Die Extraktion von Restriktionen aus Aufgabenbeschreibungen für die statische, konzeptbasierte Diagnose von Lösungs- bzw. Schaltungsentwürfen (aus der Pneumatik)

Janine Willms, Claus Möbus
Fachbereich Informatik, Abteilung Lehr-und Lernsysteme,
C.v.O. Universität Oldenburg, D-26111 Oldenburg
Email: {Janine.Willms, Claus.Moebus}@informatik.uni-oldenburg.de

Computerbasierte Lehr- und Lernsysteme sollen Wissen effizient vermitteln und den Umgang mit Problemen in Prüfungssituationen oder im Alltag vereinfachen. Damit das Gelernte in aktives, anwendbares Wissen (prozedurales Wissen) beim Teilnehmer umgewandelt wird, reicht die bloße Vermittlung von Fakten und Regeln (deklaratives Wissen) nicht aus. Die Anwendung des Wissens selbst muß in „realen" Problemsituationen trainiert werden, um prozedurales Wissen zu erwerben. Dafür wurden in unserer Arbeitsgruppe sogenannte *Intelligente Problemlöseumgebungen (IPSE)* entwickelt, in denen der Lernende freies, exploratives Problemlösen durch den uneingeschränkten Entwurf von Lösungen trainieren kann. Die Diagnosekomponenten dieser Systeme müssen mächtig genug sein, um sämtliche Entwurfsmöglichkeiten beurteilen zu können. In diesem Beitrag werden verschiedene Ansätze vorgestellt, die *freies Problemlösen* ermöglichen. Wenn der Lernende jedoch die Aufgabenstellung selbst nicht versteht, so können diese Systeme keine Hilfestellung geben. In der IPSE PULSE wurde daher eine konzeptbasierte *Spezifikationsanalyse* integriert, die den Lernenden bereits beim Verstehen der Aufgabenstellung unterstützt und hilft, Anfangsschwierigkeiten zu überwinden. Ein weiterer Vorteil der Spezifikationsanalyse ist die Einsetzbarkeit in einem Dozentenmodus, der die einfache Integration neuer Aufgaben durch einen Dozenten ermöglicht. Eine solche explizite Analyse von Restriktionen aus der Aufgabenbeschreibung wurde bisher in keinem anderen Lehr- und Lernsystem verwirklicht.

1 Einleitung

Der Einsatz von Computern in Aus- und Weiterbildung nimmt kontinuierlich zu. Die eingesetzten Programme unterscheiden sich jedoch teilweise gravierend. Mit dem Begriff *Edutainment-Systeme* werden zum Beispiel Programme bezeichnet, die in ansprechender, multimedialer Weise Fachwissen darstellen. Dabei wird häufig eine Überprüfung und Anwendung des Wissens des Lernenden vernachlässigt. In diesem Beitrag sollen speziell solche Systeme in den Vordergrund treten, die die Anwendung von Wissen ermöglichen und als Übungs- und Trainingssysteme eingesetzt werden können.
Die einfachste Art der Wissensüberprüfung ist der Multiple-Choice-Test. Dieser wird häufig in sogenannten CBT-Systemen (Computer Based Training) eingesetzt. Hierbei kann der Lernende zu einer Frage aus einer gegeben Menge von Antworten auswählen. Die Lösungsvielfalt ist somit bereits durch den Entwickler der

Lernsoftware stark eingeschränkt und ungewöhnliche, kreative Lösungen des Lernenden können durch die Systeme nicht erkannt werden.
Wissensbasierte CBT-Systeme bauen auf einer internen Wissensbasis auf, aus der im Einzelfall Entwürfe geprüft und teilweise sogar Lösungen generiert werden können. Dadurch bieten sie die Möglichkeit, einen sehr viel größeren Lösungsraum abzudecken und ihre eigenen Aktionen sowie die des Lernenden zu beurteilen und zu erklären. Häufig werden solche Systeme auch als *Intelligente Tutorsysteme* (ITS) (Wenger, 1987) oder *Intelligente Problemlöseumgebungen* (IPSE) (Möbus, 1995, 1996) bezeichnet. Je freier der Lernende in solchen Systemen seine eigenen Ideen umsetzen kann, desto schwieriger wird es für den Entwickler, dem System die Fähigkeit zu geben, die Aktionen des Lernenden zu beurteilen. Besonders problematisch ist dabei die Diagnose von Stocksituationen und die Auswahl von Systemhilfen zu deren Überwindung. Als Stocksituationen bezeichnen wir Situationen, in denen der Lernende Hilfe von außen, durch einen Tutor oder das System, benötigt.

2 Stocksituationen und Erklärungsbedarf im Problemlöseprozeß

In Problemlöseumgebungen hat der Lernende die Aufgabe, zu einer gegebenen Spezifikation (Repräsentationsformat R1) eine Lösung (Repräsentationsformat R2) zu erstellen. Diese Aufgabe birgt Fehlerquellen unterschiedlicher Art und Tragweite in sich. So kann man zum einen Fehlerquellen anhand der verschiedenen (nicht zwangsläufig in dieser Reihenfolge bearbeiteten) Subziele des Problemlöseprozesses identifizieren:

1. Lesen der Spezifikation (Analyse der Syntax)
2. Analyse der Spezifikation (Analyse der Semantik)
3. Übertragung der Aufgabenstellung in das Format der Lösung (Übertragung der Semantik)
4. Realisierung der Lösungshypothese (Realisierung der Syntax)
5. Analyse der Lösungshypothese (Eigenanalyse bzw. Selbsterklärung oder Inanspruchnahme von Fremdanalysen (Hypothesentesten))
6. Interpretation des Fehlverhaltens des Entwurfs oder der Fehlermeldung (Selbsterklärung)

Zum anderen kann man unterscheiden, ob es sich bei den Fehlerquellen um Auslöser sofortiger Stocksituationen handelt, wie beispielsweise die Unkenntnis eines in der Spezifikation erwähnten Konzeptes, oder um potentielle Auslöser späterer Stocksituationen (VanLehn, 1990), die erst zum Zeitpunkt des Hypothesentestens offensichtlich werden, wie beispielsweise die Verwechslung von Konzepten. In Tabelle 1 werden Beispiele für mögliche Fehlerquellen noch einmal zusammengefaßt.

Erklärungen dienen der Überwindung von Stocksituationen. Sie müssen daher in allen Phasen und Zielen des Problemlöseprozesses angeboten werden. Viele Systeme unterstützen jedoch lediglich die Umsetzung der Aufgabenstellung, ohne zu berücksichtigen, daß bereits beim Verständnis der Aufgabenstellung Probleme auftreten können. Im folgenden Kapitel werden daher mehrere Systeme vorgestellt und auf Möglichkeiten zur Erklärung der Aufgabenstellung hin untersucht.

Subziele des Problemlöseprozesses	*Potentielle sofortige Stocksituationen*	*Potentielle Auslöser für spätere Stocksituationen*
Lesen der Spezifikation	Unbekannte Basisbauteile von R1	Flüchtigkeitsfehler
Analyse der Spezifikation	Unkenntnis der korrekten Konzeptintension	Verwechslung von Konzepten oder Annahme andersartiger Konzeptintensionen in R1
Übertragung der Aufgabenstellung	Fehlender Plan zur Umsetzung des Konzeptes	Verwechslung von Konzepten oder Annahme andersartiger Konzeptintensionen in R2
Realisierung der Lösungshypothese	Unbekannte Basisbauteile von R2	Fehlerhafte Ausführung des gewählten Plans
Analyse der Lösungshypothese	Mangelnde Analysefähigkeit hinsichtlich des eigenen Entwurfs	Fehlerhafte Eigenanalyse bzw. Selbsterklärung
Interpretation des Fehlverhaltens oder der Fehlermeldung	Mangelnde Fähigkeiten zur Fehlerinterpretation, Reparatur und Ergänzung des Entwurfs	Nicht Erkennen der mangelnden Fähigkeiten zur Fehlerinterpretation, Reparatur und Ergänzung des Entwurfs

Tabelle 1: Ursachen für Stocksituationen

3 Beispiele für ITS und IPSE

In diesem Kapitel werden mehrere etablierte Systeme erläutert und insbesondere auf die Art der Aufgabenstellung und deren interne Repräsentation eingegangen, bevor im nächsten Kapitel am Beispiel der IPSE PULSE (Pneumatic Learning and Simulation Environment) dargestellt wird, wie eine *Spezifikationsanalyse* für Aufgaben aus der Domäne der Pneumatik eingesetzt werden kann und welche Vorteile sich daraus ergeben. Es soll damit verdeutlicht werden, daß die bisherigen Systeme zwar eine Trennung von Aufgabenstellung und Lösungsentwurf vorgeben, intern jedoch bereits eine mehr oder weniger abstrakte Form der Lösung als Aufgabe beinhalten. Entsprechend der Einteilung von (Murray, 1988), kann man Lehrsysteme nach der Art der Entwurfsanalyse unterscheiden. Murray bezieht sich dabei zwar lediglich auf ITS zur Vermittlung von Programmier- und Debuggingwissen, die Taxonomie kann jedoch auch für andere Domänen angewendet werden. Zunächst ist eine Unterscheidung in *statische* und *dynamische* Analysemethoden möglich. Die häufigsten statischen Analysen sind planbasiert und somit direkt von der Anzahl der integrierten Pläne abhängig. Dadurch decken statische Analysen den Lösungsraum oft nur unzureichend ab und können einzelne ungewöhnliche Lösungsvarianten nicht erkennen. Dynamische Analysen basieren auf dem Verhalten eines Entwurfs. Es kann sich dabei um Ein-/Ausgabeanalysen oder eine Art interner Simulation handeln. Dynamische Analysen bieten den Vorteil einer vollständigen Analysemethode. Voraussetzung sind dabei jedoch bereits weit fortgeschrittene und ablauffähige Entwürfe. In frühen Phasen des Entwurfs kann die dynamische Analyse allein daher generell Unterstützung des Lernenden bieten.

Wie die meisten der ersten ITS und Problemlöseumgebungen ist **SPADE** (Miller, 1979) ein System aus der Domäne der Informatik. Es handelt sich um einen planbasierten LOGO-Tutor, der mit Hilfe von Problemlösegrammatiken eine strukturierte Vorgehensweise bei der Programmierung und dem Debuggen von Programmen unterstützt. Es handelt sich also um eine statische Analysemethode, die den Lösungsentwurf des Lernenden parsed und Fehler durch Abweichungen von den im System integrierten Regeln einer kontextfreien Grammatik erkennt.

In **PROUST** (Johnson, 1986) wird der Aspekt der *Intentionsanalyse* von Pascal-Programmen näher untersucht. Abstrakte *Ziele* werden von *Plänen* unterschieden, um die Existenz verschiedener Lösungswege zu verdeutlichen. Die Aufgabenbeschreibung liegt als natürlichsprachlicher Text vor, während die interne Spezifikation einer Aufgabe aus einer Sequenz von Zielen besteht. Die Analyse des Programmcodes des Lernenden geschieht nach dem Schema *„Analyse durch Synthese“*. Es wird versucht, durch heuristische Auswahl und Instantiierung verschiedener Pläne, den gegebenen Code zu rekonstruieren. Um auch Programmierfehler analysieren zu können, wurden in die PROUST bekannte Planmenge auch Fehlerpläne mit aufgenommen. Die Analysefähigkeit von PROUST ist von der Anzahl der integrierten Pläne abhängig. Problematisch sind vor allem Mehrdeutigkeiten, die sich aus verschiedenen Interpretationen des Programmcodes ergeben können. Es ist somit möglich, daß korrekte Programme von der Analysekomponente nicht erkannt und als Zielhierarchie interpretiert werden können. Es handelt sich also um eine unvollständige Analysemethode, die nicht den gesamten Lösungsraum abdeckt. Die Korrektur eines fehlerhaften Programmcodes durch das System ist nicht möglich. Es werden jedoch Hinweise gegeben, in welchen Teilen des Programms sich möglicherweise Fehler befinden.

ABSYNT (Abstract Syntax Trees) ermöglicht im Gegensatz zu PROUST die Vervollständigung eines unvollständigen Entwurfes. ABSYNT ist eine IPSE für die Domäne der funktionalen Programmierung auf Basis der ISP-DL-Theorie ("Impasse-Success-Problem-Solving-Driven-Learning") (Möbus, 1995, 1996), einer kognitiven Theorie des Wissenserwerbs, die den induktiven Wissenserwerb durch Stocksituationen und die deduktive Optimierung bereits vorhandenen Wissens nach der erfolgreichen Lösung eines Problems postuliert. Weiterhin wird davon ausgegangen, daß der Problemlöseprozeß als eine Abfolge verschiedener Phasen angesehen werden kann: der *Abwägephase*, der *Planungsphase*, der *Aktionsphase* und der *Bewertungsphase*. In jeder Phase kann die IPSE Hypothesen des Problemlösers hinsichtlich der Korrektheit des Lösungsentwurfs prüfen.

In ABSYNT werden *Ziel-Mittel-Relationen* (Goal-Means-Relations „GMR“) zur Diagnose und Synthese von Programmbäumen genutzt. Dem Lernenden wird vergleichbar zu PROUST eine textuelle Aufgabe gestellt, die intern als ein bestimmtes Aufgabenziel repräsentiert wird. Die GMR zerlegt das Aufgabenziel in Subziele, die wiederum weiter ausdifferenziert werden können bis auf die Ebene der funktionalen Sprachkonstrukte. Die GMR-Basis beinhaltet ca. 1100 Regeln, durch deren Feinkörnigkeit ein sehr großer Lösungsraum abgedeckt werden kann, und so auch ungewöhnliche Lösungen umfaßt. Im Gegensatz zu PROUST können Lösungen und Teillösungen generiert werden. Damit können Teilentwürfe durch das System ergänzt werden. Jedoch ist in ABSYNT wie auch in PROUST eine vollständige Abdeckung des Lösungsraums nicht möglich. Eine Besonderheit von ABSYNT ist jedoch die explizite Repräsentierbarkeit von Zielen auf Planungsebene. Der Lernende hat die Möglichkeit, *Zielknoten* in seinen Entwurf einzufügen und durch das System testen zu lassen. Eine solche Unterstützung der Planungsphase ist aus anderen Systemen bisher nicht bekannt.

In der IPSE **PETRI-HELP** (Möbus, 1995), die die Modellierung von beliebigen Prozessen mit Hilfe von Petri-Netzen trainiert, ist eine *dynamische* Analyse integriert basierend auf dem Model-Checking-Ansatz (Clarke, Emerson & Sistla, 1986; Damm, Döhmen, Gerstner & Josko, 1990; Josko 1990). Dadurch konnte der Hypothesentest-Ansatz realisiert werden, der es dem Auszubildenden erlaubt, einen beliebigen

Entwurf auf Korrektheit zu testen. Das Model-Checking kann als interne Simulation dynamischer Systeme angesehen werden. Durch die Vollständigkeit dieser Analysemethode, die den gesamten Lösungsraum abdeckt, ist ein korrektes Feedback sichergestellt. Die Aufgabenstellung liegt dem Lernenden als Text und als eine Menge temporallogischer Formeln vor. Anhand der Formeln ist PETRI-HELP in der Lage den Entwurf des Lernenden zu überprüfen. Die Generierung der Formeln wurde jedoch per Hand durch die Entwickler von PETRI-HELP vorgenommen und kann kaum automatisiert werden, da dafür semantisches Textverständnis nötig wäre.
Das System **SYPROS** (Herzog, 1996), das Wissen zur Programmierung mit Semaphoren vermittelt, kombiniert sowohl statische als auch dynamische Analysemethoden. Die statische Analyse arbeitet vergleichbar zu PROUST mit Planschablonen und Fehlerplänen. Die dynamische Analyse untersucht das Ein-/ Ausgabeverhalten des Programmes. Die Aufgabenstellung liegt jedoch bereits als Programmschablone vor, in die lediglich verschiedene Semaphore eingesetzt werden müssen. Die Lösungsvielfalt ist daher bereits durch die Aufgabenstellung eingeschränkt.
Es wird also deutlich, daß in allen der bisher vorgestellten Systeme die Aufgabe selbst nicht explizit repräsentiert wurde. Daher kann eine Erklärung der Aufgabenstellung und Unterstützung in der Planungsphase nur unzureichend geboten werden. Ursache dieses Mangels ist eine zumeist informelle Beschreibung der Aufgabenstellung, anhand der Entwürfe nicht automatisiert überprüft werden können. Dies führte dazu, daß entweder eine formale Aufgabeschreibung zusätzlich eingeführt wurde (siehe PETRI-HELP) oder direkt (teilweise in abstrakter Form) die Lösung der Aufgabe als zu erreichendes Ziel repräsentiert wurde (SPADE, PROUST, ABSYNT, SYPROS).
In dem nächsten Kapitel wird das System PULSE vorgestellt, das ebenfalls statische und dynamische Analysemethoden vereint. Neu ist jedoch die explizite Anwendung einer Aufgabenanalyse, die den Lernenden beim Verstehen der Aufgabenstellung unterstützt.

4 PULSE

PULSE wurde auf Basis von PETRI-HELP in Zusammenarbeit mit der DIHT-Gesellschaft für Berufliche Bildung, Organisation zur Förderung der IHK-Weiterbildung mbH (Deutscher Industrie- und Handelstag in Bonn) sowie den regionalen Industrie- und Handelskammern entwickelt. Die Entwicklung wurde gefördert von der Otto Wolff von Amerongen Stiftung, der Stiftung Volkswagenwerk und dem Institut OFFIS. Das Programm soll in der Ausbildung zum Industriemeister "Metall" in einer Übungsphase des vorher durch Dozenten vermittelten Wissens eingesetzt werden. Als Spezifikation einer Aufgabe dient ein Funktionsdiagramms nach der VDI-Richtlinie 3260 und eine textuelle Beschreibung der zu realisierenden Steuerung (Abb. 1, rechts). Das Funktionsdiagramm beschreibt graphisch den dynamischen Aspekt einer pneumatischen Steuerung auf einem zeit- und zustandsdiskreten Abstraktionsgrad. Daraus soll der Lernende interaktiv einen pneumatischen Schaltplan entwickeln, der den Anforderungen der Aufgabenstellung entspricht. Dazu kann er beliebige pneumatische Bauteile nach DIN 1219 in dem graphischen Editor (Abb. 1, links) plazieren und durch Leitungen verbinden. Den Auszubildenden steht eine Sequenz von 30 Aufgaben der Prüfungs-, Aufgaben- und Lernmittelstelle (PAL) zur Verfügung. Diese wurden früher als Papier-und-Bleistift-

Aufgaben vorgelegt. Nach Lösung einer Aufgabe konnte der Dozent den Entwurf des Auszubildenden diskutieren. Es ist nicht unplausibel anzunehmen, daß vom Dozenten unbewußt alle Lernerentwürfe als Abweichungen von seinen Musterlösungen angesehen wurden. Werden Abweichungen vom Dozenten als unangenehm erlebt, so wäre eine weniger wertende Instanz (zum Beispiel PULSE) wünschenswert.

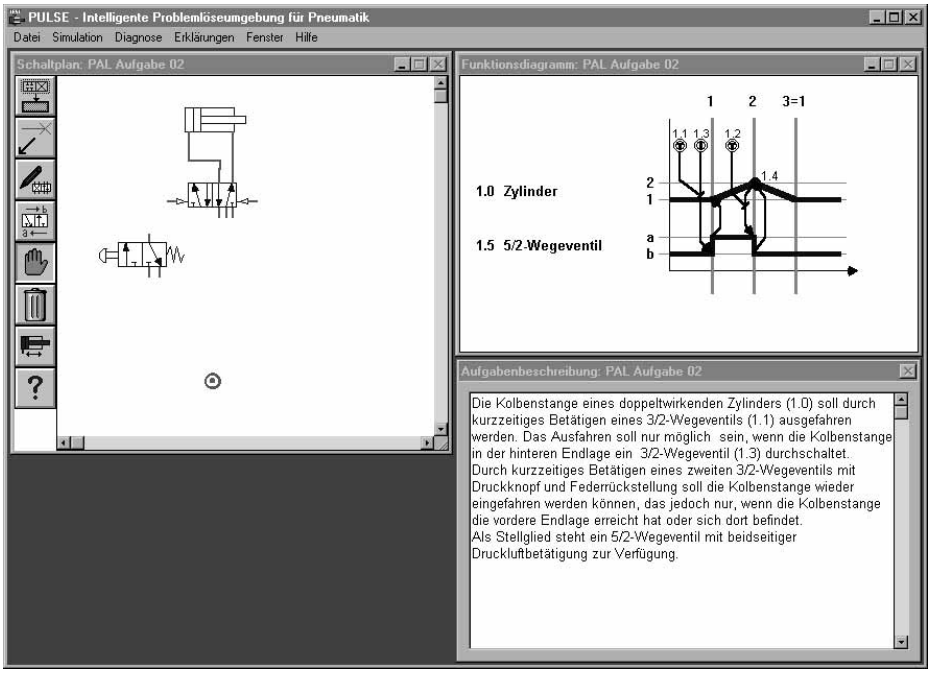

Abb. 1: Aufgabe PAL 02

Die Ziele bei der Entwicklung von PULSE gliederten sich in

- Wissenserwerb durch Problemlösen (d.h. Entwurf von Schaltungen)
- durch Eigenaktivität und Bearbeitung konstruktiver Aufgaben die Motivation des Auszubildenden aufrecht zu erhalten
- Kreativität und aktives Lernen zu unterstützen, indem beliebige, auch ungewöhnliche Entwürfe durch PULSE überprüft und kommentiert werden können.
- Hilfen und Erklärungen zu jedem Zeitpunkt und situationsbezogen anzubieten
- durch entdeckendes Problemlösen und Formulierung von Hypothesen qualitativ hochwertiges Wissen zu vermitteln
- den Dozenten bei der Betreuung der Auszubildenden zu entlasten, so daß er sich mehr um die Lernenden mit fundamentalen Verständnisschwierigkeiten mit der Domäne kümmern kann.

Kernpunkt der Entwicklung von PULSE ist der Hypothesentest-Ansatz. Dieser wird in PULSE ermöglicht durch die Kombination einer statischen, konzeptbasierten Analyse und einer dynamischen Analyse, dem Model-Checking. Das Model-Checking basiert auf einer internen Simulation der Schaltung. Somit kann PULSE für jeden Entwurf des Lernenden eine korrekte Rückmeldung liefern, ist jedoch auf einen

simulationsfähigen Entwurf angewiesen. Die statische Analyse dagegen ist in der Lage, bereits in frühen Entwurfsphasen mit Erklärungen und Ergänzungsvorschlägen den Lernenden zu unterstützen, also auch dann, wenn noch kein simulationsfähiger Entwurf existiert. Beispiele und Erklärungen zur Funktionsweise des Systems PULSE werden in (Willms, Göhler & Möbus, 1997a) und detaillierter in (Willms, Göhler & Möbus, 1997b) beschrieben.
Die statische Analyse wird von PULSE jedoch nicht erst bei der Beurteilung des Entwurfes genutzt, sondern wird bereits bei der Analyse der Aufgabenstellung selbst eingesetzt, wie das nächste Kapitel genauer verdeutlicht.

4.1 Die konzeptbasierte Spezifikationsanalyse in PULSE

Die Analyse der Aufgabenstellung in PULSE wurde entwickelt, um Verständnisschwierigkeiten mit der Aufgabenstellung entgegenzuwirken. Diese können speziell in der Domäne der Pneumatik leicht auftreten, da bereits die Aufgabenstellung domänenspezifisches Wissen erfordert. Die Daten der Spezifikation werden als PROLOG-Fakten repräsentiert. Aus den vorliegenden Fakten werden mittels domänenspezifischer Heuristiken und Regeln deduktiv Constraints extrahiert, die den Lösungsraum der aktuellen Aufgabe näher bestimmen. Im übertragenden Sinne muß sich PULSE also die Bedeutung der Aufgabenstellung aus der gegebenen Spezifikation erst erarbeiten und kann dann eine Lösung generieren. Insofern leistet es dieselben kognitiven Aufgaben, die auch ein Lernender anwenden muß, und kann diese deshalb auch dem Lernenden erklären. Eine solche Aufgabenanalyse ist bisher in keinem System realisiert worden, da in anderen Systemen aufgrund der textuellen Aufgabenbeschreibung dafür eine semantische, also inhaltliche Analyse des Textes notwendig wäre.
In PULSE liegt die Aufgabenstellung in zwei Arten vor, die beide dem Lernenden zur Verfügung stehen. Das Funktionsdiagramm beinhaltet eingeschränkte Informationen über zu realisierende Bauteile und Funktionen. Es werden daher *Objektconstraints*, welche Bedingungen zur Realisation von bestimmten Bauteilen beinhalten, und *Funktionsconstraints*, die die Existenz bestimmter Funktionseinheiten fordern, unterschieden. Die Spezifikationsanalyse sammelt zunächst mittels domänenspezifischer Heuristiken, die in das System integriert wurden, sämtliche aus dem Funktionsdiagramm erkennbaren Objektconstraints: beispielsweise den Typ eines Bauteils (Zylinder, Wegeventil, etc.), die Betätigungsart (Manuell, druckluftbetätigt, etc.) oder Initialzustand. Einige dieser Informationen können direkt dem Funktionsdiagramm entnommen werden, andere müssen erst durch Interpretation erschlossen werden. In PAL 02 kann beispielsweise direkt dem Funktionsdiagramm entnommen werden, daß Objekt 1.5 ein 5/2-Wegeventil sein muß. Daß das Ventil auch durch Druckluft betätigt werden muß, kann indirekt aus der Tatsache geschlossen werden, daß das Ventil 1.5 durch die UND-Verknüpfung der Ventile 1.1 und 1.3 (Abhängigkeitspfeil) umgeschaltet wird. Unterstützt wird diese Vermutung durch die textuelle Beschreibung, in der „ein 5/2-Wegeventil mit beidseitiger Druckluftbetätigung“ erwähnt wird. Die Zuordnung, daß damit das im Funktionsdiagramm auftretende Ventil 1.5 gemeint ist, erfolgt jedoch erst in einer späteren Phase der Spezifikationsanalyse.
Einige Daten des Funktionsdiagramms sind mehrdeutig. So kann beispielsweise das Schaltersymbol für Objekt 1.3 einen manuell betätigten Schalter darstellen, in der

Initialstellung wird auf diese Weise jedoch auch ein rollenbetätigtes Ventil repräsentiert, das durch den Zylinder aktiviert wird. In Aufgabe PAL 02 liegt letzterer Spezialfall vor. Nur die textuelle Beschreibung kann diesen Konflikt auflösen. Da auch in unserer Arbeit keine automatische semantische Analyse des Textes möglich war, wurden die Constraints der textuellen Beschreibung per Hand kodiert und verfügbar gemacht. Aus der Abstimmung der Constraints aus Funktionsdiagramm und textueller Beschreibung läßt sich der Konflikt lösen: Ventil 1.3 wird im Text explizit als ein Ventil genannt, das durch die Kolbenstange des Zylinders betätigt wird, also ein rollenbetätigtes Ventil ist.
Nach der Analyse und dem Abgleich der Objektconstraints aus Funktionsdiagramm und textueller Beschreibung, werden Funktionskonzepte gesucht. In dem Funktionsdiagramm von PAL 02 werden dabei die Funktionen UND(1.1,1.3), UND(1.2,1.4) und mehrere Aktivierungen und Deaktivierungen von Ventilen entdeckt. Diese können zu höheren Funktionskonzepten verknüpft werden, zum Beispiel zum Vor- und Rücklauf, der Steuerkette und der indirekten Steuerung. Nach dem Bottom-Up-Ansatz der Objektconstraint-Analyse wird nun versucht, Funktionsconstraints der Aufgabenbeschreibung aus Funktionsdiagramm und textueller Beschreibung einander zuzuordnen und so in einem Top-Down-Ansatz noch immer unklare Objektbezeichnungen aufzulösen. Es ergibt sich aus der Analyse der Funktionen im Funktionsdiagramm, daß das Ventil 1.5 als Stellglied bezeichnet werden kann. Jetzt kann die Verbindung zu dem unbenannten Ventil in der textuellen Beschreibung geschaffen werden und Ventil 1.5 bekommt die zu realisierende Objekteigenschaft „Beidseitige Druckluftbetätigung“ zugeschrieben.
Zur statischen Analyse von Entwürfen werden die so gewonnenen Objekt- und Funktionsconstraints genutzt. Auch die Vervollständigung und Korrektur von Schaltplänen ist dadurch möglich. Zur Erklärung der Aufgabenstellung wird Rückbezug auf die genutzten Heuristiken der Spezifikationsanalyse und deren Ausgangspunkt im Funktionsdiagramm oder der textuellen Beschreibung genommen. Doch nicht nur die Erklärungsfähigkeit ist ein Vorteil der Spezifikationsanalyse. Im nächsten Abschnitt wird darauf eingegangen, wie aufgrund der Spezifikationsanalyse ein Dozentenmodus realisiert werden kann, der die Integration beliebiger neuer Aufgaben in das System erlaubt, ohne programmieren zu müssen.

4.2 Der Dozentenmodus

Da Pulse in der Lage ist, die in der Pneumatik üblichen Repräsentationsformen zu analysieren, konnte der Prototyp eines Dozentenmodus erstellt werden, der die interaktive Eingabe beliebiger neuer Aufgaben in das System erlaubt und dennoch in der Lage ist, mit Hilfe der dynamischen Analyse die Korrektheit von Entwürfen bezüglich dieser neuen Aufgaben zu prüfen. Dabei wird jedoch nur das Verhalten der Schaltung mit dem Funktionsdiagramm verglichen, die textuelle Beschreibung wird zunächst vernachlässigt. Im Funktionsdiagramm können beliebig viele Bauteile als Linie oder als Schaltsymbol definiert und durch Abhängigkeitslinien verknüpft werden.
Ein begleitender Text kann ebenfalls durch den Dozenten eingegeben werden. Dieser Text kann entweder ohne Einfluß auf die Lösung der neuen Aufgabe bleiben oder die semantischen Inhalte müßten zur Nutzung der statischen Analyse ebenfalls explizit definiert werden, was die Definition neuer Aufgaben durch einen ungeübten Dozenten

erschwert. Prototypisch wurde eine Umgebung geschaffen, die die Zuordnung von Eigenschaften zu Objekten und einer entsprechenden Textstelle ermöglicht. Die Definition neuer Aufgaben muß jedoch sehr diszipliniert vorgenommen werden, da inkonsistente Aufgaben bisher nicht abgefangen werden. Auch die unvollständige oder ungenaue Definition von Aufgaben ist leider möglich. Während unvollständige Definitionen lediglich zu einem größeren Lösungsraum führen, können ungenaue Definitionen zu Mehrdeutigkeiten führen, die durch die statische Analyse nicht aufgelöst werden können, beispielsweise, die Aussage, daß ein 4/2-Wegeventil verwendet werden muß, ohne Angabe des konkreten Bauteilnamens und ohne Möglichkeit eindeutig zuzuordnen.
Die Funktionsweise des Dozentenmodus wurde anhand der ersten 10 PAL-Aufgaben getestet. Diese können beliebig in den Dozentenmodus übertragen, verändert und wieder in den Aufgabenmodus zurück übertragen werden. Dadurch kann die Funktionsweise des Dozentenmodus an verschiedenen Beispielen erläutert werden. Für den aktiven Einsatz wäre jedoch eine Schulung der Dozenten anzuraten, um eventuell auftretende Probleme erkennen und richtig diagnostizieren zu können. Dieses könnte durch Integration verschiedener Konsistenztest (beispielsweise der konsistenten Definition von Bauteileigenschaften in Funktionsdiagramm und Text, der korrekten Zuordnung von Symbolen zu Ventilen oder der korrekten Anzahl von möglichen Zuständen zu Objekten) noch unterstützt werden.

5 Zusammenfassung

Die Analysefähigkeit der Aufgabenstellung ist eine Eigenschaft des Systems PULSE, die in anderen Lehr-/Lernsystemen bisher nicht vorhanden ist. Sie basiert auf einer konzeptuellen Analyse des Funktionsdiagramms mittels domänenspezifischer Heuristiken und der Interpretation kodierter Informationen, die im Text der Aufgabenstellung vorhanden sind. Die Informationen werden einander zugeordnet, verknüpft und zu einer Hierarchie von Konzepten aufgebaut. Diese bilden die Basis für die spätere konzeptbasierte Analyse des Entwurfs. Durch die explizite Spezifikationsanalyse werden zwei Vorteile erreicht: die Aufgabenstellung selbst wird erklärbar und die Analyse kann auf beliebige andere Aufgaben angewendet werden, so daß die Implementierung eines Dozentenmodus möglich wurde.

6 Literatur

Clarke, E.M., Emerson, F.A. & Sistla, A.P. (1986). Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specifications. ACM Transactions on Programming Languages and Systems, Vol. 8, No. 2, 244 - 263

Damm, W., Döhmen, G., Gerstner, V., Josko, B. (1990). Modular Verification of Petri Nets. The Temporal Logic Approach, in: J.W. de Bakker, W.P. de Roever, G. Rozenberg (eds.), Proceedings REX-Workshop on stepwise refinement of distributed systems: models, formalisms, correctness. Berlin: Springer, LNCS 430.

Herzog, C. (1996). Syntaxorientierte vs. Ablauforientierte Diagnose in intelligenten Programmierumgebungen als Beispiel für den Einsatz konkurrierender Problemlöser, in: M. Thielscher, S.-E. Bornscheuer (eds.) Fortschritte der Künstlichen Intelligenz (KI 96), Dresden: Dresden University Press.

Johnson, W.L. (1986). Intention-based Diagnosis of Novice Programming Errors. Los Altos: Morgan Kaufmann.

Josko, B. (1990). Veryfying the correctness of AADL modules using model checking, in: J.W. de Bakker, W.P. de Roever, G. Rozenberg (eds.), Proceedings REX-Workshop on stepwise refinement of distributed systems: models, formalisms, correctness. Berlin: Springer, LNCS 430, 386-400.

Miller, M.L. (1979). A structured planning and debugging environment for elementory programming

Mizoguchi, R., Ikeda, M., Sinitsa, K. (1997). Roles of Shared Ontology in AI-ED Research, in: B. du Boulay, R. Mizoguchi (eds), Artificial Intelligence in Education, IOS Press, 537-544.

Möbus, C., Thole, H.J., Schröder, O. (1993). Diagnosis of Intentions and Interactive Support of Planning in a Functional, Visual Programming Language, in D.M. Towne, T. de Jong, H. Spada (eds), Simulation-Based Experiential Learning, Berlin: Springer, 61-76.

Möbus, C. (1995). Towards an Epistemology of Intelligent Problem Solving Environments: The Hypothesis Testing Approach I, in: J. Greer (ed.), Artificial Intelligence in Education, Proceedings of AI-ED 95, Washington, D.C., August 16-19, 1995, Charlottesville: AACE, 138-145

Möbus, C., (1996). Towards an Epistemology on Intelligent Problem Solving Environments: The Hypothesis Testing Approach II, in: Proceedings of EuroAIED 96, Lisbon, Portugal, Sept. 30 - Oct. 2,

Murray, W.R. (1988). Automatic Program Debugging for Intelligent Tutoring Systems. Morgan Kaufmann Publishers, Inc. San Mateo, California.

VanLehn, K., (1990). Mind Bugs: The origins of procedural misconceptions. MIT Press, Cambridge.

Wenger, E., (1987). Artificial Intelligence and Tutoring Systems. Morgan Kaufmann Publishers, Inc., Los Altos, California.

Willms, J., Göhler, H., Möbus, C., (1997a). Testing Hypotheses in an Engineering Domain: Combining Static and Dynamic Analysis of Pneumatic Circuits, in: B. Boulay, R. Mizoguchi (eds.): Artificial Intelligence in Education, Amsterdam: IOS-Press, 680-682.

Willms, J., Göhler, H., Möbus, C., (1997b). Die Integration von dynamischen und statischen Analysemethoden in einer intelligenten Lern- und Problemlöseumgebung, in: C. Herzog, Beiträge zum 8. Arbeitstreffen der GI-Fachgruppe 1.1.5/7.0.1 „Intelligente Lehr- und Lernsysteme“, Duisburg, 18.-19. September 1997, „Blaue Berichte" der TU München, 1997.