Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Research Report
RR-92-13

Planbasierte graphische Hilfe in objektorientierten Benutzungsoberflächen

Markus A. Thies, Frank Berger

März 1992

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

Postfach 20 80
D-6750 Kaiserslautern, FRG
Tel.: (+49 631) 205-3211/13
Fax: (+49 631) 205-3210

Stuhlsatzenhausweg 3
D-6600 Saarbrücken 11, FRG
Tel.: (+49 681) 302-5252
Fax: (+49 681) 302-5341

Deutsches Forschungszentrum für Künstliche Intelligenz

The German Research Center for Artificial Intelligence (Deutsches Forschungszentrum für Künstliche Intelligenz, DFKI) with sites in Kaiserslautern und Saarbrücken is a non-profit organization which was founded in 1988. The shareholder companies are Daimler Benz, Fraunhofer Gesellschaft, GMD, IBM, Insiders, Krupp-Atlas, Mannesmann-Kienzle, Philips, Sema Group Systems, Siemens and Siemens-Nixdorf. Research projects conducted at the DFKI are funded by the German Ministry for Research and Technology, by the shareholder companies, or by other industrial contracts.

The DFKI conducts application-oriented basic research in the field of artificial intelligence and other related subfields of computer science. The overall goal is to construct *systems with technical knowledge and common sense* which - by using AI methods - implement a problem solution for a selected application area. Currently, there are the following research areas at the DFKI:

- Intelligent Engineering Systems
- Intelligent User Interfaces
- Intelligent Communication Networks
- Intelligent Cooperative Systems.

The DFKI strives at making its research results available to the scientific community. There exist many contacts to domestic and foreign research institutions, both in academy and industry. The DFKI hosts technology transfer workshops for shareholders and other interested groups in order to inform about the current state of research.

From its beginning, the DFKI has provided an attractive working environment for AI researchers from Germany and from all over the world. The goal is to have a staff of about 100 researchers at the end of the building-up phase.

Prof. Dr. Gerhard Barth
Director

Planbasierte graphische Hilfe in objektorientierten Benutzungsoberflächen

Markus A. Thies, Frank Berger

DFKI-RR-92-13

Das hier veröffentlichte Papier erscheint in K. Kansy und P. Wißkirchen (Hrsg.): Innovative Programmiermethoden für Graphische Systeme, Springer Verlag, 1992.

Planbasierte graphische Hilfe in objektorientierten Benutzungsoberflächen

Markus A. Thies und Frank Berger
Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI)
Stuhlsatzenhausweg 3
W-6600 Saarbrücken 11
{thies,berger}@dfki.uni-sb.de

Kurzfassung

Wir stellen das System PLUS vor, ein planbasiertes graphisches Hilfesystem für Applikationen mit einer objektorientierten Benutzerschnittstelle. Es werden die Hilfekomponente *InCome*$^+$, die *Animationskomponente* und der graphik-orientierte Planeditor *PlanEdit*$^+$ beschrieben. PlanEdit$^+$ ermöglicht den interaktiven Aufbau der hierarchischen Planbasis, die die Grundlage für den Planerkennungsprozeß bildet. Eine zentrale Komponente der graphischen Hilfe in unserem System stellt das Modul InCome$^+$ dar, das den Interaktionskontext des Benutzers visualisiert und darüberhinaus weitere Features wie semantische Undo- und Redo-Möglichkeiten und einen kontext-sensitiven Tutor zur Verfügung stellt. Als wesentliche Erweiterung der graphischen Benutzerunterstützung wird innerhalb von PLUS die Präsentation animierter Hilfe integriert. Es werden Benutzeraktionen simuliert, indem eine Animation über die aktuelle Benutzerschnittstelle gelagert wird. Die Animationssequenz wird im Kontext der aktuell vom Benutzer verfolgten Aufgabe generiert.

Inhalt

1 Einleitung

Zielsetzung des PLUS-Projektes ist die Entwicklung und Implementierung eines planbasierten Hilfesystems unter Verwendung und Umsetzung bestehender Forschungsergebnisse aus verschiedenen Bereichen der Künstlichen Intelligenz. Dabei versteht man unter einem Plan eine Sequenz von Aktionen, die zur Bearbeitung einer Aufgabe und somit zum Erreichen eines bestimmten Zieles abgearbeitet werden müssen. Im Gegensatz zu den bisherigen Hilfesystemen, die meist für Kommando-orientierte Schnittstellen entwickelt wurden (siehe z.B. [6], [7], [18], [17], [2]), arbeitet PLUS in Applikationen, die dem Benutzer graphische Bedienoberflächen zur Verfügung stellen, und deren Interaktion auf dem Prinzip eines benutzergeführten Dialoges mittels direkter Manipulation (vgl. [12], [13]) basiert — sogenannte "Direkt-Manipulative Benutzerschnittstellen" (DMI).

Die Besonderheit solcher DMI-Umgebungen 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 entwikkelnde 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, kontext-abhängig auf die Probleme des Benutzers einzugehen.

Um diesem Anspruch gerecht zu werden, soll unser Hilfesystem verschiedene Formen intelligenter Benutzerunterstützung bereitstellen — *aktive*, *passive*, *implizite* und *kooperative* Hilfe.

Der Prototyp des PLUS Systems wurde in Smalltalk entwickelt. Die Konzeption und das Design des Prototypen orientieren sich an modernen Programmentwurfsmethoden und Konzepten der objektorientierten Softwareentwicklung (siehe u.a. [19], [3]).

Abschnitt 2 gibt einen Überblick über die Architektur des PLUS Systems und die Funktionalität der einzelnen Komponenten. In Abschnitt 3 folgt eine Beschreibung des graphik-orientierten Planeditors PlanEdit$^+$, der eine komfortable, interaktive Eingabe der hierarchischen Planbasis in einer objektorientierten Oberfläche ermöglicht.

Um dem Benutzer eine adäquate Unterstützung bei seiner Arbeit in einer graphischen Bedienoberfläche geben zu können, bietet PLUS seine Hilfe ebenfalls graphisch aufbereitet an. Diese graphische Hilfe wird realisiert durch die Komponente InCome$^+$, die in Abschnitt 4 beschrieben wird. In Abschnitt 5 wird gezeigt, wie animierte Hilfe als sinnvolle Erweiterung graphischer Benutzerunterstützung eingesetzt werden kann.

2 Das System PLUS

Das zentrale Modul eines planbasierten Hilfesystems ist der Planerkenner. Während der Benutzer mit dem Anwendungssystem arbeitet, versucht der Planerkenner, die ausgeführten Aktionen auf Pläne abzubilden, um Annahmen bezüglich der vom Benutzer intendierten Ziele zu treffen. Diese Hypothesen bilden die Grundlage, um dem Benutzer verschiedene Arten von Hilfe anbieten zu können.

Es existieren zwei grundlegend verschiedene Ansätze zur Realisierung planbasierter Systeme. Man kann unterscheiden zwischen Systemen, bei denen zur Laufzeit mittels eines Plangenerierungssystems Pläne erzeugt werden, die als Grundlage für den Planerkenner dienen (vgl. z.B. [2]), und Systemen, dei denen eine vorgefertigte Planbasis die Grundlage für den Erkennungsprozeß bildet. Innerhalb

von PLUS wird der zweite Ansatz verfolgt.

Um eine adäquate Umsetzung der Erfordernisse, die an eine Planerkennung in einer DMI-Umgebung gestellt werden, zu gewährleisten, wird in PLUS ein zweistufiger Planerkennungsprozeß verwendet. Die erste Stufe verarbeitet die low-level-Events wie Mausaktionen und Tastatureingaben. Mit Hilfe dieser ersten Stufe der Planerkennung ist es möglich, bevorzugte Interaktionsstile des Benutzers zu erkennen — verwendet er häufiger die Maus oder bevorzugt er 'short-paths' — und diese in einem einfachen Benutzermodell mitzuprotokollieren. Dieses Benutzermodell kann einerseits dazu verwendet werden, Hilfeinformation an die Gewohnheiten des Benutzers anzupassen, andererseits ist es auch möglich, auf fehlendes Wissen seitens des Benutzers bzgl. alternativer Interaktionskonzepte zu schließen und ihm eine entsprechende Hilfe anzubieten. Darüberhinaus kann diese Ebene beispielsweise auch bei der Generierung von Hilfesequenzen dazu verwendet werden, die effektivste Interaktion zum Ausführen einer bestimmten Aktion zu bestimmen. Dieser zweistufige Ansatz hat den Vorteil, daß die Ebene der Events behandelt wird, ohne den eigentlichen Planerkennungsprozeß zu belasten. Man kann die erste Ebene als eine Art Kurzzeitgedächtnis auffassen, welches nur den momentanen Zustand der aktuellen Ereignisse kennt.

Die Ergebnisse der ersten Planerkennungsstufe sind die vom Benutzer ausgeführten Applikationsspezifischen Aktionen, die etwa als *Items* innerhalb von *Pulldown-Menüs* selektierbar sind. Diese Aktionen werden in einer Dialoghistorie gespeichert, die als Eingabe für die zweite Stufe des Planerkennungsprozesses dient. Diese zweite Stufe wird durch das Modul PlanRecognizer$^+$ realisiert.

PlanRecognizer$^+$ versucht mit Hilfe eines *Spreading Activation* Algorithmus, die in der Dialoghistorie gespeicherten Aktionen auf Pläne in einer vorgefertigten hierarchischen Planbasis abzubilden. 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 mittels des graphik-orientierten Planeditors PlanEdit$^+$ eingegeben (vgl. Abschnitt 3).

Der Spreading Activation Algorithmus baut zur Laufzeit eine sogenannte *dynamische Planbasis* auf. Das in der dynamischen Planbasis enthaltene hypothetische Wissen über die aktuell vom Benutzer verfolgten Pläne und Ziele wiederum dient zusammen mit einer Wissensbasis über allgemeine Hilfe-Konzepte als Grundlage für die Hilfekomponenten und für InCome$^+$ (vgl. Abb. 1, PLUS System-Architekur).

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 Fokussierungs-Techniken:

- Mit Hilfe von *Time Frames* werden die in der dynamischen Planbasis enthaltenen Pläne in verschiedene Klassen unterteilt. Diese Klassifizierung wird beispielsweise dazu benutzt, 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 Objektes, das in der Parameter-Liste eines Planes enthalten ist.

- Pläne können auf bestimmte *Views*[1] beschränkt werden. Wird ein View geschlossen, so können alle mit diesem View assoziierten Planhypothesen verworfen werden.

[1] Views sind typisierte Windows, die eine bestimmte Sichtweise auf Objekte erlauben.

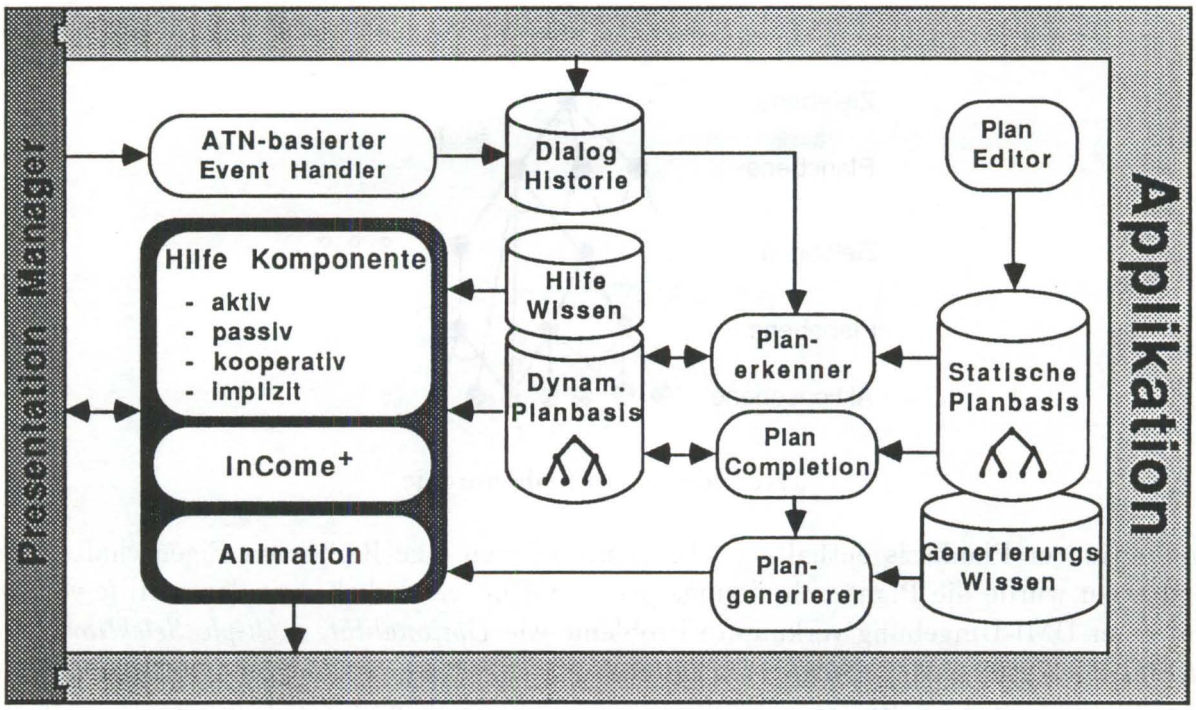

Abbildung 1: PLUS System-Architektur

Durch die Beschränkung der Anzahl der Planhypothesen ist der Einsatz dieser Fokussierungs-Techniken auch ein wesentliches Hilfsmittel zum Erzielen einer annehmbaren Laufzeit, was einen wichtigen Aspekt für die Benutzer-Akzeptanz eines Hilfesystems darstellt.

Die folgenden Abschnitte enthalten eine detailliertere Beschreibung der Komponenten *Planeditor*, *InCome*$^+$ und *Animation*.

3 Der Planeditor

Unsere Planhierarchie umfaßt 3 Typen von Objekten: *Aktionen*, *Pläne* und *Ziele*. Sie ist wie folgt aufgebaut (vgl. Abb. 2):

- Die unterste Ebene besteht aus den *Aktionen*, die der Benutzer bei seiner Arbeit mit der Applikation ausführen kann. Aktionen können Teile von Plänen sein. Diese Beziehung wird repräsentiert durch eine Kante zwischen einem Aktionsknoten und dem entsprechenden Planknoten.

- Ein *Plan* setzt sich zusammen aus einer Menge von Aktionen und/oder Teilzielen. 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.

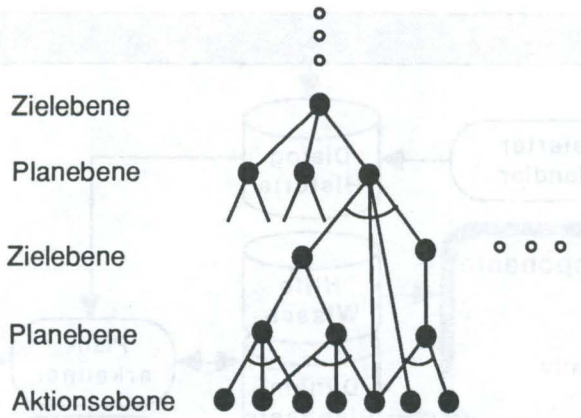

Abbildung 2: Planhierarchie

Für die in der Planbasis enthaltenen Elemente können eine Reihe von Eigenschaften definiert werden. Dazu wurde die Planbeschreibungssprache GPL$^+$ entwickelt, mit deren Hilfe verschiedene eng mit einer DMI-Umgebung verknüpfte Probleme wie *Optionalität, multiple Selektion, Iteration, Parallelität* sowie verschiedene *Sichten* auf Objekte abgebildet werden können. Daneben können allgemein bei Planerkennung auftretende Merkmale wie *Parameter-* und *Zeit-Constraints, Plan-Abbruch* und *Plan-Interaktionen* modelliert werden. GPL$^+$ bietet außerdem die Möglichkeit, die einzelnen Elemente mit Hierarchie-Information zu versehen, um so die Struktur der Planbasis zu definieren.

Aufgrund des Aufbaus unserer Planbasis — eine Objekt-Hierarchie mit spezifischen Eigenschaften pro Objekt — war es naheliegend, einerseits eine objektorientierte interne Implementation der Planbasis zu wählen, und andererseits dem Plandesigner auch eine möglichst komfortable, objektorientierte Eingabemöglichkeit zur Verfügung zu stellen. Zu diesem Zweck wurde der graphik-orientierte Planeditor PlanEdit$^+$ entwickelt, der den interaktiven Aufbau der hierarchischen Planbasis in einer DMI-Oberfläche ermöglicht.

Abb. 3 zeigt das Arbeitsfenster von PlanEdit$^+$, in dem der größte Teil der Interaktion stattfindet. Die Elemente der Planbasis werden als graphische Objekte dargestellt. Jedes Objekt besteht aus einem Icon, das den Typ des Elementes repräsentiert, und dem Namen des Elementes. In der *Type Box* in der unteren linken Ecke des Arbeitsfensters werden Icons für die drei verschiedenen in der Planbasis enthaltenen Elemente zur Verfügung gestellt: Aktionen, Pläne und Ziele. Diese Icons dienen als 'Reservoir' für das Erzeugen neuer Elemente. Die Eigenschaften der Elemente können in einer Reihe von Dialog-Boxen spezifiziert werden. Die Anordnung der Elemente im Arbeitsfenster ist beliebig, d.h. der Anordnung wird keine Bedeutung bezüglich der Struktur der Planbasis beigemessen. Eine graphische Unterstützung zur Visualisierung der Baumstruktur etwa durch die Einbeziehung von Kanten wäre zwar wünschenswert, allerdings lag der Hauptschwerpunkt der Arbeit im PLUS Projekt in der Entwicklung des Hilfesystems, so daß aufgrund der begrenzten personellen Resourcen bisher keine Weiterentwicklung der graphischen Aufbereitung der Planbasis erfolgen konnte.

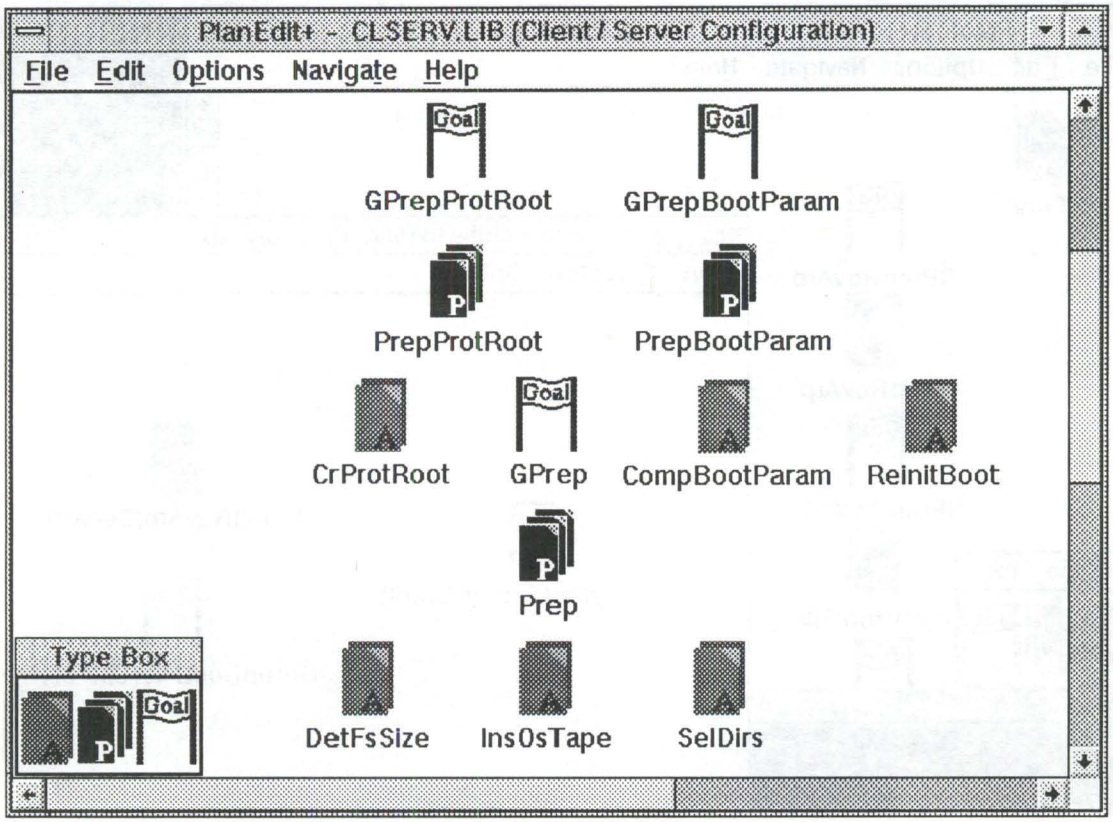

Abbildung 3: PlanEdit$^+$ Arbeitsfenster

Da der Inhalt des Arbeitsfensters bei einer wachsenden Planbasis sehr schnell unübersichtlich wird, wurde ein zweiter Fenster-Typ hinzugefügt, der es ermöglicht, sich die Struktur bereits definierter Pläne und Ziele separat anzusehen und Modifizierungen vorzunehmen (vgl. Abb. 4, Beispiel eines Plan-Fensters).

Im Gegensatz zum Arbeitsfenster entspricht die Anordnung der Objekte innerhalb eines Plan-Fensters der durch die zeitlichen Constraints definierten logischen Abfolge der entsprechenden Elemente innerhalb des Planes. Die Elemente sind in chronologischer Folge von links nach rechts angeordnet. Für Elemente, die in einer Spalte angeordnet sind, sind untereinander keine zeitlichen Constraints spezifiziert.

Nach der vollständigen Spezifikation der Planbasis mit PlanEdit$^+$ wird ein entsprechendes Smalltalk-Modul zur Modellierung der internen, objektorientierten Repräsentation der statischen Planbasis erzeugt. Dieses Modul wird vom Planerkenner zur Laufzeit geladen.

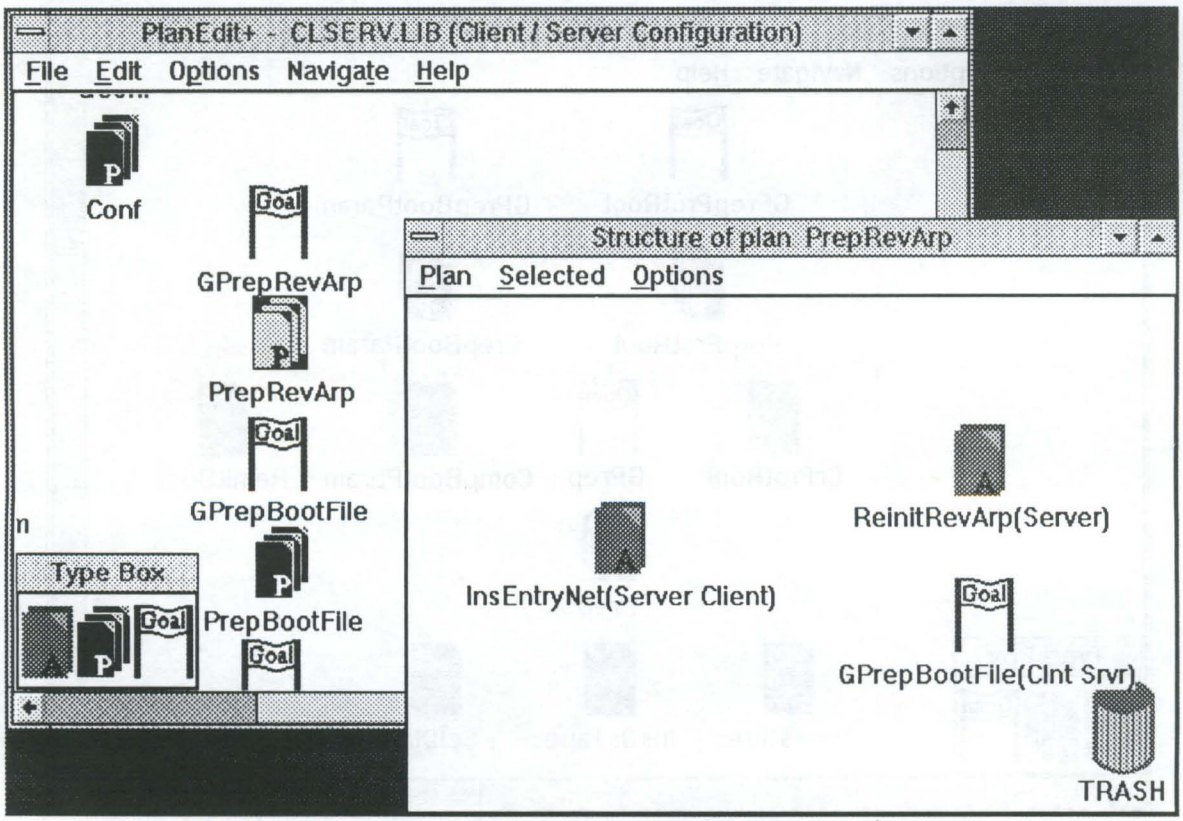

Abbildung 4: Verschiedene Fenstertypen

4 Die Komponente InCome$^+$

Eine zentrale Komponente der graphischen Hilfe im PLUS System ist der *Interaction Control Manager* InCome$^+$ (vgl. [15], [5]). Er erzeugt eine graphische Visualisierung des aktuellen Interaktionskontextes, 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 (vgl. [9]), 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. Die Planvervollständigungskomponente generiert auf Anfrage gültige Aktionssequenzen für Planhypothesen, die in der dynamischen Planbasis enthalten sind. Dabei

werden zeitliche Constraints erfüllt und anhand der Parameter-Constraints bekannte Parameterwerte propagiert.

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. Um die Sequenz von Aktionen eines Plans widerzuspiegeln, werden die Knoten eines Plans mit Pfeilen verbunden (siehe Abb. 5).

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 unterschiedlicher Farbe visualisiert.

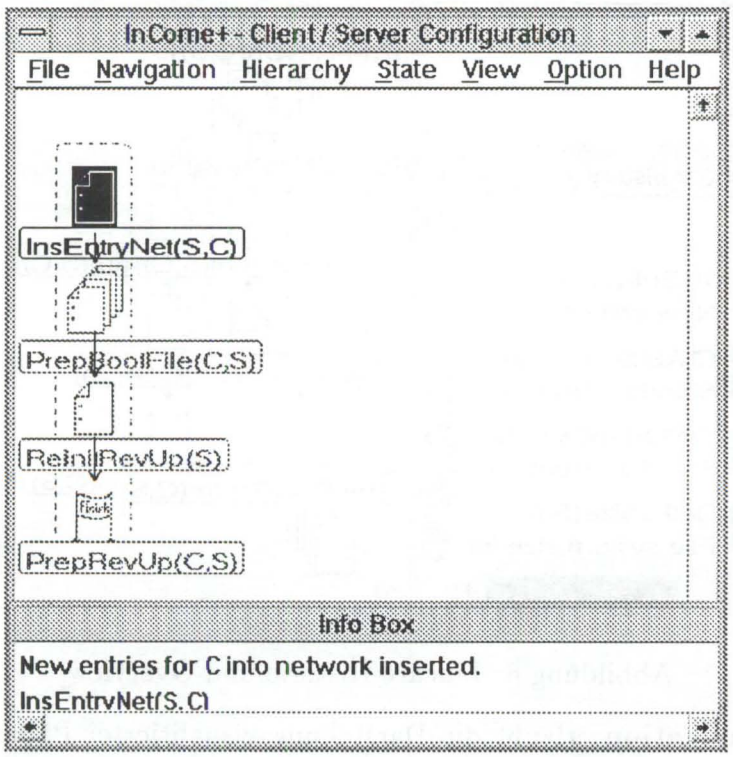

Abbildung 5: Dargestellte Elemente in InCome$^+$

In Abb. 5 sind die drei modellierten Objektklassen *Aktion*, *Plan* und *Ziel* sichtbar. 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.

InCome$^+$ präsentiert sich in einem eigenen Fenster. Die dargestellten Knoten sind maus-sensitiv. Die InCome$^+$-spezifischen Aktionen werden in *Pull-down Menus* zusammengefaßt.
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*, *overview* und *suchen* von Knoten nach bestimmten Suchkriterien. Das Overview Fenster enthält den gesamten visualisierten Graphen, der graphisch abstrahiert und in einer reduzierten Größe dargestellt wird. Innerhalb des Overview Fensters werden weitere navigatorische Aktionen angeboten wie indirektes und direktes Positionieren des

InCome+ Fensters. 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. In Abb. 6 ist im linken unteren Fenster die lineare Dialoghistorie sichtbar. Die ikonifiziert dargestellten und nach oben gerichteten Pfeile markieren *reversible Aktionen.*

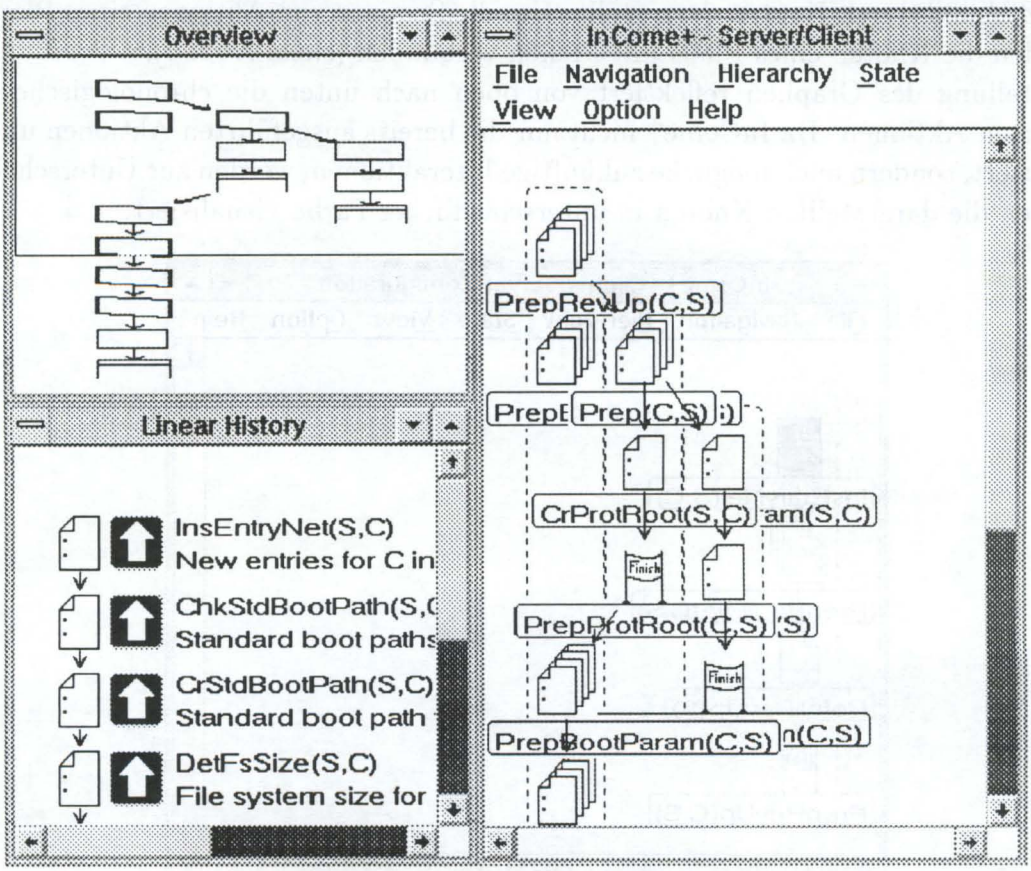

Abbildung 6: Lineare Historie und Overview

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, die die Planinteraktionen reflektiert (vgl. Abb. 7 und Abb. 8). Es werden Aktionen zum *Expandieren* und zum *Abstrahieren* von Plänen angeboten. In Abb. 7 überlappen sich die beiden Pläne *PrepBootParam(C,S)* und *PrepProtRoot(S,C)* mit dem Plan *Prep(C,S).*

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 die optimale 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 (vgl. Abb. 9). Von jetzt an beobachtet InCome+ die Interaktionsschritte des Benutzers und vergleicht diese mit der *To-Do-List.* Die anschließend 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 (vgl. Abb. 10).

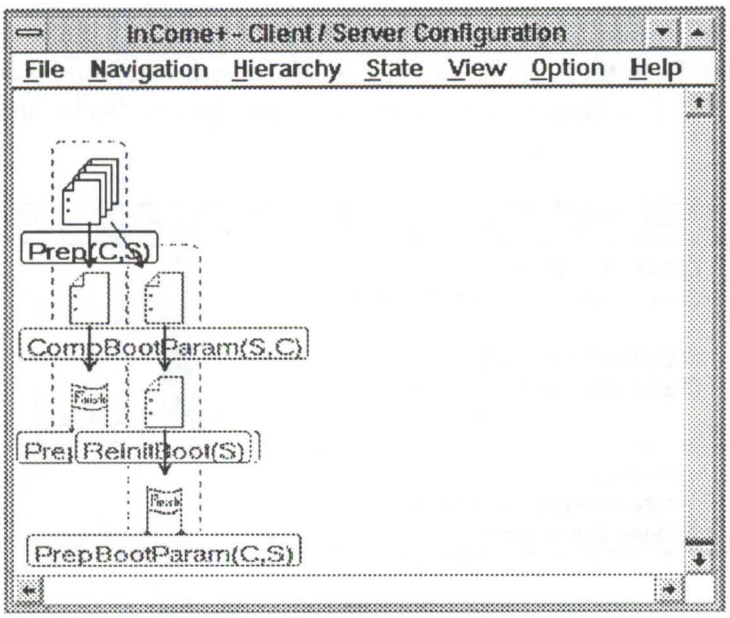

Abbildung 7: Überlappende Darstellung

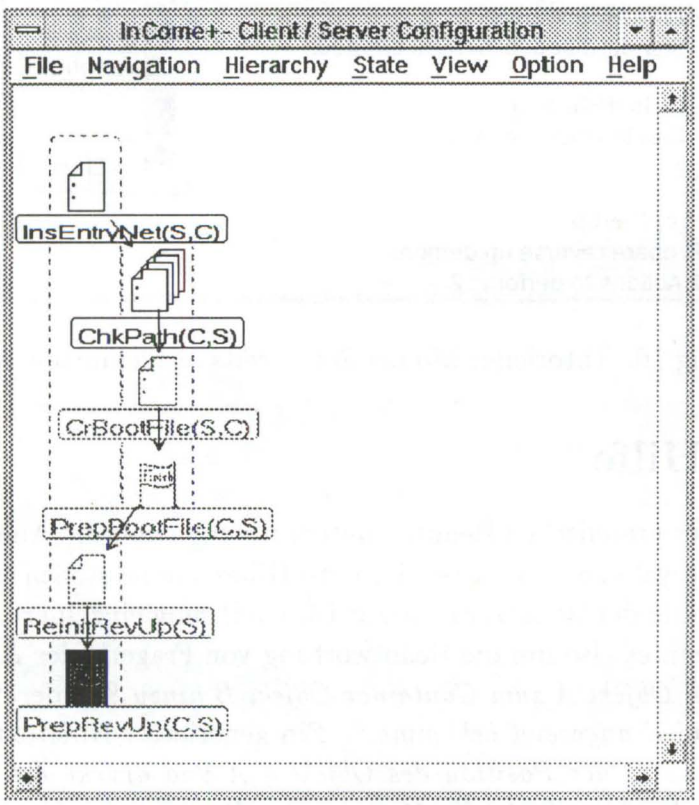

Abbildung 8: Planeinbindung

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.

InCome$^+$ bietet eine Schnittstelle zu den Undo-Mechanismen der Applikation an. Um zwei verschiedene Undo Prinzipien (funktionsorientiert vs. zustandsorientiert; siehe u.a. [10], [11], [20]) behandeln zu können, arbeitet InCome$^+$ mit einem erweiterten funktionsorientierten Ansatz unter Verwendung von *freezing-points* (vgl. [9]), 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* (siehe u.a. [10]) ist im jetzigen System nicht vorgesehen.

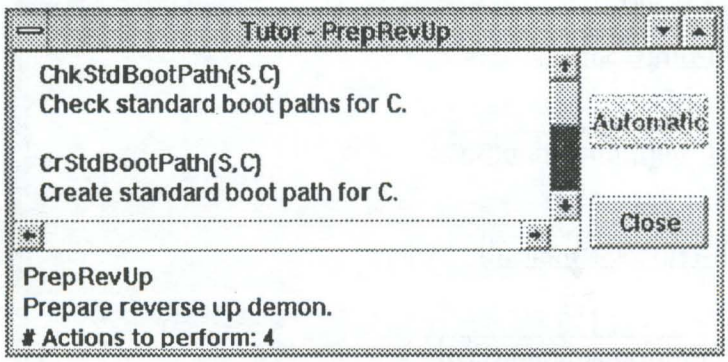

Abbildung 9: Tutorieller Modus

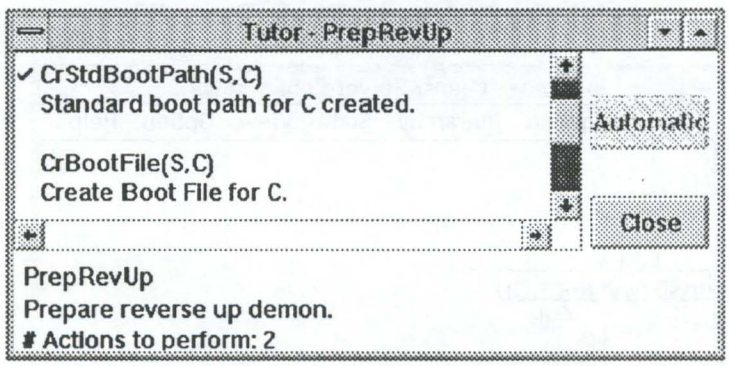

Abbildung 10: Tutorieller Modus mit bereits ausgeführten Aktionen

5 Animierte Hilfe

Einen weiteren Aspekt der graphischen Benutzerunterstützung stellt die Anbindung animierter Hilfe dar. Konventionelle Hilfesysteme und wissensbasierte Hilfesysteme stoßen mit einer rein textuellen Hilfe an ihre Grenzen, sobald der Benutzer Hilfe zur Durchführung einzelner Interaktionsschritte einer Applikation benötigt, wenn es also um die Beantwortung von Fragen oder Aufforderungen folgender Form geht: *"Wie füge ich Objekt A zum Container-Objekt B 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 des Objektes A und drücke die linke Maustaste nieder. Bewege nun mit niedergedrückter Taste die Maus zu der Position des Container-Objektes B. Lasse die Maustaste wieder los."*. Es wird deutlich, daß eine animierte Sequenz der Interaktionsschritte dem Benutzer eine adäquatere Unterstützung bietet.

Innerhalb des PLUS-Projektes wird eine Animationskomponente entwickelt, die, im Gegensatz zur animierten Hilfe in den Systemen GAK (Graphical Animation from Knowledge, vgl. [8]) und Cartoonist (siehe [14]), einen stärkeren Bezug zur momentan vom Benutzer verfolgten Aufgabe erreicht. Durch die Anbindung der Animationskomponente an den Planerkenner und die Planvervollständigungskomponente, kann die Animationskomponente auf Anfrage gezielt für eine Planhypothese eine Sequenz von Animationsschritten generieren.

Die Generierung der Animationsschritte erfolgt in zwei Phasen. In der ersten Phase komplettiert die Planvervollständigungskomponente eine Planhypothese durch Generieren einer Aktionssequenz. Diese Aktionssequenz dient der Animationskomponente als Grundlage für die inkrementelle deduktive

Generierung der Animationsschritte (zweite Phase). Die Animationskomponente greift hierbei auf die in der Wissensbasis für jede Aktion definierten Vor- und Nachbedingungen zu. Eine inkrementelle Generierung ist nötig, um jeweils den neuen Bildschirmkontext berücksichtigen zu können.

Die Durchführung der Animationsschritte erfolgt durch imitierte Mauseingaben, die von der Animationskomponente erzeugt werden. Die so simulierten Mausaktionen werden der Benutzungsoberfläche so zugeführt, daß diese reagiert, als würden die Eingaben vom Benutzer stammen.

Die Wissensbasis ist in die Teile *aktionsspezifisches*, *animationsspezifisches*, *generisches* und *interfacespezifisches* Wissen untergliedert. Die Wissensbasis dient als eine Erweiterung der statischen Planbasis und ist nur durch die Teile des *aktionsspezifischen* und *interfacespezifischen* Wissens applikationsabhängig.

Durch die Repräsentation von generischen Interface-Konzepten in der Wissensbasis, können auch navigatorische Animationsschritte generiert werden. So werden z.B. Animationssequenzen zum Verschieben des sichtbaren Bereiches eines Fensters oder zum Hinzufügen von Objekten, die in der aktuellen Darstellung ausgeblendet sind, in die momentane Visualisierung generiert.

6 Implementation

Der Prototyp des PLUS-Systems wurde in Smalltalk/V PM unter dem Betriebssystem OS/2 auf einem IBM PS/2 mit 8MB Hauptspeicher entwickelt. Die Konzeption und das Design des Prototypen orientiert sich an modernen Programmentwurfsmethoden und -konzepten (siehe u.a. [19], [3]). Dabei konnte die Struktur der objektorientierten Benutzungsoberfläche (Presentation Manager) des OS/2 Betriebssystems dank der sehr guten Einbettung des Smalltalk/V Systems ausgenutzt werden.

7 Zusammenfassung und Ausblick

In diesem Papier haben wir gezeigt, wie graphische Hilfe als adäquates Mittel zur Benutzerunterstützung eingesetzt werden kann. Dabei zeigt die Integration von Animation in ein Hilfesystem eine interessante Perspektive auf. Zusätzlich zur animierten Darstellung der Interaktionsschritte könnte eine automatisch generierte Animationssequenz zur Verdeutlichung der Handhabung des Eingabemediums erfolgen.

Um der Anforderung einer *'narrated animation'* (siehe [1]) gerecht zu werden, ist an eine natürlichsprachliche Ausgabe gedacht, die zusätzlich die Animation kommentiert. Dabei sollten die Kommentare auch einen erklärenden Charakter besitzen (siehe u.a. in [16], [4]).

8 Danksagung

Die in diesem Papier vorgestellten Forschungsarbeiten wurden innerhalb des Kooperationsprojektes PLUS (**PL**an-based **U**ser **S**upport) zwischen dem IBM Labor Böblingen, der IBM Deutschland GmbH und dem DFKI durchgeführt. Die Laufzeit von PLUS beträgt 2 Jahre (1.10.1990 - 30.9.1992). Folgende Wissenschaftler sind im PLUS Projekt involviert: Prof. Dr. Wolfgang Wahlster (DFKI), Frank Berger (DFKI), Markus A. Thies (DFKI), Dr. Thomas Fehrle (IBM Labor) und Volker Schölles (IBM Labor).
Wir danken Wolfgang Wahlster und Thomas Fehrle für wertvolle Anregungen zu dieser Arbeit.

9 Literaturverzeichnis

[1] N. I. Badler, B. A. Barsky, and D. Zeltzer. *Making Them Move: Mechanics, Control, and Animation of articulated Figures.* Morgan Kaufmann Publishers, Inc, San Mateo, California, 1991.

[2] M. Bauer, S. Biundo, D. Dengler, M. Hecking, J. Köhler, and G. Merziger. Integrated Plan Generation and Recognition - A Logic-Based Approach. In W. Brauer and D. Hernández, editors, *Verteilte Künstliche Intelligenz und kooperatives Arbeiten. 4. Internationaler GI-Kongress Wissensbasierte Systeme*, Berlin, Heidelberg, 1991. Springer. Also DFKI Research Report RR-91-26.

[3] Grady Booch. *Object Oriented Design with Applications.* The Benjamin/Cummings Publishing Company, Redwood City, California, USA, 1991.

[4] Th. Fehrle. *Menüorientierte, wissensbasierte Klärungsdialoge für ein natürlichsprachliches Auskunftssystem.* PhD thesis, Institut für Informatik der Universität Stuttgart, 1989.

[5] Th. 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.

[6] T. W. Finin. Providing help and advice in task oriented systems. In *Proc. IJCAI-83*, pages 176–178, Karlsruhe, Deutschland, 1983.

[7] G. Fischer, A. Lemke, and T. Schwab. Knowledge-based help systems. In *Proceedings CHI-85*, San Francisco, CA, 1985.

[8] D. Neiman. Graphical Animation from Knowledge. In *Proceedings of AAAI*, 1982.

[9] 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.

[10] M. Rathke. Undo/redo - szenarien und anforderungen für eine anwendungsneutrale implementierung. In M. Paul, editor, *GI - 17. Jahrestagung Computerintegrierter Arbeitsplatz im Büro*, Berlin, Heidelberg, New York, London, Paris, Tokyo, 1987. Springer.

[11] M. Rathke. Erweiterung interaktiver anwendungen um undo-mechanismen. In *Software Ergonomie: Aufgabenorientierte Systemgestaltung und Funktionalität, GI Band 32*, Stuttgart, 1989. Teubner.

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

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

[14] P. Sukaviriya and J. D. Foley. Coupling a ui framework with automatic generation of context-sensitive animated help. In *Proceedings of ACM SIGGRAPH 1990 Symposium on User Interface Software and Technology (UIST'90)*, pages 152–166, Snowbird, Utah, October 1990.

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

[16] W. Wahlster. *Natürlichsprachliche Argumentation in Dialogschnittstellen.* Informatik Fachberichte 48. Springer-Verlag, 1981.

[17] W. Wahlster, D. Dengler, M. Hecking, and C. Kemke. SC: The SINIX consultant. In P. Norvig, W. Wahlster, and R. Wilensky, editors, *Intelligent Help Systems for Unix - Case Studies in Artificial Intelligence.* Springer, Heidelberg, 1990.

[18] 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.

[19] P. Wisskirchen. *Object-Oriented Graphics. From GKS and PHIGS to Object-Oriented Systems.* Symbolic Computation. Springer-Verlag, Berlin, Heidelberg, 1990.

[20] 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.

Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

DFKI
-Bibliothek-
PF 2080
D-6750 Kaiserslautern
FRG

DFKI Publikationen

Die folgenden DFKI Veröffentlichungen sowie die aktuelle Liste von allen bisher erschienenen Publikationen können von der oben angegebenen Adresse bezogen werden.
Die Berichte werden, wenn nicht anders gekennzeichnet, kostenlos abgegeben.

DFKI Publications

The following DFKI publications or the list of all publisched papers so far can be ordered from the above address.
The reports are distributed free of charge except if otherwise indicated.

DFKI Research Reports

RR-91-08
Wolfgang Wahlster, Elisabeth André, Som Bandyopadhyay, Winfried Graf, Thomas Rist: WIP: The Coordinated Generation of Multimodal Presentations from a Common Representation
23 pages

RR-91-09
Hans-Jürgen Bürckert, Jürgen Müller, Achim Schupeta: RATMAN and its Relation to Other Multi-Agent Testbeds
31 pages

RR-91-10
Franz Baader, Philipp Hanschke: A Scheme for Integrating Concrete Domains into Concept Languages
31 pages

RR-91-11
Bernhard Nebel: Belief Revision and Default Reasoning: Syntax-Based Approaches
37 pages

RR-91-12
J.Mark Gawron, John Nerbonne, Stanley Peters: The Absorption Principle and E-Type Anaphora
33 pages

RR-91-13
Gert Smolka: Residuation and Guarded Rules for Constraint Logic Programming
17 pages

RR-91-14
Peter Breuer, Jürgen Müller: A Two Level Representation for Spatial Relations, Part I
27 pages

RR-91-15
Bernhard Nebel, Gert Smolka: Attributive Description Formalisms ... and the Rest of the World
20 pages

RR-91-16
Stephan Busemann: Using Pattern-Action Rules for the Generation of GPSG Structures from Separate Semantic Representations
18 pages

RR-91-17
Andreas Dengel, Nelson M. Mattos: The Use of Abstraction Concepts for Representing and Structuring Documents
17 pages

RR-91-18
John Nerbonne, Klaus Netter, Abdel Kader Diagne, Ludwig Dickmann, Judith Klein: A Diagnostic Tool for German Syntax
20 pages

RR-91-19
Munindar P. Singh: On the Commitments and Precommitments of Limited Agents
15 pages

RR-91-20
Christoph Klauck, Ansgar Bernardi, Ralf Legleitner FEAT-Rep: Representing Features in CAD/CAM
48 pages

RR-91-21
Klaus Netter: Clause Union and Verb Raising Phenomena in German
38 pages

RR-91-22
Andreas Dengel: Self-Adapting Structuring and Representation of Space
27 pages

RR-91-23
Michael Richter, Ansgar Bernardi, Christoph Klauck, Ralf Legleitner: Akquisition und Repräsentation von technischem Wissen für Planungsaufgaben im Bereich der Fertigungstechnik
24 Seiten

RR-91-24
Jochen Heinsohn: A Hybrid Approach for Modeling Uncertainty in Terminological Logics
22 pages

RR-91-25
Karin Harbusch, Wolfgang Finkler, Anne Schauder: Incremental Syntax Generation with Tree Adjoining Grammars
16 pages

RR-91-26
M. Bauer, S. Biundo, D. Dengler, M. Hecking, J. Koehler, G. Merziger: Integrated Plan Generation and Recognition - A Logic-Based Approach -
17 pages

RR-91-27
A. Bernardi, H. Boley, Ph. Hanschke, K. Hinkelmann, Ch. Klauck, O. Kühn, R. Legleitner, M. Meyer, M. M. Richter, F. Schmalhofer, G. Schmidt, W. Sommer: ARC-TEC: Acquisition, Representation and Compilation of Technical Knowledge
18 pages

RR-91-28
Rolf Backofen, Harald Trost, Hans Uszkoreit: Linking Typed Feature Formalisms and Terminological Knowledge Representation Languages in Natural Language Front-Ends
11 pages

RR-91-29
Hans Uszkoreit: Strategies for Adding Control Information to Declarative Grammars
17 pages

RR-91-30
Dan Flickinger, John Nerbonne: Inheritance and Complementation: A Case Study of *Easy* Adjectives and Related Nouns
39 pages

RR-91-31
H.-U. Krieger, J. Nerbonne: Feature-Based Inheritance Networks for Computational Lexicons
11 pages

RR-91-32
Rolf Backofen, Lutz Euler, Günther Görz: Towards the Integration of Functions, Relations and Types in an AI Programming Language
14 pages

RR-91-33
Franz Baader, Klaus Schulz: Unification in the Union of Disjoint Equational Theories: Combining Decision Procedures
33 pages

RR-91-34
Bernhard Nebel, Christer Bäckström: On the Computational Complexity of Temporal Projection and some related Problems
35 pages

RR-91-35
Winfried Graf, Wolfgang Maaß: Constraint-basierte Verarbeitung graphischen Wissens
14 Seiten

RR-92-01
Werner Nutt: Unification in Monoidal Theories is Solving Linear Equations over Semirings
57 pages

RR-92-02
Andreas Dengel, Rainer Bleisinger, Rainer Hoch, Frank Hönes, Frank Fein, Michael Malburg: Π_{ODA}: The Paper Interface to ODA
53 pages

RR-92-03
Harold Boley: Extended Logic-plus-Functional Programming
28 pages

RR-92-04
John Nerbonne: Feature-Based Lexicons: An Example and a Comparison to DATR
15 pages

RR-92-05
Ansgar Bernardi, Christoph Klauck, Ralf Legleitner, Michael Schulte, Rainer Stark: Feature based Integration of CAD and CAPP
19 pages

RR-92-07
Michael Beetz: Decision-theoretic Transformational Planning
22 pages

RR-92-08
Gabriele Merziger: Approaches to Abductive Reasoning - An Overview -
46 pages

RR-92-09
Winfried Graf, Markus A. Thies: Perspektiven zur Kombination von automatischem Animationsdesign und planbasierter Hilfe
15 Seiten

RR-92-11
Susane Biundo, Dietmar Dengler, Jana Koehler:
Deductive Planning and Plan Reuse in a Command Language Environment
13 pages

RR-92-13
Markus A. Thies, Frank Berger:
Planbasierte graphische Hilfe in objektorientierten Benutzungsoberflächen
13 Seiten

RR-92-14
Intelligent User Support in Graphical User Interfaces:

1. InCome: A System to Navigate through Interactions and Plans
   Thomas Fehrle, Markus A. Thies
2. Plan-Based Graphical Help in Object-Oriented User Interfaces
   Markus A. Thies, Frank Berger

22 pages

RR-92-15
Winfried Graf: Constraint-Based Graphical Layout of Multimodal Presentations
23 pages

RR-92-17
Hassan Aït-Kaci, Andreas Podelski, Gert Smolka:
A Feature-based Constraint System for Logic Programming with Entailment
23 pages

RR-92-18
John Nerbonne: Constraint-Based Semantics
21 pages

DFKI Technical Memos

TM-91-01
Jana Köhler: Approaches to the Reuse of Plan Schemata in Planning Formalisms
52 pages

TM-91-02
Knut Hinkelmann: Bidirectional Reasoning of Horn Clause Programs: Transformation and Compilation
20 pages

TM-91-03
Otto Kühn, Marc Linster, Gabriele Schmidt:
Clamping, COKAM, KADS, and OMOS:
The Construction and Operationalization of a KADS Conceptual Model
20 pages

TM-91-04
Harold Boley (Ed.):
A sampler of Relational/Functional Definitions
12 pages

TM-91-05
Jay C. Weber, Andreas Dengel , Rainer Bleisinger:
Theoretical Consideration of Goal Recognition Aspects for Understanding Information in Business Letters
10 pages

TM-91-06
Johannes Stein: Aspects of Cooperating Agents
22 pages

TM-91-08
Munindar P. Singh: Social and Psychological Commitments in Multiagent Systems
11 pages

TM-91-09
Munindar P. Singh: On the Semantics of Protocols Among Distributed Intelligent Agents
18 pages

TM-91-10
Béla Buschauer, Peter Poller, Anne Schauder, Karin Harbusch: Tree Adjoining Grammars mit Unifikation
149 pages

TM-91-11
Peter Wazinski: Generating Spatial Descriptions for Cross-modal References
21 pages

TM-91-12
Klaus Becker, Christoph Klauck, Johannes Schwagereit: FEAT-PATR: Eine Erweiterung des D-PATR zur Feature-Erkennung in CAD/CAM
33 Seiten

TM-91-13
Knut Hinkelmann:
Forward Logic Evaluation: Developing a Compiler from a Partially Evaluated Meta Interpreter
16 pages

TM-91-14
Rainer Bleisinger, Rainer Hoch, Andreas Dengel:
ODA-based modeling for document analysis
14 pages

TM-91-15
Stefan Bussmann: Prototypical Concept Formation
An Alternative Approach to Knowledge Representation
28 pages

TM-92-01
Lijuan Zhang:
Entwurf und Implementierung eines Compilers zur Transformation von Werkstückrepräsentationen
34 Seiten

DFKI Documents

D-91-01
Werner Stein , Michael Sintek: Relfun/X - An Experimental Prolog Implementation of Relfun
48 pages

D-91-02
Jörg P. Müller: Design and Implementation of a Finite Domain Constraint Logic Programming System based on PROLOG with Coroutining
127 pages

D-91-03
Harold Boley, Klaus Elsbernd, Hans-Günther Hein, Thomas Krause: RFM Manual: Compiling RELFUN into the Relational/Functional Machine
43 pages

D-91-04
DFKI Wissenschaftlich-Technischer Jahresbericht 1990
93 Seiten

D-91-06
Gerd Kamp: Entwurf, vergleichende Beschreibung und Integration eines Arbeitsplanerstellungssystems für Drehteile
130 Seiten

D-91-07
Ansgar Bernardi, Christoph Klauck, Ralf Legleitner
TEC-REP: Repräsentation von Geometrie- und Technologieinformationen
70 Seiten

D-91-08
Thomas Krause: Globale Datenflußanalyse und horizontale Compilation der relational-funktionalen Sprache RELFUN
137 Seiten

D-91-09
David Powers, Lary Reeker (Eds.):
Proceedings MLNLO'91 - Machine Learning of Natural Language and Ontology
211 pages
Note: This document is available only for a nominal charge of 25 DM (or 15 US-$).

D-91-10
Donald R. Steiner, Jürgen Müller (Eds.):
MAAMAW'91: Pre-Proceedings of the 3rd European Workshop on „Modeling Autonomous Agents and Multi-Agent Worlds"
246 pages
Note: This document is available only for a nominal charge of 25 DM (or 15 US-$).

D-91-11
*Thilo C. Horstmann:*Distributed Truth Maintenance
61 pages

D-91-12
Bernd Bachmann:
$H^{iera}C_{on}$ - a Knowledge Representation System with Typed Hierarchies and Constraints
75 pages

D-91-13
International Workshop on Terminological Logics
Organizers: Bernhard Nebel, Christof Peltason, Kai von Luck
131 pages

D-91-14
Erich Achilles, Bernhard Hollunder, Armin Laux, Jörg-Peter Mohren: KRIS : Knowledge Representation and Inference System
- Benutzerhandbuch -
28 Seiten

D-91-15
Harold Boley, Philipp Hanschke, Martin Harm, Knut Hinkelmann, Thomas Labisch, Manfred Meyer, Jörg Müller, Thomas Oltzen, Michael Sintek, Werner Stein, Frank Steinle:
µCAD2NC: A Declarative Lathe-Worplanning Model Transforming CAD-like Geometries into Abstract NC Programs
100 pages

D-91-16
Jörg Thoben, Franz Schmalhofer, Thomas Reinartz:
Wiederholungs-, Varianten- und Neuplanung bei der Fertigung rotationssymmetrischer Drehteile
134 Seiten

D-91-17
Andreas Becker:
Analyse der Planungsverfahren der KI im Hinblick auf ihre Eignung für die Abeitsplanung
86 Seiten

D-91-18
Thomas Reinartz: Definition von Problemklassen im Maschinenbau als eine Begriffsbildungsaufgabe
107 Seiten

D-91-19
Peter Wazinski: Objektlokalisation in graphischen Darstellungen
110 Seiten

Planbasierte graphische Hilfe in objektorientierten Benutzungsoberflächen

Markus A. Thies, Frank Berger

RR-92-13
Research Report