Vorgehensweise bei der Wissensmodellierung üben und festigen. Außerdem sollten die Studenten praktische Erfahrungen beim Umgang mit Werkzeugen zum Entwurf wissensbasierter Systeme sammeln und dabei eventuell auftretende Probleme kennen und vermeiden lernen.

Aber bereits auch die vorliegende kleine Modellierungsaufgabe zeigte deutlich, daß man besonders auf Modularität, Erweiterbarkeit und Wissensstrukturierung Wert legen sollte. Nur so können Wissensbasen erst recht bei komplexeren Aufgaben effektiv verwaltet und modifiziert werden. Auch die Standardisierung der Schnittstellen der einzelnen Module ist von großer Bedeutung. Dies sind nur einige aber sehr wichtige Schwerpunkte – vor allem bei komplexeren Problemstellungen, bei denen oft im Team gearbeitet wird. Auch dabei sollte DESIRE Unterstützung bieten, beispielsweise durch einen Konsistenztest für Wissensbasen. Denn schon bei Modifikationen der in unserer Anwendung verwendeten recht kleinen Wissensbasen war es nicht immer leicht, Konsistenz „per Hand“ nachzuweisen.

Ein weiteres Problem entstand durch die von DESIRE konservativ verwalteten Datenpuffer der Module. Es war an verschiedenen Stellen nötig, diese Puffer explizit zu löschen. Bei der Modellierung eines realen technischen Systems würde es aber eher der Wirklichkeit entsprechen, wenn *vorhandene* Puffer im technischen System explizit modelliert werden müssen, als wenn man *nicht vorhandene* Puffer durch zusätzliche Maßnahmen simulieren muß.

Besonders günstig machten sich die Strukturierungsmöglichkeiten des Entwurfes mittels DESIRE bemerkbar. So wurden beim „Zoomen“ in eine tiefere Modellierungsebene in dieser Ebene nicht relevante Informationen „ausgeblendet“. Schon dadurch wird die Konzentration auf das Wesentliche gelenkt. Auch die getrennte Repräsentation von Steuerungswissen erwies sich als vorteilhaft. Sehr elegant lassen sich Module und Links mit Hilfe des grafischen Editors *destool* anlegen bzw. ändern. Auch die Vererbungmöglichkeiten von Sorten, Relationen, usw. sind hilfreich.

Selbst dieser, gemessen an realen Aufgabenstellungen, sicher sehr kleine Entwurf enthält etwa 40 *information types* mit diversen Sorten und Objekten, circa 40 Relationen und über 250 Regeln. Spätestens an dieser Stelle wird deutlich, daß die bereits oben genannten Schwerpunkte (Modularität, Erweiterbarkeit, usw.) beim Entwurf wissensbasierter Systeme für praktische Anwendungen äußerst wichtig sind. Das verwendete Werkzeug muß dies in geeigneter Weise unterstützen.

DESIRE wird sicher in den kommenden Jahren zu einem leistungsfähigen Tool weiterentwickelt werden, so daß man auch bei komplexen Anwendungen entsprechende Unterstützung findet.

6 Danksagung

Ich möchte allen bei der Vorbereitung und Durchführung dieser Arbeit beteiligten Personen für ihre Unterstützung danken. Besonders Barbara Heller, die mich bei der Vorbereitung und Durchführung des Seminars tatkräftig unterstützte und den von mir in Vorbereitung des Seminars erstellten Entwurf mehrfach kritisch begutachtet hat. Weiterhin danke ich Heinrich Herre, mit Hilfe dessen guter Kontakte zu den Systementwicklern in Amsterdam uns das System DESIRE zur Verfügung gestellt wurde. Nicht vergessen möchte ich die fleißigen Studenten, die trotz der Kürze der Zeit und der mitunter mühsamen Einarbeitung in DESIRE recht gute Entwürfe erstellten.

Literatur

[AJ94] Oksana Arnold and Klaus P. Jantke. Therapy plans as hierarchically structured graphs. In *Fifth International Workshop on Graph Grammars and their Application to Computer Science, Williamsburg, Virginia, USA*, November 1994.

[All83] James F. Allen. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26(11):832–843, November 1983.

[All84] James F. Allen. Towards a General Theory of Action and Time. *Artificial Intelligence*, 23(2):123–154, 1984. (Reprinted in Allen, Hendler and Tate, (ed.), Readings in Planning, Morgan Kaufman, 1990).

[Arn96] Oksana Arnold. *Die Therapiesteuerungskomponente einer wissensbasierten Systemarchitektur für Aufgaben der Prozeßführung*, volume 130 of *DISKI, Dissertationen zur Künstlichen Intelligenz*. Infix, 1996.

[Art97] Artificial Intelligence Group, Department of Computer Science, Vrije Universiteit Amsterdam, Amsterdam, The Netherlands. *Desire Graphical Editor and General Manager System for DESIRE Specifications – User Manual*, 1997.

[BG88] A.H. Bond and L. Gasser. *Readings in Distributed Artificial Intelligence*. Morgan Kaufmann, 1988.

[BHJR94] Bodo Busch, Thomas Herrmann, Katharina Just, and Markus Rittenbruch. *Systeme für Experten statt Expertensysteme*. Infix Verlag, Sankt Augustin, 1994.

[Bib93] Wolfgang Bibel. *Wissensrepräsentation und Inferenz*. Vieweg Verlag, Braunschweig, Germany, 1993.

[BJT96a] Frances Brazier, Catholijn Jonker, and Jan Treur. *Design of Intelligent Multi-Agent Systems*. Artificial Intelligence Group, Department of Computer Science, Vrije Universiteit Amsterdam, Amsterdam, The Netherlands, 1996.

[BJT96b] Frances Brazier, Catholijn Jonker, and Jan Treur. *Syllabus - Ontwerp van Kennissystemen*. Artificial Intelligence Group, Department of Computer Science, Vrije Universiteit Amsterdam, Amsterdam, The Netherlands, 1996.

[Cha87] David Chapman. Planning for Conjunctive Goals. *Artificial Intelligence*, 32:333–377, 1987.

[DW91] T.L. Dean and M.P. Wellman. *Planning and Control*. Morgan Kaufmann, 1991.

[FN71] Richard E. Fikes and Nils J. Nilsson. STRIPS: A New Approach to Theorem Proving in Problem Solving. *Artificial Intelligence*, 2:189–208, 1971.

[GS96] Johann Gamper and Friedrich Steimann. Medizinische Expertensysteme - Eine kritische Betrachtung. *APIS - Zeitschrift für Politik, Ethik, Wissenschaft und Kultur im Gesundheitswesen*, 1996.

[Hel96] Barbara Heller. *Modularisierung und Fokussierung erweiterbarer komplexer Wissensbasen auf der Basis von Kompetenzeinheiten*, volume 120 of *DISKI, Dissertationen zur Künstlichen Intelligenz*. Infix, 1996.

[Hen91] Ralf-Dirk Hennings. *Informations- und Wissensverarbeitung – Theoretische Grundlagen Wissensbasierter Systeme*. Walter de Gruyter, Berlin, 1991.

[KH92] Subbaro Kambhampati and James A. Hendler. A Validation-Structure-Based Theory of Plan Modification and Reuse. *Artificial Intelligence*, 55:193–258, 1992.

[LB89] Gerd Lorscheid and Christian Bauer. Verteiltes Planen verteilter Problemlösungen. Arbeitspapiere der GMD 357, Gesellschaft für Mathematik und Datenverarbeitung, 1989.

[Lif87] V. Lifschitz. On the Semantics of STRIPS. In Michael P. Georgeff and Amy L. Lansky, editors, *Reasoning about Actions and Plans, Proceedings of the 1986 Workshop, Timberline, OR, June 30 - July 2, 1986*, pages 1–9, San Mateo, CA, 1987. Morgan Kaufmann Publishers Inc.

[Mar93] Frank von Martial. Planen in Multi-Agenten Systemen. In J. Müller, editor, *Verteilte Künstliche Intelligenz: Methoden und Anwendungen*, Reihe Informatik, chapter 4. BI-Wissenschaftsverlag, 1993.

[MH68] John McCarthy and Patrick J. Hayes. Some Philosophical Problems from the Standpoint of Artificial Intelligence. Report Memo AI-73, Stanford University, Departement of Computer Science, Stanford, California, November 1968.

[MH69] John McCarthy and Patrick J. Hayes. Some Philosophical Problems from the Standpoint of Artificial Intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, 1969.

[Min74] Marvin Minsky. A Framework for Representing Knowledge. Technical Report Memo No. 306, Massechusetts Institute of Technology, A.I. Laboratory, June 1974. See also: paper of the same name in Winston, P. H. (ed.) *The Psychology of Computer Vision*, (1975) New York: McGraw-Hill. Reprinted in Ronald J. Brachman and Hector J. Levesque (eds.) *Readings in Knowledge Representation*, (1985) Los Altos: Kaufman.

[Mül96] Jörg P. Müller. *The design of intelligent agents: a layered approach*, volume 1177 of *Lecture Notes in Artificial Intelligence and Lecture Notes in Computer Science*. Springer-Verlag Inc., New York, NY, USA, 1996.

[Pet97] Charles Petrie. What is an Agent? In Jörg P. Müller, Michael J. Wooldridge, and Nicholas R. Jennings, editors, *Proceedings of the ECAI'96 Workshop on Agent Theories, Architectures, and Languages: Intelligent Agents III*, volume 1193 of *LNAI*, pages 41–44, Berlin, August 12–13 1997. Springer.

[Pup90] Frank Puppe. *Problemlösungsmethoden in Expertensystemen*. Studienreihe Informatik. Springer-Verlag, Berlin, 1990.

[Pup91] Frank Puppe. *Einführung in Expertensysteme*. Springer-Verlag, Berlin, 2 edition, 1991.

[Rie94] Doug Riecken. Intelligent agents. *Communications of the ACM*, 37(7):18–21, July 1994.

[Sac77] E.D. Sacerdoti. *A Structure for Plans and Behavior*. Elsevier/North Holland, 1977.

[Sch98] Michael Schroeder. *Autonomous, Model-Based Diagnosis Agents*. Kluwer Academic Publisher, Boston/Dordrecht/London, April 1998.

[Str91] Peter Struß. *Wissensrepräsentation*. Oldenbourg-Verlag, München, Wien, 1991.

[Tat77] Austin Tate. Generating Project Networks. In *Proceedings of the fifth International Joint Conference on Artificial Intelligence, IJCAI-77*, pages 888–893, San Mateo, CA, 1977. Morgan Kaufmann Publishers.

[TW92] Jan Treur and Thomas Wetter. *Formal Specification Of Complex Reasoning Systems*. Ellis Horwood Ltd., London, 1992.

[Ver83] S.A. Vere. Planning in Time: Windows and Durations for Activities and Goals. *IEEE Trans, Pattern Analysis and Machine Intelligence*, 5(3):246–267, 1983.

[WJ95] Michael J. Wooldridge and Nick Jennings. Agent theories, architectures, and languages: a survey. In Michael J. Wooldridge and Nick Jennings, editors, *Intelligent agents: Proceedings of the ECAI-94 Workshop on Agent Theories, Architectures, and Languages*, volume 890 of *Lecture Notes in Artificial Intelligence and Lecture Notes in Computer Science*, pages 1–39, New York, NY, USA, 1995. Springer-Verlag.

A Details zum Reaktormodell 1

Temperatur	Druck	gelber Alarm	roter Alarm
low	low	off	off
low	medium	off	off
low	high	off	off
medium	low	off	off
medium	medium	off	off
medium	high	on	off
high	low	off	off
high	medium	on	off
high	high	off	on

Tabelle 2: Grenzwerte für Alarme

Heizung	Temperaturänderung
dec	-1
no change	0
inc	+1

Tabelle 3: Änderung der Temperatur

In Mass	Out Mass	Druckänderung
dec	dec	0
dec	no change	-1
dec	inc	-2
no change	dec	+1
no change	no change	0
no change	inc	-1
inc	dec	+2
inc	no change	+1
inc	inc	0

Tabelle 4: Änderung des Drucks

B Details zum Reaktormodell 2

Temperatur	Druck	gelber Alarm	roter Alarm
low	low	off	off
low	medium	off	off
low	high	off	off
medium	low	off	off
medium	medium	off	off
medium	high	on	off
high	low	off	off
high	medium	on	off
high	high	off	on

Tabelle 5: Grenzwerte für Alarme

Heizung	Temperaturänderung
dec	-1
no change	0
inc	+1

Tabelle 6: Änderung der Teperatur

In_Mass	Out_Mass	Heizung	Druckänderung
dec	dec	dec	0
dec	dec	no change	0
dec	dec	inc	0
dec	no change	dec	-1
dec	no change	no change	-1
dec	no change	inc	0
dec	inc	dec	-2
dec	inc	no change	-2
dec	inc	inc	-1
no change	dec	dec	0
no change	dec	no change	+1
no change	dec	inc	+1
no change	no change	dec	0
no change	no change	no change	0
no change	no change	inc	0
no change	inc	dec	-1
no change	inc	no change	-1
no change	inc	inc	0
inc	dec	dec	+1
inc	dec	no change	+2
inc	dec	inc	+2
inc	no change	dec	0
inc	no change	no change	+1
inc	no change	inc	+1
inc	inc	dec	0
inc	inc	no change	0
inc	inc	inc	0

Tabelle 7: Änderung des Drucks

C Details zum Reaktormodell 3

Temperatur	Druck	gelber Alarm	roter Alarm
low	low	off	off
low	medium	off	off
low	high	off	off
medium	low	off	off
medium	medium	off	off
medium	high	on	off
high	low	off	off
high	medium	on	off
high	high	off	on

Tabelle 8: Grenzwerte für Alarme

In_Mass	Out_Mass	Heizung	Druckänderung	Temperaturänderung
dec	dec	dec	0	-1
dec	dec	no change	0	0
dec	dec	inc	0	+1
dec	no change	dec	-1	-1
dec	no change	no change	-1	0
dec	no change	inc	0	+1
dec	inc	dec	-1	-2
dec	inc	no change	-1	-1
dec	inc	inc	-1	0
no change	dec	dec	0	-1
no change	dec	no change	+1	0
no change	dec	inc	+1	+1
no change	no change	dec	0	-1
no change	no change	no change	0	0
no change	no change	inc	0	+1
no change	inc	dec	-1	-1
no change	inc	no change	-1	0
no change	inc	inc	0	+1
inc	dec	dec	+1	0
inc	dec	no change	+1	+1
inc	dec	inc	+1	+2
inc	no change	dec	0	-1
inc	no change	no change	+1	0
inc	no change	inc	+1	+1
inc	inc	dec	0	-1
inc	inc	no change	0	0
inc	inc	inc	0	+1

Tabelle 9: Änderung von Druck und Teperatur