In: Jähnichen, S. (ed.): Innovative Softwaretechnologien. Velbert Online Verlag, S. 401- 426

Kooperative Hilfesysteme für Anwendungssoftware Entwicklungsstand und Forschungstrends

Wolfgang Wahlster, Mathias Bauer, Susanne Biundo,
Dietmar Dengler, Jana Köhler, Gabriele Paul, Markus A. Thies

Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI)
Stuhlsatzenhausweg 3
66123 Saarbrücken
Tel: 0681 302 5252, Fax: 0681 302 5341

1 Intelligente Hilfesysteme

Firmen beklagen sich in den letzten Jahren immer häufiger darüber, daß bis zu 6 Stunden pro Woche an wertvoller Arbeitszeit ihrer Angestellten durch Bedienprobleme bei Computerprogrammen verloren gehen. Allein in den USA sollen laut Umfragen ca. 100 Milliarden $ Verluste durch Fehlbedienungen, Herumprobieren, Suchen in Handbüchern und andere Formen des unproduktiven Arbeitens mit Anwendungssoftware entstehen. *Intelligente Hilfesysteme* für Softwareprodukte sollen in dieser Situation Abhilfe schaffen: Sie sollen den Anwender bei der Erreichung seines Arbeitsziels so unterstützten, daß er die Software möglichst effizient und ohne Fehlbedienungen einsetzt.
Die Akzeptanz komplexer Software hängt beim Anwender immer stärker von der Qualität der Hilfen ab, die er bei der Einarbeitung in ein neues System und später bei Bedienproblemen angeboten bekommt. Man kann daher feststellen, daß die sog. *Helpware* zur dritten Dimension der Informationstechnik, neben der Software und Hardware, geworden ist. Bei immer stärker konvergierenden Software- und Hardware-Konzepten kann die Qualität der angebotenen Helpware daher zum ausschlaggebenden Unterscheidungsmerkmal eines Produktes werden, was zum Beispiel durch den Erfolg der Firma Dell in den USA durch ihr umfassendes Hilfeangebot für PC-Käufer bestätigt wird.
Bei der Entwicklung intelligenter Hilfesysteme geht man davon aus, daß ein Benutzer zur Erreichung seines Arbeitsziels einen bestimmten Plan verfolgt, der sich in einer Folge von Bedienaktionen konkretisiert. Ein solcher Plan kann fehlerhaft, suboptimal, oder unvollständig sein, weil der Bediener die Funktionaliät des Softwaresystems nur zum Teil beherrscht. Der Begriff *planbasierte Hilfesysteme* weist auf die zentrale Rolle von Bedienplänen für die Unterstützung des Benutzers hin: einerseits muß das System zunächst den intendierten Plan des Benutzers möglichst frühzeitig erkennen, um rechtzeitige und effiziente Hilfestellungen anzubieten. Andrerseits muß das Hilfesystem auch in der Lage sein, selbst Pläne zu generieren, die das angenommene Benutzerziel effizient erreichen, z.B. wenn der Benutzer selbst nicht weiterkommt oder einen offensichtlich fehlerhaften oder im gegebenen Kontext suboptimalen Plan verfolgt. Das bedeutet, daß planbasierte Hilfesysteme über Module zur Erkennung und zur Generierung von Plänen verfügen müssen.
Der Umgang mit Plänen ist ein wissensintensiver Prozess, der z.B. auf die Semantik der elementaren Bedienaktionen Bezug nehmen muß. Unsere Arbeitshypothese ist, daß solches Planungswissen am besten so repräsentiert wird, daß es sowohl für die Plangenerierung als auch für die Planerkennung nutzbar ist.
Wir stellen im folgenden einige Ergebnisse von zwei DFKI-Projekten zum Themenbereich *Planbasierte Hilfesysteme* vor, die beide in Anwendungsdomänen von kommerziell eingesetzter Software erfolgreich erprobt wurden.
Das Projekt PHI (Plan-basierte Hilfesysteme) ist ein vom BMFT gefördertes Projekt, während PLUS (Plan-Based User Support) ein von IBM finanziertes Tandemprojekt zu PHI ist. In beiden Projekten wurden vollständig implementierte Prototypen entwickelt. Beide Systeme wenden explizit repräsentiertes Planungswissen bidirektional für die Erkennung und die Generierung von Bedienplänen an. Während von PHI Hilfen bei der Verwendung der textuelle Kommandosschnittstelle des UNIX Mail-Systems generiert werden, arbeitet PLUS im Kontext einer direkt-manipulativen Schnittstelle.

2 PHI: Ein logik-basiertes Hilfesystem

PHI ist das erste intelligente Hilfesystem, das vollständig logik-basiert ist. Das bedeutet, daß eine spezielle Logik als Wissensrepräsentationssprache verwendet wird und alle benötigten Inferenzen von einem logischen Beweissystem durchgeführt werden. Die Planerkennung wird in diesem Rahmen logisch als *Abduktion* betrachtet, während die Plangenerierung als *Deduktion* realisiert ist.
Die Planerkennungskomponente analysiert das beobachtete Benutzerverhalten und identifiziert Ziele und Pläne eines Benutzers. Die Plangenerierungskomponente erzeugt Pläne, mit denen ein Benutzer gewünschte Ziele erreichen kann und bildet damit die Basis für flexible und individuelle Hilfeleistung.

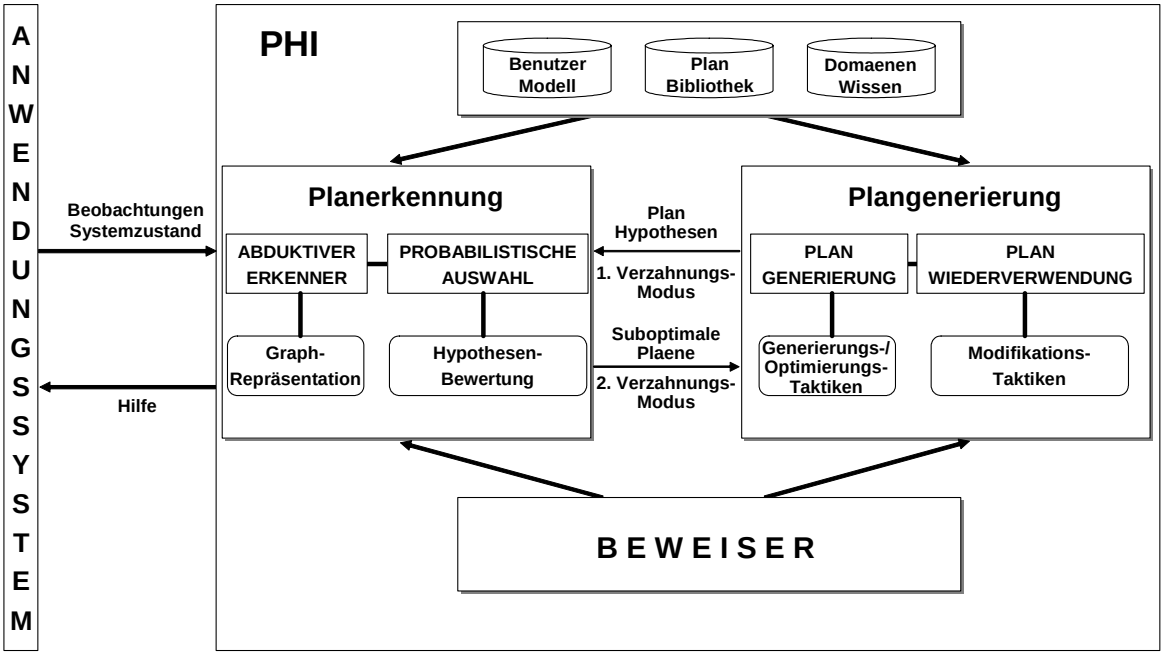

Abbildung 1: PHI - Ein logikbasierter Kern für intelligente Hilfesysteme

Ziel des PHI-Projektes ist die Entwicklung eines logikbasierten Kernsystems, das als anwendungsunabhängige Komponente für intelligente Hilfesysteme in verschiedenen Anwendungen eingesetzt werden kann. Dieses Kernsystem PHI (vgl. Abb. 1) besitzt drei Hauptmerkmale:

- Erstmalig werden hier *Plangenerierung* und *Planerkennung* vollständig integriert.

- Beiden liegt ein *gemeinsamer logischer Formalismus* zugrunde.

- Das System realisiert zwei Arten wechelseitiger Kooperation zwischen den Komponenten, sogenannte *Verzahnungsmodi.*
  Der erste Verzahnungsmodus realisiert Planerkennung auf der Basis von Planhypothesen, die die Generierungskomponente zur Verfügung stellt. Im zweiten Verzahnungsmodus werden dem Benutzer optimale Pläne präsentiert, falls das System suboptimales Verhalten erkannt hat.

2.1 Deduktives Planen

Im PHI-Projekt wurde ein deduktiver Planungsansatz gewählt (vgl. [BDK92, BBD+93]), d.h., Planen geschieht durch Beweisen von formalen Planspezifikationsformeln in einem logischen Kalkül. Es wird das Paradigma verfolgt, *Pläne als Programme* anzusehen (vgl. [Bib86, MW87]).

2.1.1 Der Planungsmechanismus

Als Grundlage für den deduktiven Planungsansatz wurde eine spezielle Logik (LLP) entwickelt (vgl. [BDK92]), die den Anforderungen der Anwendungsdomäne - Hilfesysteme für kommandosprachlich-gesteuerte Anwendungssoftware - gerecht wird. LLP ist eine intervallbasierte modale Temporallogik erster Ordnung und enthält die folgenden Modaloperatoren zum Ausdruck zeitlicher Beziehungen (die Tabelle enthält die intuitive Bedeutung der Operatoren):

- ◊ (sometimes): $\Diamond\phi$ sagt, die Formel ϕ gilt ab irgendeinem Zustand in der Zukunft (eingeschlossen dem aktuellen Zustand);
- ○ (next): $\bigcirc\phi$ bedeutet, die Formel ϕ gilt ab dem nächsten Zustand;
- □ (always): $\Box\phi$ bedeutet, daß ϕ ab jedem erreichbaren Zustand in der Zukunft (eingeschlossen dem aktuellen Zustand) gilt;
- ; (chop): $\phi;\psi$ bedeutet, daß man einen Zeitraum so aufteilen kann, daß zuerst ϕ gilt und anschließend daran die Formel ψ.

Mit Hilfe von LLP kann man nun sowohl die Spezifikationen für Pläne, die man generieren möchte, beschreiben als auch die Pläne selbst; beides sind ausgezeichnete, logische Formeln in LLP.
Pläne können neben Kontrollstrukturen, wie Hintereinanderausführung (;), Fallunterscheidung (if ... then ... else) und Wiederholung (while), wobei die beiden letzten in LLP definiert wurden, auch verschiedene Formen von Abstraktionen enthalten:

- Variablen: man abstrahiert von konkreten Objekten;
- Nichtdeterminismus: man abstrahiert von der Auswahl von Teilplänen, die den gleichen Effekt erreichen;
- Nichtlinearität: man abstrahiert von der Ausführungsreihenfolge von unabhängigen Teilplänen;
- Temporale Abstraktion: man abstrahiert vom konkreten Ausführungszeitpunkt eines Teilplanes.

In dem logischen Rahmen von LLP werden nun die Kommandos der Anwendungsdomäne, ähnlich wie Zuweisungen in Programmlogiken, als elementare Operationen axiomatisiert. Damit erhält man ein Axiomenschema für jede Operation, das sowohl ihre Vorbedingungen und Effekte, als auch ihre Invarianten beschreibt.

Eine typische Instanz eines solchen Schemas ist etwa für das Kommando *delete* in der Domäne "Bearbeitung elektronischer Post" gegeben:

$$\forall x \quad open\_flag(mbox) = 1 \wedge delete\_flag(msg(x, mbox)) = 0 \wedge EX(delete(x, mbox)) \rightarrow \\ \bigcirc delete\_flag(msg(x, mbox)) = 1$$

Das Axiom beschreibt, daß eine nicht gelöschte Nachricht mit Index *x* in dem Briefkasten *mbox,* der geöffnet ist, durch Ausführung des Kommandos *delete* im nächsten Zustand gelöscht ist.

Eine Generierungsskizze Der Planungsprozess startet mit einer Planspezifikations-formel, z.B. von der Form

$$[precondition \wedge \mathbf{Plan}] \rightarrow goals.$$

Diese Formel besagt, daß, wenn unter den Vorbedingungen *precondition* ein entsprechender Plan ausgeführt wird, man damit die Ziele *goals* erreicht. **Plan** ist darin eine Metavariable für einen Plan. Diese Metavariable wird bei einem erfolgreichen konstruktiven Beweis der Spezifikationsformel durch eine LLP-Planformel instanziiert, die die Spezifikation garantiert erfüllt.
precondition ist eine prädikatenlogische Formel, die eine Startsituation mehr oder weniger genau beschreiben kann, *goals* hingegen ist eine modallogische Formel, die das zeitliche Auftreten von einzelnen Zielsituationen, die der Plan erreichen muß, beschreibt. Ein Beispiel für eine einfache Spezifikation, wiederum aus der Domäne "Elektronische Post", ist:

$$[open\_flag(mbox) = 1 \wedge read\_flag(msg(x, mbox)) = \mathbf{0} \wedge \mathbf{Plan}] \rightarrow \\ \Diamond[read\_flag(msg(x, mbox)) = 1 \wedge save\_file(msg(x, mbox)) = Datei1 \wedge \\ delete\_flag(msg(x, mbox)) = 1]$$

Sie besagt, daß eine Nachricht, die noch nicht gelesen ist, irgendwann gelesen, abgespeichert und gelöscht sein muß.
Der Beweis einer Spezifikationsformel geschieht grob nach folgendem Schema: die Ziele werden in Einzelziele aufgespalten und gleichzeitig wird eine entsprechende Kontrollstruktur in die Planmetavariable eingeführt; dann sucht man nach Teilplänen, die die Einzelziele erreichen, und instanziiert damit die Planmetavariable vollständig. Zu einem vollständigen Beweis der Spezifikationsformel gehört auch der Beweis von Zusicherungen, die der resultierende Plan erfüllen muß. Dies sind etwa Aussagen der Form, daß ein Teilplan bestimmte Eigenschaften nicht zerstört, weil sie von einem anderen, zeitlich später folgenden Teilplan als Vorbedingung benötigt werden.
Ein Plan, der nun obige Spezifikation erfüllt, ist der folgende:

[if delete_flag(msg(x,mbQx)) = 1 then EX(undelete(x))];
EX(type(x));EX(save(x,Datei1));EX(delete(x))

Da die Vorbedingungen unvollständig spezifiziert waren, ist eine Fallunterscheidung eingeführt worden, die angibt, was zu tun ist, wenn die Nachricht *x* bereits gelöscht ist.
Gemäß dem Ansatz des *Taktischen Theorembeweisens* (vgl. [HRS90, Pau90]) wurde zur Realisierung des deduktiven Planers eine Taktikensprache implementiert, in der die notwendigen Beweis- bzw. Planungsstrategien explizit formuliert werden können. Dies hat den Vorteil, daß ein spezifischer Beweiser für die Logik LLP implementiert werden

kann, der das besondere Vorgehen beim deduktiven Planen unterstützt. Ein positiver Nebeneffekt ist zudem, daß man damit einen Planer erhält, der in seiner Performanz vergleichbar ist mit nicht-deduktiven Planungsansätzen, allerdings mit der Gewissheit, daß ein gefundener Plan garantiert seine Spezifikation erfüllt.

2.1.2 Generierung optimaler Pläne

Betrachtet man den Planer als Komponente in einem intelligenten Hilfesystem, so sind die Pläne, die man von ihm erwartet, sehr unterschiedlich bzgl. ihrer Struktur und ihrem Lösungsweg bezogen auf die zu erreichenden Ziele. Dienen Pläne etwa als Hypothesen für die Beobachtung eines Benutzers, so sollten sie den Eigenheiten und dem Kenntnisstand des Benutzers angepaßt sein. Insbesondere wird ein unerfahrener Benutzer oft suboptimale Pläne ausführen, die dann natürlich auch generierbar sein müssen. Andererseits ist eine wesentliche Aufgabe eines intelligenten Hilfesystems, daß es dem Benutzer des Anwendungssytems aktiv einen Verbesserungsvorschlag für von ihm vollzogenes suboptimales Verhalten anbietet, d.h. der Planer muß in der Lage sein, den Lösungsweg für ein Problem zu optimieren. In welchem Maße dies geschieht, hängt von der gewünschten Hilfestrategie ab. Man kann sich etwa bei der Optimierung nur auf den Wissensstand des Benutzers beziehen, oder aber man erlaubt auch, daß dem Benutzer noch unbekanntes Wissen benutzt werden darf.
Die Realisierung eines Planers, der unterschiedlichste Anforderungen an seine zu produzierenden Ergebnisse erfüllen kann, erfolgt durch eine Verallgemeinerung des oben beschriebenen deduktiven Planungsansatzes. Ausgehend von einer Spezifikation von Qualitätsmerkmalen für den gesuchten Plan wird automatisch eine Kontrollstrategie in Form einer Taktik konfiguriert, die den Beweisprozess - und damit den Planungsprozess - so steuert, daß Pläne der gewünschten Qualität erzeugt werden können. Die Ansteuerung des Beweisprozesses wird damit jeweils veränderten Qualitätsanforderungen angepaßt.
Als Beispiel für eine Optimierungsstrategie des Planers sei eine Strategie zur Behandlung konjunktiver Zielbeschreibungen erwähnt. Es wurde eine Heuristik zur Ordnung von Teilzielen entwickelt, die negative Interaktionen beim Planen vorab zu vermeiden versucht (vgl. [CI89]). Die Heuristik beruht zum einen auf einer statischen Analyse der axiomatisierten Aktionen der Anwendungsdomäne, um Informationen über negative Interaktionen zwischen Aktionen zu gewinnen, und zum anderen auf domänenunabhängigen allgemeinen Ordnungskriterien, z.B. das Gruppieren von Zielen, die sich auf den gleichen Objektkontext beziehen. Man erreicht mit einer solchen Heuristik nicht nur eine Effizienzsteigerung bei der Plansuche, sondern im allgemeinen auch kürzere Pläne.

2.2 Wiederverwendung von Plänen in PHI

Die Generierung von Plänen aus vorhandenen Spezifikationen ist ein aufwendiger und komplizierter Prozeß. Seit einigen Jahren wird deshalb immer wieder von Forschern diskutiert, wie die Effizienz von Planungssystemen gesteigert werden kann. Ein offensichtlicher Ansatzpunkt ist dabei die Wiederverwendung einmal generierter Pläne: Ein generierter Plan stellt Problemlösungswissen dar, das von einem Planungsssystem wiederverwendet werden sollte, wenn es ähnliche Planungsaufgaben zu lösen hat. Anstatt also immer wieder nach Lösungen zu suchen, soll ein Planungssystem in der Lage sein,

sich diese Lösungen zu "merken", um zukünftige Planungsaufgaben effizienter lösen zu können.
Am weitesten entwickelt sind die Forschungen zu dieser Problematik auf dem Gebiet des automatischen Planens, die durch Arbeiten zum sogenannten *Fall-basierten Schließen* und durch Anforderungen aus der *Robotik* motiviert wurden. Eine Vielzahl unterschiedlichster Ansätze wurde bislang entwickelt, die einzelne Teilprobleme, die bei der Wiederverwendung von Plänen auftreten untersuchen und dafür unterschiedliche Lösungstechniken entwickeln (vgl. [Ham90, KH92, HW92]).
Innerhalb des PHI Projektes konnte ein Beitrag zur Lösung wichtiger offener Fragestellungen innerhalb dieses Forschungsgebietes geleistet werden.
Zunächst wurde ein formales Modell entwickelt, das die Wiederverwendung von Plänen als strukturierten Prozeß beschreibt [Koe92]. Dieses Modell gliedert den Wiederverwendungsprozeß in vier Phasen und ermöglicht eine Beschreibung der bei der Wiederverwendung von Plänen zu lösenden Aufgaben unabhängig von einer bestimmten Anwendungsdomäne und einem konkreten deduktiven Planungssystem.

2.2.1 Das 4-Phasen Modell des Wiederverwendungsprozesses

Der Prozeß der Wiederverwendung von Plänen erfolgt in vier Phasen:

- **Phase I: Plan Determination**

  Ausgangspunkt ist die Beschreibung eines aktuellen Planungsproblems in Form einer formalen Planspezifikation, für die ein Plan als Lösung gefunden werden soll. Der Wiederverwendungsprozeß beginnt mit der Suche in einer Planbibliothek, in der Informationen aus vorangegangenen Plangenerierungsprozessen gespeichert sind, wie zum Beispiel alte Planspezifikationen und die dazu generierten Pläne. Ziel des Suchprozesses ist es, ausgehend von der gegebenen aktuellen Spezifikation eine geeignete Menge von Wiederverwendungskandidaten, d. h. Plänen zu finden. Der Suchprozess in der Bibliothek muß dabei sicherstellen, daß falls eine Lösung für das aktuelle Planungsproblem in der Bibliothek existiert, diese auch gefunden wird. Existiert eine solche unmittelbare Lösung in der Bibliothek nicht, dann soll ein solcher Plan bestimmt werden, der mit nur geringem Änderungsaufwand zu einer Lösung modifiziert werden kann.
  Ergebnis der Plandeterminationsphase ist eine Menge von Wiederverwendungskandidaten. Diese Menge kann leer sein, wenn die Suche in der Planbibliothek erfolglos war. In diesem Fall kann das aktuelle Planungsproblem nicht durch das Wiederverwenden einer gespeicherten Bibliothekslösung bearbeitet werden und die Plangenerierung wird aktiviert. Konnten mehrere Wiederverwendungskandidaten bestimmt werden, dann können diese Kandidaten nach heuristischen Kriterien geordnet und ein "bester" Kandidat bestimmt werden. Dieser Kandidat ist die Eingabe für die nächste Phase.

- **Phase II: Plan Interpretation**

  Der in der Bibliothek gefundene Wiederverwendungskandidat umfaßt den wiederverwendbaren Plan, die Planspezifikation, die dieser Plan erfüllt und weitere Informationen. Als nächste Aufgabe muß nun festgestellt werden, ob dieser Bibliotheksplan tatsächlich auch eine korrekte Lösung des aktuellen Planungsproblems ist. Dies geschieht nicht direkt durch einen Vergleich von wiederverwendetem Plan und aktueller Planspezifikation sondern indirekt, indem die wiederverwendete und die

aktuelle Planspezifikation in der Planinterpretationsphase miteinander verglichen werden. Dabei soll festgestellt werden, welche Gemeinsamkeiten und Unterschiede in beiden Planungsproblemen auftreten. Da wir einen formallogischen Ansatz für diesen Vergleich entwickeln, wird während der Planinterpretation überprüft, ob das aktuelle Planungsproblem eine logische Instanz des alten, aus der Bibliothek gewonnenen Planungsproblems ist. Ist dies der Fall, dann ist die Lösung des alten Planungsproblems auch eine Lösung des aktuellen Planungsproblems, das heißt der wiederverwendete Plan ist hinreichend für die aktuelle Planspezifikation.
Wir sprechen deshalb von Planinterpretation, da durch den Vergleich beider Planungsprobleme der wiederverwendete Plan gleichzeitig in der aktuellen Planungssituation "interpretiert" wird, d. h. eine geeignete Instanziierung des wiederverwendeten Planes mit Parametern aus der aktuellen Planspezifikation gewonnen wird.
Wenn der Vergleich beider Planungsprobleme negativ ausfällt, d. h. nicht gezeigt werden konnte, daß das neue Planungsproblem eine Instanz des wiederverwendeten Planungsproblems ist, dann stellt der aus der Bibliothek gewonnene Plan keine Lösung für das aktuelle Planungsproblem dar und muß modiziert werden.
Die notwendigen Informationen für eine erfolgreiche Modifikation werden aus dem fehlgeschlagenen Vergleich der beiden Planungsprobleme gewonnen, da andere Informationsquellen für das System nicht zur Verfügung stehen.

- **Phase III: Plan Adaption**

  Wird die Planinterpretationsphase mit einem negativen Resultat beendet, dann erfordert das aktuelle Planungsproblem mehr Planungsaufgaben zu lösen, als der wiederverwendete Plan tatsächlich löst. Der in der Bibliothek Plan gefundene ist somit keine Lösung für das aktuelle Planungsproblem und muß verändert (angepaßt) werden. Aufgabe der Planadaption ist es, ausgehend von einer Analyse der Planinterpretationsphase, aus dem Bibliotheksplan ein Planskelett zu konstruieren. Dieses Planskelett enthält diejenigen Teile des Bibliotheksplans die zur Lösung des aktuellen Planungsproblems wiederverwendet werden können und stellt eine unvollständige Teillösung des aktuellen Planungsproblems dar. In einem nachfolgenden Planungsprozeß wird dieses Planskelett zu einem korrekten Plan vervollständigt. Dieser Planungsprozeß überprüft, ob die wiederverwendeten Teilpläne korrekt sind, d. h. Teilziele aus der aktuellen Planspezifikation erreichen und ergänzt weitere Teilpläne für die noch offenen Teilziele, die der wiederverwendete Plan nicht erreicht.
  Nach dem erfolgreichen Abschluß der Planadaption liegt eine korrekte Lösung des aktuellen Planungsproblems vor.

- **Phase IV: Aktualisierung der Planbibliothek**

  Der Wiederverwendungsprozeß endet mit einer Aktualisierung der Planbibliothek, bei der ein neuer Eintrag in die Bibliothek aufgenommen wird. Im Unterschied zu anderen, in der Literatur existierenden Ansätzen, gehen wir davon aus, daß die Planbibliothek keine vom Benutzer vor definierten Einträge umfaßt, sondern dynamisch vom System aufgebaut wird. Das Modell des Wiederverwendungsprozesses beinhaltet deshalb auch den Aufbau und die Pflege der Planbibliothek durch das System.
  Der Aufbau eines Eintrages für die Bibliothek erfolgt automatisch durch das System nach einem erfolgreich abgeschlossenen Generierungsprozeß, bei dem zu einer gegebenen Planspezifikation durch Plangenerierung oder durch Planmodifikation ein Plan als Lösung gefunden wurde. Ein Bibliothekseintrag enthält dabei diesen Plan mit seiner dazugehörigen Spezifikation sowie Informationen, die durch eine Analyse des

Generierungsprozesses gewonnen werden.
Die hierarchische Strukturierung der Planbibliothek erfolgt durch eine Indexierung der Planeinträge, wobei der Index jedes Planeintrags durch eine Kodierung der Planspezifikation bestimmt wird. Eine geschickte Indexierung der Planbibliothek ist eine wesentliche Voraussetzung für die effiziente Bestimmung von Wiederverwendungskandidaten durch die Plandeterminationsphase.

Das 4-Phasen Modell beschreibt den Planwiederverwendungsprozess als einen geschlossenen, vollautomatischen Zyklus. Die Phasen I bis III sind notwendig, um einen Plan durch Wiederverwendung einer gespeicherten Bibliothekslösung zu generieren. Die Phase IV dient dem Aufbau und der Pflege einer Wissensbasis in Form einer Planbibliothek, in der Wissen aus vorangegangenen Planungsprozessen gespeichert wird. Das hier vorgestellte Modell liefert die Grundlage einer allgemeinen Strukturierung des Gesamtsprozesses der Wiederverwendung von Plänen. Ausgehend von diesem Modell wurde ein einheitlicher Formalismus entwickelt, der die bisher in der Literatur existierende Trennung zwischen Formalismen zur Planmodifikation und Formalismen zur Repräsentation von Planbibliotheken überwindet.
Die abstrakte Sicht auf den Wiederverwendungsprozess bildete einerseits die Grundlage für die Entwicklung einer deduktiven Planwiederverwendungskomponente innerhalb des PHI Systems, andererseits motivierte sie eine allgemeine theoretische und empirische Analyse der Planwiederverwendung in Kooperation mit dem DFKI-Projekt WIP [NK93a, NK93b].

2.2.2 Realisierung einer deduktiven Wiederverwendungskomponente

Innerhalb des PHI Systems wurde erstmals ein deduktiver und formallogischer Ansatz für die Wiederverwendung von Plänen in deduktiven Planungssystemen entwickelt. Ausgehend vom 4-Phasen Modell betrachtet die logische Formalisierung solche Phasen zusammen, die ähnliche Aufgaben zu lösen haben.
Operationen in der Planbibliothek bilden die Grundlage für die Phasen I und IV und werden durch ein formales Wissensrepräsentationssystem und die auf ihm definierten Inferenzverfahren formalisiert. Das formale Wissensrepräsentationssystem basiert auf einem hybriden Repräsentationsformalismus, der die Logik LLP mit einer terminologischen Logik verbindet, um den Anforderungen, die die Repräsentation von Planbibliotheken stellt, gerecht zu werden: Wissen aus Planungsprozessen soll in einer *abstrakten* und *wiederverwendbaren* Form repräsentiert werden. Dies bedeutet, in Spezifikationen von Planungsproblemen von konkreten Objekten zu abstrahieren und das Wissen so in einer Bibliothek strukturiert zu repräsentieren, daß ein effizienter Zugriff darauf möglich ist.
Terminologische Logiken unterstützen die strukturierte Repräsentation abstrakten Wissens in besonders geeigneter Weise und lassen sich aufgrund ihrer formalen Semantik mit der Planungslogik in einen *hybriden Repräsentationsformalismus* integrieren. Die Verbindung der Planungslogik mit einer terminologischen Logik führte zu entscheidenden Vorteilen: Die Planbibliothek als ein Bestandteil eines deduktiven Planungssystems basiert auf den gleichen logischen Grundlagen wie das Planungssystem selbst.
Die Phasen II und III umfassen die eigentliche Modifikation von Plänen und sind durch ein deduktives Verfahren auf Planspezifikationen formalisiert [Koe93]. Dieses deduktive Verfahren wurde im für die Logik LLP entwickelten Sequenzenkalkül durch einen Beweis realisiert. Ähnlich wie bei der Plangenerierung werden für die Beweissteuerung Taktiken verwendet. PHI ist das erste System, in dem eine deduktive Planmodifikation entwickelt

wurde. Dieser deduktive Ansatz sichert, daß ein Plan, der durch Modifikation eines vorhandenen Plans gewonnen wurde, korrekt ist, d. h. tatsächlich eine Lösung des aktuellen Planungsproblems darstellt. Diese wichtige Eigenschaft kann nur von sehr wenigen, in der Literatur beschriebenen Planungssystemen gesichert werden.
Darüberhinaus ist das PHI System in der Lage, kompliziertere Pläne wiederzuverwenden und zu modifizieren als dies in anderen Systemen bisher möglich war. Existierende Systeme betrachteten bisher nur sequentielle Pläne, d. h. einfache Folgen von Aktionen konnten wiederverwendet und modifiziert werden. Im PHI System werden darüber hinaus auch Pläne mit Kontrollstrukturen, wie Fallunterscheidungen und Iterationen, behandelt.

2.3 Abduktive Planerkennung

Pläne zu erkennen, heißt die Pläne oder Ziele eines Benutzers ausgehend von den vorhandenen Evidenzen zu identifizieren. Diese Evidenzen umfassen z.B. die Aktionen des Benutzers oder auch Informationen über seine Präferenzen, die in einem Benutzermodell abgelegt sein können (vgl. z.B. [Car90b]).
Zu wissen, welche Ziele ein Benutzer verfolgt, ist eine entscheidende Voraussetzung für kooperatives Verhalten von Hilfesystemen (s. [GL92]). Planerkennungskomponenten bei Benutzungsschnittstellen erlauben zum Beispiel, die zukünftigen Aktionen eines Benutzers vorherzusagen und bilden damit die Basis für *semantische Planvervollständigung*. Das heißt dem Benutzer kann die automatische Ausführung des Restplans vom System vorgeschlagen werden. Desweiteren ist das Erkennen von *suboptimalem Benutzerverhalten* eine Voraussetzung für aktive Hilfe in Form von optimalen Planvorschlägen. Können außerdem die Ursachen für einen Fehler bei der Planausführung festgestellt werden - wie z.B. eine unerfüllte Planvorbedingung - so ist es möglich, dem Benutzer eine *flexible Fehlerbehandlung* anzubieten, die seinem aktuellen Kontext angepaßt ist. Kooperativere Hilfesysteme können auch durch den Einbau von *Planüberwachungs-* oder *Lernkomponenten* erreicht werden, die auf Planerkennungsergebnissen aufbauen.
Es ist wichtig, daß solche Systeme neben einer praktischen Anwendbarkeit auch formal fundiert sind (vgl. [GL92]). Ihre Funktionalität sollte dabei die Vorhersage möglicher Nachfolgeaktionen, inkrementelle Erkennung nach jeder Beobachtung sowie eine (probabilistische) Bewertung der Planhypothesen umfassen. Dies wird im Rahmen von PHI auf der Basis eines formalen Rahmens realisiert [BP93]. Durch abduktives Schließen in der Modallogik LLP wird inkrementell die Menge der jeweils gültigen Planhypothesen bestimmt. Durch eine auf Dempster-Shafer-Theorie basierende Selektion (vgl. Abschnitt 2.4) ist es möglich, jederzeit die "beste" Hypothese auszuwählen, um gezielt Hilfe anbieten zu können.
Das Problem der Planerkennung kann dabei als eine natürliche Anwendung des *abduktiven Schließens* betrachtet werden: Pläne sind im allgemeinen Hypothesen, die zum aktuellen Zeitpunkt und bezüglich des aktuellen Wissenstandes plausibel sind und die bisher beobachteten Aktionen erklären. Die Grundidee der Abduktion läßt sich durch eine Art invertierten *Modus Ponens* darstellen, das heißt aus einer Beobachtung ω und der Regel "φ impliziert ω" kann φ als eine *plausible* Hypothese oder Erklärung für ω inferiert werden (siehe auch [Pei58]). Abduktion ist also eine Form der "anfechtbaren" Inferenz, d.h. die berechneten Hypothesen sind lediglich plausibel und müssen verifiziert werden. Für einen detaillierten Überblick siehe [Pau93].
Im Zusammenhang mit Planerkennung wird Abduktion erst seit kürzerer Zeit betrachtet. Allerdings entwickelte sich sehr schnell die Auffassung, daß jede Art der Planerkennung einen abduktiven Prozeß darstellt (s. [May92]).

Jedoch existieren noch keine Ansätze zur Abduktion in einer temporalen Modallogik wie LLP. Im Rahmen des PHI-Projekts konnte gezeigt werden, daß die klassischen abduktiven Ansätze, die auf Prädikatenlogik basieren, im Falle einer Logik wie LLP sogar zu unintuitiven und unerwünschten Ergebnissen führen. Es wurde deshalb ein verallgemeinertes Abduktionsprinzip mit einer schwächeren Definition des Erklärungsbegriffs entwickelt. Das Grundprinzip dabei ist, die beobachteten Aktionen in die Planhypothesen so "einzubauen" , daß zum Beispiel temporale Abstraktionen, das heißt fehlende Angaben über eine zeitliche Reihenfolge der einzelnen Aktionen, aufgelöst werden können. Die Hypothesen werden also nach jedem Beobachtungsschritt mit der aktuellen Beobachtung *konkretisiert,* falls dies möglich ist. Dabei werden auch vorher ungebundene Parameter (d.h. Variablen) instantiiert.

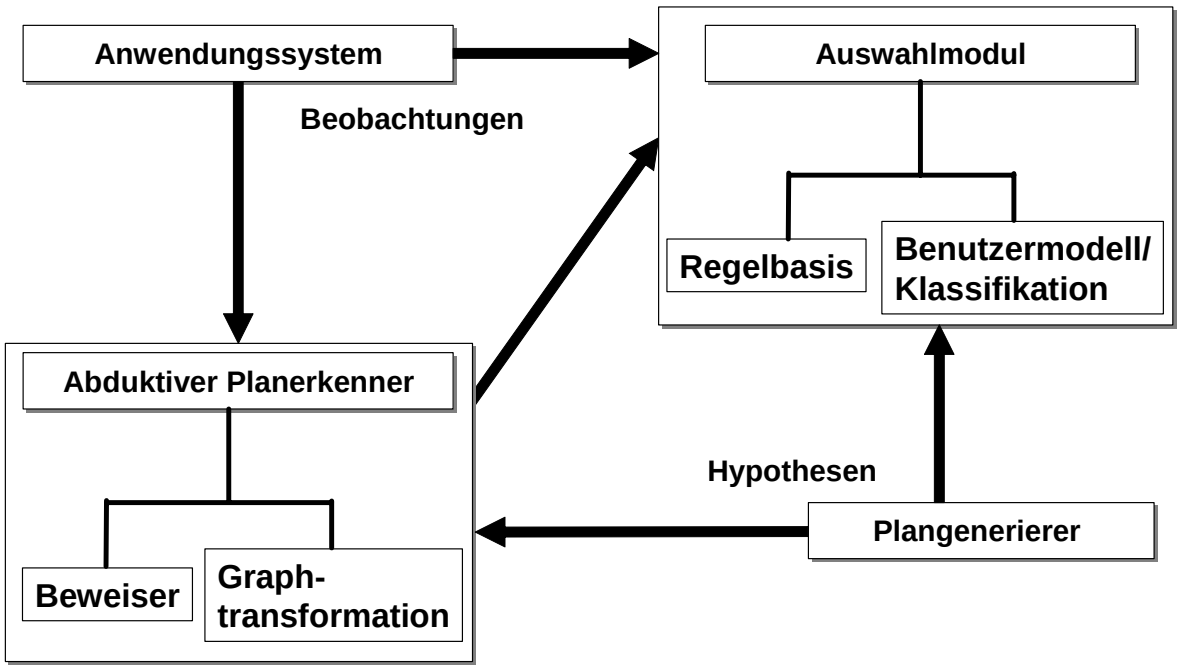

Abbildung 2: Architektur der Planerkennung in PHI

Die Gesamtarchitektur des Planerkenners sowie die Schnittstellen zu Anwendungssystem und Plangenerierung sind in Abbildung 2 dargestellt.
Der abduktive Planerkenner transformiert die vom Generierer erhaltenen Planhypothesen in eine Graphsyntax, die einen effizienteren Erkennungsprozeß ermöglicht, da der Graph einer Hypothese deren inhärente temporale Struktur explizit macht. Die Knoten des Graphen, in denen die einzelnen erwarteten Aktionen abgelegt sind, und Teilgraphen können durch unterschiedlich markierte Kanten miteinander verbunden werden, die die erwartete zeitliche Abfolge der Aktionen beschreiben.
Vom Anwendungssystem erhält der Erkenner zu jedem Zeitpunkt die aktuelle beobachtete Aktion des Benutzers. Die Hypothese wird beibehalten, falls es möglich ist, die Beobachtung in den Plan "einzubauen". Das heißt, die an den aktuellen Knoten abgelegten Aktionen sind mit der Beobachtung kompatibel, und die Hypothese kann damit konkretisiert werden. Zusätzlich werden vom Erkenner auch mögliche Planvorbedingungen durch Anfragen an das Anwendungssystem getestet. Sind diese nicht erfüllt, so muß die Hypothese verworfen werden.
Mit diesem Algorithmus ist es möglich, über Aktionssequenzen hinausgehend auch Kontrollstrukturen wie Schleifen oder Verzweigungen zu erkennen, was in klassischen

Planerkennern nicht der Fall ist. Desweiteren können in einem Plan auch abstrakte Aktionen enthalten sein. Um diese der konkret beobachteten Aktion zuordnen zu können, wird ein Beweiser verwendet, mit dessen Hilfe gezeigt werden kann, daß die konkrete Aktion die abstrakte impliziert.
Der bisher beschriebene Erkenner ermöglicht die Auswahl der Menge aller zu einem Zeitpunkt gültigen Planhypothesen. Um zu jedem Zeitpunkt dem Benutzer auf Anfrage einen Vorschlag für die automatische Fortführung seines Plans machen zu können, ist es sinnvoll, die Hypothesen weiter zu bewerten. Dies ermöglicht das im folgenden beschriebene Auswahlmodul.

2.4 Selektion von Planhypothesen

Der bisher beschriebene abduktive Planerkenner liefert als Ergebnis eine Menge gleichermaßen plausibler Planhypothesen. In bestimmten Situationen kann es jedoch notwendig sein, sich für eine der Alternativen zu entscheiden. Wenn der Benutzer beispielsweise um Hilfe zur Vervollständigung seines Plans fragt, muß ein Maß zur Bewertung der "Qualität" der Planhypothesen vorhanden sein, mit dessen Hilfe die "beste" unter ihnen herausgesucht wird. Ein solches Auswahlkriterium kann man erhalten, wenn das Wissen über das typische Benutzerverhalten und die möglichen Auswirkungen zusätzlicher Beobachtungen in einem numerischen Formalismus zur Behandlung von Unsicherheit kodiert wird, wobei die numerischen Werte die Wahrscheinlichkeiten der einzelnen Hypothesen repräsentieren. Als Kandidaten für einen solchen Formalismus bieten sich u.a. Wahrscheinlichkeitstheorie und Dempster-Shafer-Theorie (DST) an. Ähnlich wie in [Car90a] wurde in PHI eine erweiterte Version von DST eingesetzt. Die Gründe dafür sind folgende: DST (s. [Sha76]) erlaubt die Arbeit mit unterspezifizierten Modellen. D.h., daß es im Unterschied zur Wahrscheinlichkeitstheorie nicht notwendig ist, *alle* bedingten und Vorabwahrscheinlichkeiten zu kennen oder Unabhängigkeitsannahmen einzuführen, um Berechnungen durchführen zu können. Stattdessen genügt es, untere und obere Schranken für die Wahrscheinlichkeiten einiger Ereignisse anzugeben und den Rest unspezifiziert zu lassen. Dies entspricht genau unserer Situation: Selbst mit Langzeitbeobachtungen des Benutzers ist es kaum möglich, die exakte Wahrscheinlichkeit einer gegebenen Planhypothese zu bestimmen - unser Wissen über den Benutzer wird immer unvollständig bleiben. DST erlaubt es, diese partielle Unwissenheit zu berücksichtigen und von Unsicherheit zu unterscheiden, während Wahrscheinlichkeitstheorie die Anwendung eines Meta-Kriteriums wie *maximale Entropie* erfordert, um die vorhandene Information künstlich zu vervollständigen.
Der zweite Grund ist, daß die Verwendung von Dempsters Regel zur Kombination unabhängiger Informationen es ermöglicht, die schrittweise Einschränkung der Menge der möglichen Hypothesen als Folge neuer Beobachtungen adäquat zu modellieren, was eine wesentliche Voraussetzung für die inkrementelle Planerkennung ist. DST als der zugrundeliegende Basisformalismus zur Behandlung von Unsicherheit stellt somit ein Werkzeug dar, mit dessen Hilfe die Ausgangssituation in einer dem tatsächlichen Wissensstand entsprechenden Granularität ebenso modelliert werden kann wie der dynamische Prozeß der Aktualisierung dieser Beschreibung beim Auftreten neuer Informationen. Im Gegensatz dazu müssen auf Wahrscheinlichkeitstheorie aufbauende Systeme wie Wimp3 (s. [CG91]), in dem dynamisch generierte Bayessche Netzwerke verwendet werden, zahlreiche Annahmen bzgl. einer gleichmäßigen Verteilung von Wahrscheinlichkeiten machen, die nicht aus der gegebenen Situation heraus begründet werden können. Ein Problem bei der Verwendung von DST in PHI ist die Tatsache, daß die Planhypothesen dynamisch generiert werden, so daß keinerlei Wissen über ihre

Vorabwahrscheinlichkeiten vorhanden ist. Es ist jedoch möglich, aus dem beobachteten Benutzerverhalten typische Aktionssequenzen und oft verfolgte Standardziele zu extrahieren, deren relative Häufigkeiten statistisch ermittelt werden können. Beide Arten von Information können als LLP-Formeln repräsentiert werden und dienen zur Klassifikation von Plänen. Wird nun eine Menge von Planhypothesen generiert, so ist es möglich, diese mittels einer speziellen Taktik denjenigen Zielbeschreibungen und Teilplänen zuzuordnen, die in ihnen enthalten sind. Somit können die Vorabwahrscheinlichkeiten der Klassifikationsformeln an die einzelnen Hypothesen vererbt werden.
Als Seiteneffekt dieser Klassifikation ist es möglich, bestimmte Pläne zu kennzeichnen, die bestimmte als suboptimal erkannte Aktionssequenzen beinhalten. Wird ein solcher Plan als Hypothese für das Verhalten des Benutzers erkannt, so kann dieser in einem tutoriellen Modus auf diese Tatsache hingewiesen werden. Optional kann der Plangenerierer dazu verwendet werden, einen optimalen Plan für das aktuell verfolgte Ziel zu erzeugen, der dem Benutzer als Alternative präsentiert wird.
Die Verbindung zwischen den beobachteten Aktionen und den zugehörigen Planhypothesen stellen Regeln dar, die aus der Beschreibung der einzelnen Pläne als LLP-Formeln gewonnen werden und eine gewichtete Beziehung zwischen Beobachtungen und Hypothesen repräsentieren. Die intuitive Semantik dieser Regeln ist

"wenn **aktion1** beobachtet wird und **Bedingung1** erfüllt ist,
so weise der Hypothese H1 den Vertrauensgrad **x1** zu."

Dabei steht **Bedingung1** für die Tatsache, daß die beobachtete Aktion alle strukturellen Anforderungen der Hypothese erfüllt wie Bindung der aktuellen Parameter und temporale Struktur des Plans. Die entsprechende Information wird vom abduktiven Planerkenner geliefert durch die Mitteilung, ob Hl durch **aktion1** konkretisiert werden konnte. Wird nun eine bestimmte Aktion beobachtet, so wird mittels der anwendbaren Regeln eine sogenannte *Massenverteilung* über der Menge der Planhypothesen induziert. Diese Verteilung von Vertrauensgraden kann nun mit Dempsters Regel mit der bisherigen Beurteilung der Hypothesen kombiniert werden. Mithilfe dieser aktualisierten Bewertung ist es jederzeit möglich, nach verschiedenen Kriterien wie höchster Plausibilität oder wahrscheinlichster Einzelhypothese eine geeignete, benutzerspezifische Auswahl zu treffen, die dann z.B. als Basis für semantische Planvervollständigung verwendet wird.

3 PLUS: Ein Hilfesystem für direkt-manipulative Schnittstellen

Im Gegensatz zu den meisten bisherigen Hilfesystemen, die für kommando-orientierte Schnittstellen entwickelt wurden (siehe z.B. [BBD+93, Fin83, FLS85, WHK88, WCL+88]), arbeitet PLUS in Applikationen, die dem Benutzer graphische Benutzungsoberflächen zur Verfügung stellen, deren Interaktion auf dem Prinzip eines benutzergeführten Dialogs mittels direkter Manipulation [Shn83, Shn87] basiert - sogenannte „Direkt-Manipulative Benutzungsschnittstellen“ (DMI).
Die Besonderheit solcher DMI-Umgebungen im Vergleich zu kommandobasierten Schnittstellen liegt in der großen Flexibilität des Benutzers bei der Ausführung von Aktionen. Die zu einem Plan gehörenden Aktionen unterliegen i.a. weder einer strengen Reihenfolgebeziehung, noch ist ein enger zeitlicher Zusammenhang für ihre Ausführung erforderlich. Der Benutzer kann mehrere Pläne parallel verfolgen und beliebig zwischen

ihnen hin- und herspringen. Erleichtert diese Flexibilität einerseits dem erfahrenen Anwender die Arbeit mit einem solchen System, so kann andererseits der im Umgang mit einer DMI-Oberfläche ungeübte Benutzer aufgrund der entstehenden Komplexität leicht auf Probleme stoßen. Das im PLUS-Projekt zu entwickelnde Hilfesystem soll dabei einen menschlichen Experten ersetzen, den ein Benutzer in einer solchen Situation um Hilfe bitten würde oder der ihm bei seiner Arbeit „über die Schulter schaut“ und ihm gegebenenfalls Ratschläge gibt, sobald er ein ineffizientes oder fehlerhaftes Vorgehen erkennt. Wie ein solcher Experte ist unser Hilfesystem im Gegensatz zu Handbüchern oder einer statischen Online-Hilfe in der Lage, kontextabhängig auf die Probleme des Benutzers einzugehen, indem es ihm Ratschläge bezüglich der von ihm aktuell bearbeiteten Aufgaben erteilt.

3.1 Das System PLUS

Es existieren zwei grundlegend verschiedene Ansätze zur Realisierung planbasierter Systeme. Man unterscheidet zwischen Systemen, bei denen zur Laufzeit mittels eines Plangenerierungssystems Pläne erzeugt werden, die als Grundlage für den Planerkenner dienen (vgl. [AKPT91, BBD+93]), und Systemen, bei denen eine vorgefertigte Planbasis die Grundlage für den Erkennungsprozeß bildet. Innerhalb von PLUS [BFK+93, TB92] wird der zweite Ansatz verfolgt.

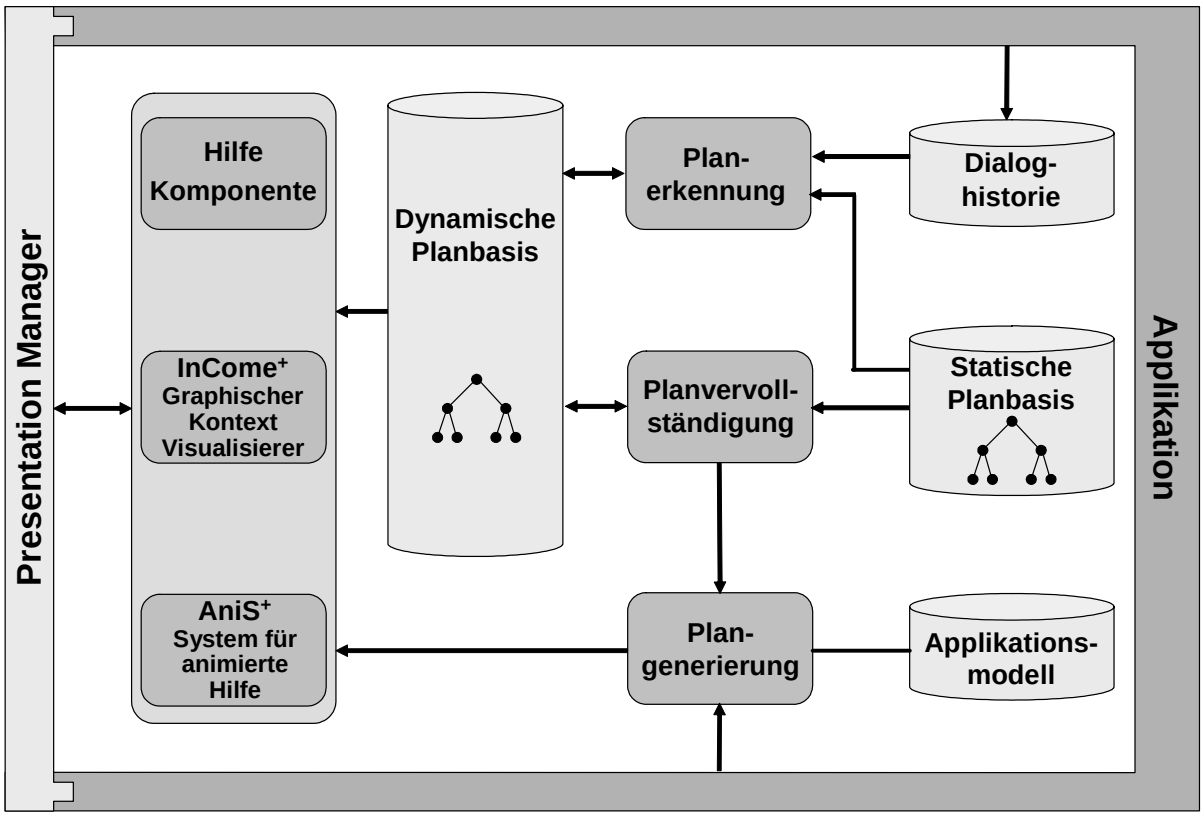

Abbildung 3: PLUS System-Architektur

Vom Benutzer ausgelöste und von der Applikation erfolgreich ausgeführte Aktionen werden in einer Dialoghistorie gespeichert. Sie dienen als Eingabe für den Planerkennungsprozeß, der durch das Modul *Planerkennung* realisiert wird.
Der Planerkenner versucht mit Hilfe eines *Spreading-Activation-Algorithmus* die in der

Dialoghistorie gespeicherten Aktionen auf Pläne, die in einer hierarchischen Planbasis vordefiniert sind, abzubilden. Der Planerkenner arbeitet synchron zur Applikation, d.h. in die Dialoghistorie eingehende Aktionen werden sofort an ihn weitergeleitet. Eine Hierarchisierung der Planbasis ist unerläßlich, um dem Benutzer adäquate Unterstützung auf einem geeigneten Abstraktionsniveau geben zu können und um einen effizienten Planerkennungsprozeß zu gewährleisten. Da logisch zusammenhängende Aktionssequenzen sehr oft in verschiedenen Plänen als Teilsequenzen auftauchen, ist es naheliegend, diese als eigenständige Pläne zu definieren, und sie dann als Teilpläne der entsprechenden abstrakteren Pläne zu definieren. Dadurch ergibt sich eine Planhierarchie mit mehreren Ebenen. Diese sogenannte *statische Planbasis* wird mit Hilfe der Plansprache *GPL*$^+$ definiert und kann mittels des graphikorientierten Planeditors PlanEdit$^+$ interaktiv spezifiziert werden.

3.2 Die Plansprache GPL$^+$

In der statischen Planbasis werden drei Typen von Objekten unterschieden: *Aktionen, Pläne* und *Ziele.* Sie ist wie folgt aufgebaut (Abbildung 4):

- Die unterste Ebene besteht aus den *Aktionen*, die der Benutzer bei seiner Arbeit mit der Applikation ausführen kann. Aktionen können Teile von Plänen sein.

- Ein *Plan* setzt sich aus einer Menge von Aktionen und/oder Teilzielen zusammen. Mit jedem Plan ist genau ein Ziel assoziiert, das durch die Ausführung des Planes erreicht wird.

- Ein *Ziel* kann auf verschiedenen Wegen erreicht werden, wobei jeder Weg einem Plan entspricht. Einige davon können suboptimal oder sogar falsch sein. Ziele können wie Aktionen Teile von abstrakteren, d.h. in der Hierarchie höher angesiedelten Plänen sein.

Die Plansprache GPL$^+$ stellt Definitionsschemata für Aktionen, Pläne und Ziele zur Verfügung. Mit Hilfe dieser Schemata können die unterschiedlichen Elemente der Planbasis definiert werden, indem für jedes Elemente ein entsprechendes Schema vom Applikationsdesigner mit den passenden Werten gefüllt wird. In GPL$^+$ werden Pläne primär als Mengen von Aktionen und Subzielen verstanden, es können jedoch durch Constraints relative und absolute temporale Beziehungen zwischen Elementen eines Planes definiert werden. Daneben sind weitere Constraints zur Modellierung von Parameterabhängigkeiten, Objekthierarchien und zur Iteration einzelner Elemente spezifizierbar. Zusätzlich können allgemein bei Planerkennung auftretende Merkmale wie *Planabbruch* und *Planinteraktionen* abgebildet werden. Durch die Trennung der statischen Planbasis (Abbildungsvorschrift) von der dynamischen Planbasis (Dialogmodell) können weitere Planinteraktionen wie *multiple Planhypothesen, parallele Pläne* und die Verfolgung mehrerer, zu einem Ziel führender *alternativer Pläne* modelliert werden.

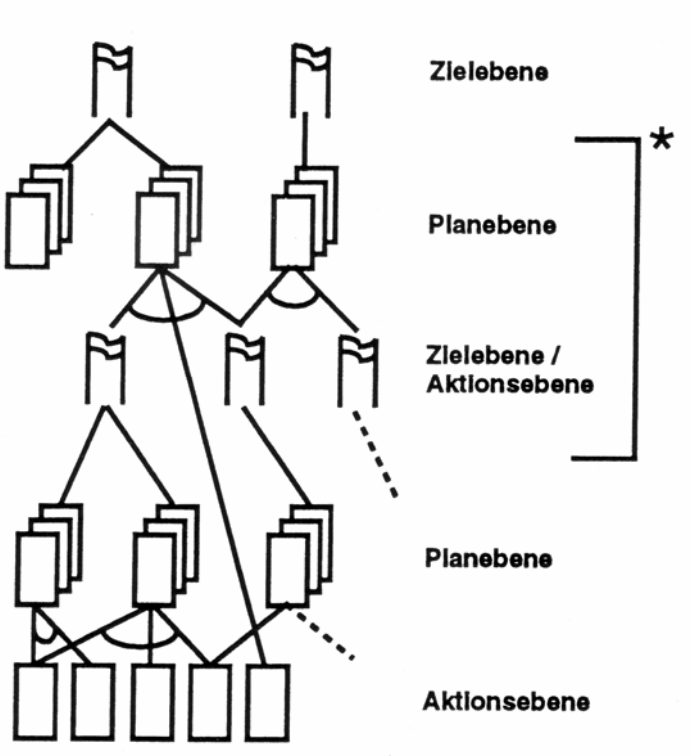

Abbildung 4: Struktureller Aufbau der statischen Planbasis[1]

3.3 Das Planerkennungsverfahren

Der Spreading-Activation-Algorithmus baut zur Laufzeit eine *dynamische Planbasis* auf. Das in der dynamischen Planbasis enthaltene Wissen (in Form von Planhypothesen) über die aktuell vom Benutzer verfolgten Pläne und Ziele wiederum dient als Grundlage für die Hilfekomponente sowie für InCome$^+$ und AniS$^+$ (Abbildung 3).
Informell kann der *Spreading-Activation-Algorithmus* als die Steuerung der Ausbreitung von Aktivierungen beschrieben werden. Hierbei gehen initiale Aktivierungen von Aktionen aus. Diese geben ihre Aktivierungen an Pläne weiter, die dann die Aktivierung an ihre Ziele weiterleiten. Ziele geben ihre Aktivierungen analog zu Aktionen an Pläne weiter. Die Menge der aktivierten Aktionen, Pläne und Ziele bildet die dynamische Planbasis (das Dialogmodell) und repräsentiert somit zu jedem Zeitpunkt den aktuellen Dialogzustand. Die Ausbreitung von Aktivierungen endet, wenn ein Ziel nicht Subziel ist, also nicht in anderen Plänen enthalten ist. Sie endet ebenfalls, wenn die Aktivierung aufgrund von unerfüllten Constraints nicht weitergeleitet werden konnte.
Ein einfaches Beispiel verdeutlicht das Prinzip. Die im linken Teil der Abbildung 5 dargestellte statische Planbasis ist die Grundlage des Beispiels.
Nach Ausführung der Aktion **a** durch den Benutzer wird in der dynamischen Planbasis eine Instanz für sie erzeugt. Nach der Verifikation von Constraints gibt die Aktion ihre Aktivierung an die Pläne weiter, für die alle Constraints erfüllt waren; im Beispiel sind es

[1] Die Semantik des Operators * entspricht der des Kleene'schen Operators; 0 - *n* Wiederholungen.

die beiden Planhypothesen **P** und **Q** (siehe mittleren Teil der Abbildung 5).
Als nächste Aktion führt der Benuzter **b** durch. Diesmal kann Aktion **b** nur der Planhypothese **P** zugeordnet werden. Aktion **b** gibt ihre Aktivierung an diese Planhypothese weiter. Da keine weiteren Aktionen oder Subziele für die Durchführung von Planhypothese **P** notwendig sind, gilt sie als erkannt. Sie gibt nun ihre Aktivierung an ihr Ziel weiter, das daraufhin versucht, die Aktivierung weiter an andere Planhypothesen zu reichen (vgl. rechten Teil der Abbildung 5, Häkchen markieren erkannte Pläne bzw. erreichte Ziele).

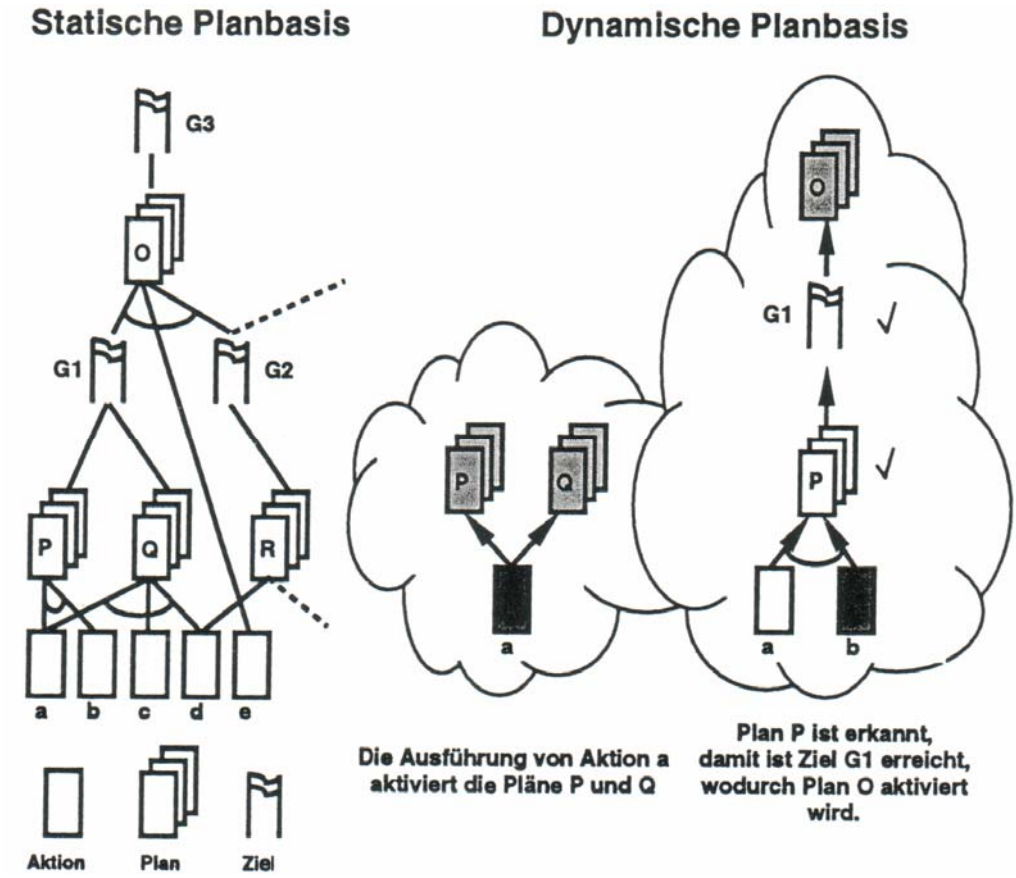

Abbildung 5: Funktionsweise des Spreading-Activation-Algorithmus

Aufgrund der großen Flexibilität des Benutzers bei der Bearbeitung seiner Aufgaben kann die Zahl der in der dynamischen Planbasis enthaltenen Hypothesen sehr schnell wachsen. Um sie einzuschränken, verwenden wir eine Reihe von Fokussierungstechniken:

- Mit Hilfe von *Time Frames* werden die in der dynamischen Planbasis enthaltenen Pläne in verschiedene Klassen unterteilt. Diese Klassifizierung wird beispielsweise dazu verwendet, dem Benutzer bei einer Hilfeanforderung eine geeignete Auswahl der Planhypothesen zu präsentieren.

- Für jeden Plan können *Cancel-Aktionen* definiert werden, deren Ausführung durch den Benutzer zum sofortigen Verwerfen der jeweiligen Planhypothese führt. Ein typisches Beispiel für eine Cancel-Aktion ist das Löschen eines Objekts, das in der Parameterliste eines Planes enthalten ist.

Durch die Beschränkung der Anzahl der Planhypothesen ist der Einsatz dieser Fokussierungstechniken auch ein wesentliches Hilfsmittel zum Erzielen einer annehmbaren Laufzeit, die wiederum ein wichtiges Kriterium für die Akzeptanz des Hilfesystems beim Benutzer ist.
Während des Planerkennungsvorgangs werden nur Aktionen verarbeitet, die nicht zur Navigation auf der Oberfläche dienen. Unter *navigatorischen Aktionen* werden z.B. Aktionen zum Selektieren von Objekten, zum Hinzufügen oder Entfernen von Objekten aus der Darstellung eines Fensters oder zum Verschieben des sichtbaren Ausschnittes eines Fensters verstanden. Diese Aktionen sind prinzipiell Bestandteil eines jeden Planes; würden sie mit Hilfe von GPL$^+$ modelliert werden, würde der Planerkenner nach jeder beobachteten, navigatorischen Aktionen sämtliche Pläne als intendiert markieren. Dies ist jedoch kein gewünschter Effekt und motiviert die alleinige Berücksichtigung navigatorischer Aktionen während des Plangenerierungsprozesses.
Zur Komplettierung begonnener Planhypothesen wird eine Planvervollständigungskomponente eingesetzt. Diese generiert Komplettierungssequenzen, die sich auf nicht navigatorische Aktionen beschränken, da die Planvervollständigung die Planspezifikationen der statischen Planbasis als Vorlag zur Komplettierung verwendet. Bekannte Parameterwerte werden gemäß definierter Parameterconstraints entlang der generierten Sequenz propagiert.

3.4 Kontextsensitiver Einstieg für eine hypertextbasierte Hilfe

Das PLUS-System enthält drei verschiedene Komponenten (Hilfekomponente, InCome$^+$ und AniS$^+$) zur Erzeugung von Hilfestellungen. Die einfachste Komponente des Systems ist die in Abbildung 3 bezeichnete *Hilfekomponente.* Sie bietet einen kontextsensitiven Einstieg in ein hypertextbasiertes Hilfesystem. Anhand der aktuellen Planhypothesen in der dynamischen Planbasis erfolgt eine Eingrenzung der Themenauswahl, über die der Benutzer in das hypertextbasierte Hilfesystem einsteigen kann. Nachdem der Benutzer über diesen Einstiegsdialog in das Hilfesystem gewechselt ist, kann er dort auf die gesamte Information jenes Hilfessystems zugreifen.

3.5 Die Komponente InCome$^+$

Eine zentrale Komponente der graphischen Hilfe im PLUS System ist der *Interaction Control Manager* InCome$^+$ [FT91, Thi90]. Er erzeugt eine graphische Visualisierung des aktuellen Dialogmodells, der Dialoghistorie und der möglichen zukünftigen Interaktionsschritte, die der Benutzer ausführen kann, um bestimmte Ziele zu erreichen. InCome$^+$ bietet somit dem Benutzer eine schnelle und hilfreiche Erinnerungsstütze, um zum Beispiel eine temporär unterbrochene Arbeit mit dem Computer wieder aufzunehmen. Es unterstützt den Benutzer beim Verlassen von Systemzuständen, die ihm nicht vertraut sind, und beim explorativen Agieren [Pau89], indem die nächsten möglichen Interaktionsschritte zum Erreichen eines Ziels visualisiert werden. InCome$^+$ erfüllt folgende Anforderungen:

- Adäquate Visualisierung von Benutzerinteraktionen,
- Darstellung verschiedener Abstraktionsebenen von Plänen (interaktiv veränderbar),
- Visualisierung möglicher zukünftiger Interaktionen,

- Graphische Navigationsfunktionen,
- Darstellung von Planinteraktionen wie Planeinbettung, Planüberlappung und Planunterbrechung, und
- Semantische Undo/Redo Möglichkeiten.

Die beiden Komponenten, auf die InCome$^+$ zurückgreift, sind der Planerkenner und die Planvervollständigungskomponente.
InCome$^+$ wird fortlaufend über den Planerkennungsprozeß informiert. Aus den eingehenden Daten generiert InCome$^+$ eine interne Darstellung des Interaktionskontextes und visualisiert diese analog zu einem gerichteten Graphen auf dem Bildschirm. Die Instanzen der Objektklassen *Plan, Aktion* und *Ziel* sind durch Knoten dargestellt und maussensitiv. Eine Aktion wird als einzelnes Blatt Papier, ein Plan als ein Stapel von Blättern visualisiert. Das zu einem Plan gehörende Ziel wird am Ende der Aktionssequenz des Plans durch eine stilisierte Zielfahne visualisiert. Um die Sequenz der Elemente eines Planes wiederzuspiegeln, werden die Knoten eines Plans mit Pfeilen verbunden (Abbildung 6). Die Darstellung des Graphen reflektiert von oben nach unten die chronologische Reihenfolge der ausgeführten Aktionen. Da InCome$^+$ nicht nur die bereits ausgeführten Aktionen und erkannten Pläne visualisiert, sondern auch mögliche zukünftige Interaktionen, werden zur Unterscheidung dieser beiden Klassen die dargestellten Knoten in unterschiedlichen Farben visualisiert.
Benutzeraktionen in InCome$^+$ können in drei Kategorien aufgeteilt werden: *graphische Navigation, hierarchische Navigation* und *indirektes Interagieren* mit der Applikation.
Graphische Navigation enthält Aktionen wie *scrolling* und *suchen* von Knoten nach bestimmten Suchkriterien. Zusätzlich kann die *lineare Dialoghistorie* in einem separaten Fenster angezeigt werden. In dieser Darstellung werden *reversible Aktionen* und in der Applikation gesetzte *freezing-points* visuell hervorgehoben.
Hierarchische Navigation erlaubt die Darstellung visualisierter Pläne in verschiedenen Abstraktionsstufen. Durch die Modellierung der unterschiedlichen Planinteraktionen in der hierarchischen Planbasis und deren Behandlung während des Planerkennungsprozesses baut InCome$^+$ eine Visualisierung auf, welche die Planinteraktionen reflektiert. Es werden Aktionen zum *Expandieren* und zum *Abstrahieren* von Plänen angeboten. Zusätzlich zu diesem stufenweisen vertikalen Bewegen in der Hierarchie wird eine analog zu einem *Fish-Eye Objektiv* arbeitende Funktion angeboten. Dabei wird jedes Objekt, welches nicht im Interessenbereich des Benutzers liegt, soweit wie möglich abstrahiert, ohne die Abstraktionsebene des fokussierten Bereichs zu verändern.
Ein **indirektes Interagieren** mit der Applikation wird durch einen *tutoriellen Modus* und durch den Zugriff auf *Undo-* und *Redo-Mechanismen* der Applikation ermöglicht. Der Benutzer selektiert ein Ziel, zu dem er „geführt" werden möchte, und aktiviert den tutoriellen Modus. InCome$^+$ erfragt daraufhin eine effiziente Aktionssequenz zum Erreichen des gewählten Ziels von der Planvervollständigungskomponente. Die erhaltene Aktionssequenz wird in einem separaten Fenster textuell in einer Art *To-Do-List* dem Benutzer visualisiert. Von jetzt an beobachtet InCome$^+$ die Interaktionsschritte des Benutzers und vergleicht diese mit der *To-Do-List.* Die vom Benutzer korrekt ausgeführten Aktionen der *To-Do-List* werden mit einer Markierung versehen. Nach jeder Aktion wird die *To-Do-List* entsprechend sortiert, damit sie die nun gültige Aktionssequenz zum Erreichen des gewählten Ziels reflektiert.
Hat der Benutzer eine Aktion ausgeführt, die es unmöglich macht, das gewählte Ziel

zu erreichen, informiert ihn InCome$^+$ über diesen Zustand. Kann die Aktion storniert werden oder kann der Systemzustand, der vor der Ausführung der Aktion gültig war, erreicht werden, zeigt InCome$^+$ dem Benutzer die notwendigen Schritte zum direkten (via reversibler Aktionen) bzw. indirekten (via Zurücksetzen auf *freezing-points und* eventuelles *Redo* von Aktionen) Stornieren der Aktion an.

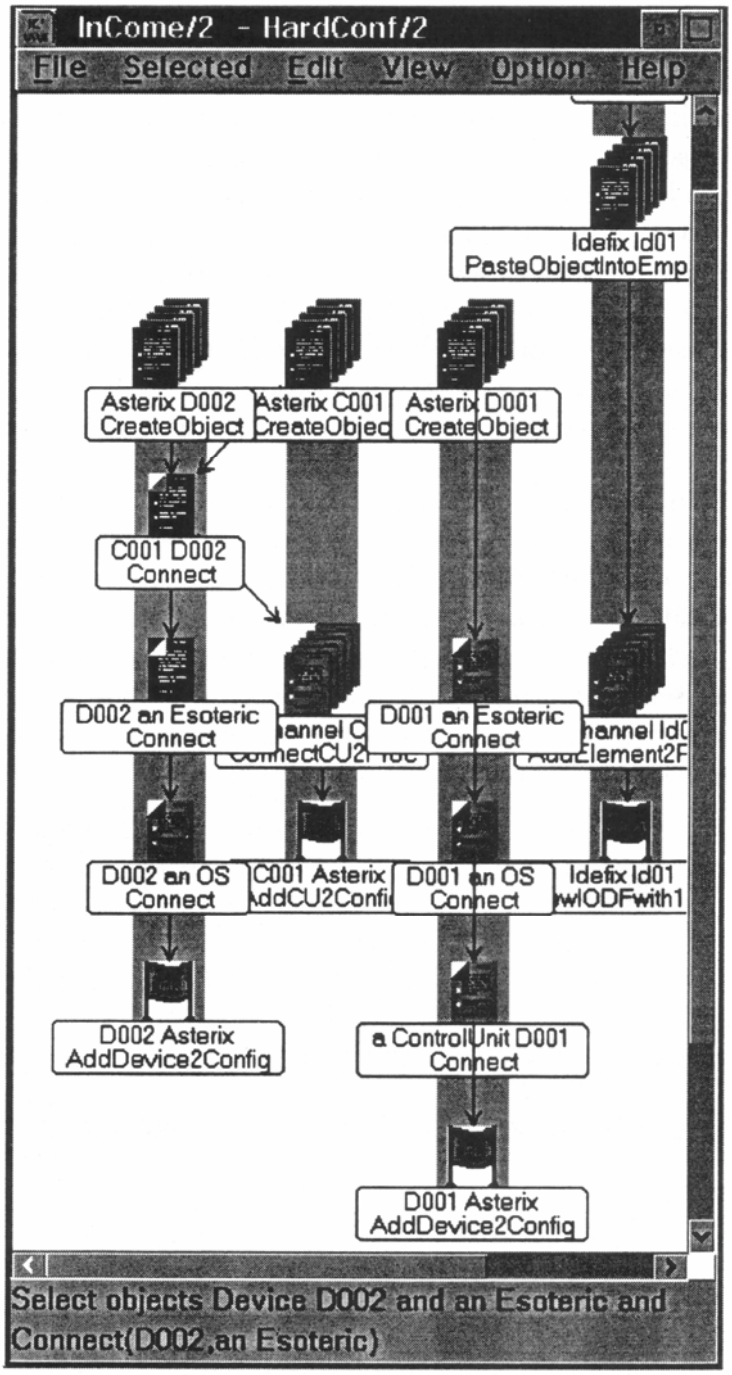

Abbildung 6: Darstellung des Interaktionskontextes durch InCome$^+$

Um zwei verschiedene Undo Prinzipien (funktionsorientiert vs. zustandsorientiert, [Rat87, Rat89, Yan90]) behandeln zu können, arbeitet InCome$^+$ mit einem erweiterten funktionsorientierten Ansatz unter Verwendung von *freezing-points* [Pau89], auf die mit Hilfe von Applikationsfunktionen zurückgesetzt werden kann. Durch die Darstellung des

Interaktionskontextes auf einer abstrakteren Ebene als der Aktionsebene ist der Benutzer in der Lage, Stornierungen auf einer *semantischen Ebene* (der Planebene) auszudrücken. Die Unterstützung eines *freien Undos* [Rat87] ist im jetzigen System nicht vorgesehen.

3.6 Die Komponente AniS$^+$

Einen wesentlichen Aspekt der graphischen Benutzerunterstützung stellt die Anbindung animierter Hilfe an das planbasierte System dar. Hilfesysteme mit einer rein textuellen Darstellung stoßen an ihre Grenzen, sobald der Benutzer Unterstützung zur Durchführung einzelner Interaktionsschritte einer Applikation benötigt, wenn es also um die Beantwortung von Fragen oder Aufforderungen folgender Form geht: *„Wie füge ich die Datei* brief.txt *in dem Verzeichnis* Beispiel *hinzu* ?“ oder *„Zeige mir bitte, wie ich nur die Objekte X, Y und Z angezeigt bekomme.*“. Ein generierter Hilfetext könnte möglicherweise lauten: *„Bewege die Maus zu der Position der Datei* brief.txt *und drücke die linke Maustaste nieder. Bewege nun mit niedergedrückter Taste die Maus zu der Position des Verzeichnisses* Beispiel. *Lasse die Maustaste wieder los.*“. Es wird deutlich, daß eine animierte Sequenz der Interaktionsschritte dem Benutzer eine adäquatere Unterstützung bietet.
Innerhalb des PLUS-Projekts ist eine Animationskomponente entwickelt worden, die im Gegensatz zu bisherigen Systemen mit animiert dargestellter Hilfe, wie z.B. das System *GAK (Graphical Animation from Knowledge,* [Nei82]) oder die animierte Hilfe im System *UIDE* [FGKK88, SF90], einen stärkeren Bezug zur momentan vom Benutzer verfolgten Aufgabe erreicht. Durch die Anbindung der Animationskomponente *AniS*$^+$ an den Planerkenner und die Planvervollständigungskomponente kann sie auf Anfrage gezielt für eine Planhypothese eine Sequenz von Animationsschritten generieren, wobei durch den Erkennungsprozeß bereits bekannte Parameterwerte propagiert werden.

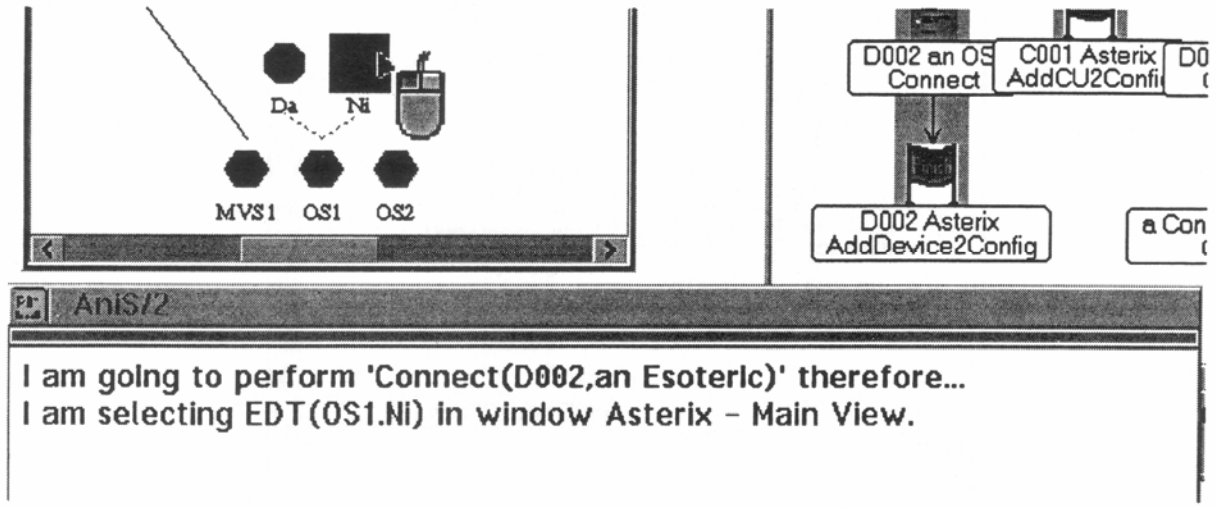

Abbildung 7: AniS$^+$ generiert kommentierte animierte Darstellungen…

Die Generierung der Animationsschritte erfolgt in zwei Phasen [Thi93, Thi94]. In der ersten Phase komplettiert die Planvervollständigungskomponente eine Planhypothese durch die Generierung einer Aktionssequenz. Diese Aktionssequenz ist die Grundlage für die inkrementelle Generierung von Animationsschritten, die zur Aktivierung der Aktionen der Sequenz notwendig sind (zweite Phase). Die Plangenerierungskomponente arbeitet mit einem backward-chaining-Algorithmus, der auf die im Applikationsmodell

für jede Aktion definierten Vor- und Nachbedingungen zugreift. Durch die inkrementelle Generierung kann auf den jeweils nach Ausführung einzelner Animationsschritten veränderten Benutzungsoberflächenkontext Bezug genommen werde.
Die Durchführung der Animationsschritte erfolgt durch simulierte Mauseingaben, die von AniS$^+$ erzeugt werden. Diese werden der Benutzungsoberfläche so zugeführt, daß diese reagiert, als würden die Eingaben vom Benutzer stammen.
Das Applikationsmodell ist in die Teile *aktions-, animations-, generisches* und *schnittstellenspezifisches* Wissen untergliedert. Das Modell dient als eine Erweiterung der statischen Planbasis und ist nur durch die Teile des *aktions-* und *schnittstellenspezifischen* Wissens applikationsabhängig.

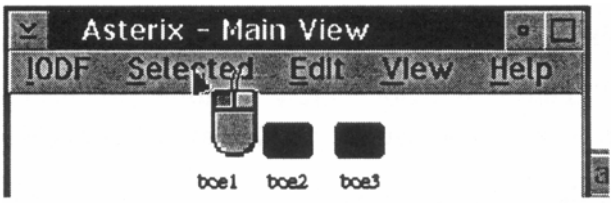

Abbildung 8: ...und...

Durch die Repräsentation von generischen Interaktionsrichtlinien im Applikationsmodell können auch navigatorische Animationsschritte generiert werden. So werden z.B. Animationssequenzen zum Verschieben des sichtbaren Bereichs eines Fensters oder zum Hinzufügen von Objekten, die in der aktuellen Darstellung ausgeblendet sind, generiert.

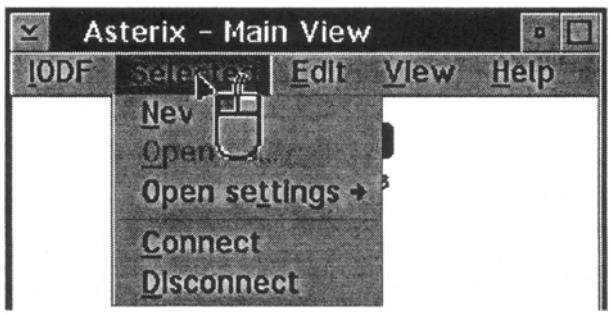

Abbildung 9: ...simuliert...

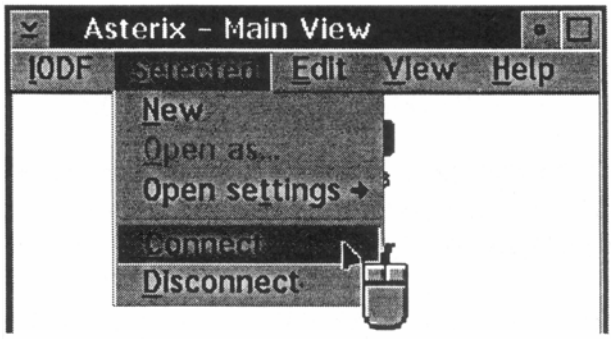

Abbildung 10: ...Mausinteraktionen...

Literatur

[AKPT91] J. F. Allen, H. A. Kautz, R. N. Pelavin, and J. D. Tenenberg, editors. *Reasoning about Plans.* Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1991.

[BBD+93] M. Bauer, S. Biundo, D. Dengler, J. Koehler, and G. Paul. PHI - A Logic-Based Tool for Intelligent Help Systems. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence,* Chambéry, France, 1993.

[BDK92] S. Biundo, D. Dengler, and J. Koehler. Deductive planning and plan reuse in a command language environment. In B. Neumann, editor, *Proceedings of the 10th European Conference on Artificial Intelligence,* pages 628-632, Vienna, Austria, August 1992. John Wiley & Sons.

[BFK+93] F. Berger, T. Fehrle, K. Klöckner, V. Scholles, M. A. Thies, and W. Wahlster. PLUS - Plan-based User Support. Research Report RR-93-15, DFKI, 1993.

[Bib86] W. Bibel. A deductive solution for plan generation. *New Generation Computing,* 4:115-132, 1986.

[BP93] M. Bauer and G. Paul. Logic-Based Plan Recognition for Intelligent Help Systems. In *Proceedings of the 2nd European Workshop on Planning, Vadstena.* IOS Press, Amsterdam, 1993.

[Car90a] S. Carberry. A model of plan recognition that facilitates default inferences. In *Proceedings of the Second International Workshop on User Modeling (UM90),* Honolulu, Hawaii, 1990.

[Car90b] S. Carberry. *Plan Recognition in Natural Language Dialogue.* MIT Press, Cambridge, MA, 1990.

[CG91] E. Charniak and R. Goldman. A probabilistic model of plan recognition. In *Proceedings of the 9th National Conference of the American Association for Artificial Intelligence,* pages 160-165, Anaheim, CA, July 1991. MIT Press.

[CI89] J. Cheng and K. Irani. Ordering problem subgoals. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence,* 1989.

[FGKK88] J. D. Foley, C. Gibbs, W. C. Kim, and S. Kovacevic. A knowledge-based user interface management system. In *CHI'88 Human Factors in Computer Systems, Conference Proceedings,* Washington, D.C., 1988.

[Fin83] T. W. Finin. Providing help and advice in task oriented systems. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence,* pages 176-178, Karlsruhe, Germany, 1983.

[FLS85] G. Fischer, A. Lemke, and T. Schwab. Knowledge-based help systems. In *Proceedings of the CHI'85 Conference on Human Factors in Computing Systems.* acm Press, 1985.

[FT91] T. Fehrle and M. A. Thies. InCome: A system to navigate through interactions and plans. In H.-J. Bullinger, editor, *Human Aspects in Computing: Design and Use of Interactive Systems and Information Management,* Amsterdam, London, New York, Tokyo, 1991. Elsevier Science Publishers B.V.

[GL92] B. Goodman and D. Litman. On the interaction between plan recognition and intelligent interfaces. *User Modeling and User-Adapted Interaction,* 2(1-2)-83-116, 1992.

[Ham90] K. J. Hammond. Explaining and repairing plans that fail. *Artificial Intelligence,* 45:173-228, 1990.

[HRS90] M. Heisel, W. Reif, and W. Stephan. Tactical theorem proving in program verification. In *Proceedings of the 10th Conference on Automated Deduction,* pages 117-131. Springer LNCS 449, 1990.

[HW92] S. Hanks and D. S. Weld. Systematic adaptation for case-based planning. In *Proceedings of the 1st International Conference on Artificial Intelligence Planning Systems,* pages 96-105, Washington, D.C., 1992. Morgan Kaufmann, Menlo Park.

[KH92] S. Kambhampati and J. A. Hendler. A validation-structure-based theory of plan modification and reuse. *Artificial Intelligence,* 55:193 - 258, 1992.

[Koe92] J. Koehler. Towards a logical treatment of plan reuse. In *Proceedings of the 1st International Conference on Artificial Intelligence Planning Systems,* pages 285-286, Washington, D.C., 1992. Morgan Kaufmann, Menlo Park.

[Koe93] J. Koehler. Flexible Plan Reuse in a Formal Framework. In *Proceedings of the 2nd European Workshop on Planning, Vadstena.* IOS Press, Amsterdam, 1993.

[May92] J. Mayfield. Controlling inference in plan recognition. *User Modeling and User-Adapted Interaction,* 2(l-2):55-82, 1992.

[MW87] Z. Manna and R. Waldinger. How to clear a block: Plan formation in situational logic. *Journal of Automated Reasoning,* 3:343-377, 1987.

[Nei82] D. Neiman. Graphical Animation from Knowledge. In *Proceedings of the 2nd National Conference of the American Association for Artificial Intelligence,* Pittsburgh, PA, 1982. AAAI Press.

[NK93a] B. Nebel and J. Koehler. Plan modification versus plan generation: A complexity-theoretic perspective. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence,* Chambéry, France, 1993. Morgan Kaufmann, Menlo Park.

[NK93b] B. Nebel and J. Koehler. Plan reuse versus plan generation: A theoretical and empirical analysis. DFKI Research Report RR-93-33, German Research Center for Artificial Intelligence, 1993.

[Pau89] H. Paul. Exploratives Agieren in interaktiven EDV-Systemen. In B. Endres-Niggemeyer, T. Herrmann, A. Kobsa, and D. Rösner, editors, *Interaktion und Kommunikation mit dem Computer. Informatik Fachbericht 238.* Springer Verlag, Berlin, 1989.

[Pau90] L. Paulson. Isabelle: The next 700 theorem provers. In P. Odifredi, editor, *Logic and Computer Science.* Academic Press, 1990.

[Pau93] G. Paul. Approaches to abductive reasoning - an overview. *Artificial Intelligence Review,* 7:109-152, 1993.

[Pei58] C.S. Peirce. *Collected Papers of Charles Sanders Peirce (eds. C. Hartshorne et al).* Harvard University Press, 1931-1958.

[Rat87] M. Rathke. Undo/redo - Szenarien und Anforderungen für eine anwendungsneutrale Implementierung. In M. Paul, editor, *GI - 17. Jahrestagung Comuterintegrierter Arbeitsplatz im Büro,* Berlin, Heidelberg, New York, London, Paris, Tokyo, 1987. Springer.

[Rat89] M. Rathke. Erweiterung interaktiver Anwendungen um Undo-Mechanismen. In *Software Ergonomie: Aufgabenorientierte Systemgestaltung und Funktionalität, GI Band 32,* Stuttgart, 1989. Teubner.

[SF90] P. Sukaviriya and J. D. Foley. Coupling a UI Framework with Automatic Generation of Context-Sensitive Animated Help. In *Proceedings of the ACM SIGGRAPH Symposium on User Interface Software (UIST'90),* New York, 1990. ACM SIGGRAPH, acm Press.

[Sha76] G. Shafer. *A Mathematical Theory of Evidence.* Princeton University Press, Princeton, NJ, 1976.

[Shn83] B. Shneiderman. Direct Manipulation: A step beyond programming languages. *IEEE Computer,* 16, 1983.

[Shn87] B. Shneiderman. *Designing the User Interfaces: Strategies for effective Human-Computer Interaction.* Addison Wesley, Massachusetts, 1987.

[TB92] M. A. Thies and F. Berger. Plan-based graphical help in object-oriented user interfaces. In T. Catarci, M. F. Costabile, and S. Levialdi, editors, *Proceedings of the International Workshop AVI'92, Advanced Visual Interfaces,* volume 36 of *World Scientific Series in Computer Science,* Rome, Italy, May 1992. World Scientific.

[Thi90] M. A. Thies. Interaction Control Manager: Ein System zum Navigieren durch Interaktionen und Pläne. Master's thesis, Fakultat Informatik, Universität Stuttgart, 1990.

[Thi93] M. A. Thies. Animated help as a sensible extension of a plan-based help system. In *Proceedings of the 5th International Conference on Human-Computer Interaction, HCI International '93,* Orlando, FL, USA, August 1993.

[Thi94] M. A. Thies. *Planbasierte Hilfeverfahren fiir direkt-manipulative Systeme: Erkennung, Vervollständigung und Visualisierung von Interaktionsplänen.* PhD thesis, Fakultät Informatik der Universität Stuttgart, 1994. in Vorbereitung.

[WCL+88] R. Wilensky, D. N. Chin, M. Luria, J. Martin, J. Mayfield, and D. Wu. The Berkeley UNIX Consultant Project. *Computational Linguistics,* 14:35-84, 1988.

[WHK88] W. Wahlster, M. Hecking, and C. Kemke. SC: Ein intelligentes Hilfesystem fur SINIX. In B. Gollan, W.J. Paul, and A. Schmitt, editors, *Innovative Informations-Infrastrukturen,* pages 81-100. Springer IFB 184, Heidelberg, 1988.

[Yan90] Y. Yang. Current approaches & new guidlelines for undo support design. In H.-J. Bullinger and B. Shackel, editors, *Human-Computer Interaction - INTERACT'90,* North-Holland, 1990. Elsevier Science Publishers B.V.