Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

**Research Report**
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**

**March 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

# 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**

DFKI-RR-92-14

The paper 'InCome: A System to Navigate through Interactions and Plans' is published in H.-J. Bullinger: Human Aspects in Computing: Design and Use of Interactive Systems and Information Management, Elsevier Science Publishers B.V., 1991

# InCome : A System to Navigate through Interactions and Plans

Th. Fehrle
IBM Laboratory Böblingen
Schönaicherstr. 220
W-7030 Böblingen
Germany

M. A. Thies
German Research Center for AI (DFKI)
Stuhlsatzenhausweg 3
W-6600 Saarbrücken
Germany
thies@dfki.uni-sb.de

**Abstract**

This paper presents a frontend to an intelligent help system based on plans called InCome (**In**teraction **Co**ntrol **M**anager**e**). It visualizes user actions previously executed in a specific application as a graph structure and enables the user to navigate through this structure. A higher level of abstraction on performed user actions shows the dialog history, the interaction context and reachable goals. Finally, the user is able to act on the application via InCome by performing undo mechanisms as well as specifying user goals infered already by the help system.

# Contents

# 1 Introduction

Some effort in the research area of intelligent user support has led to plan- based help systems (cf. [6], [7], [1]), that deduce reachable user goals from interactions performed by a user within an application. The fundamental knowledge of such intelligent help systems is represented by plans describing sequences of actions, that have to be executed to reach a specific goal. Based on that technique intelligent help systems can answer user questions concerning the dialog context as well as suggest ways to fulfil a goal. Currently implemented plan-based help systems support the user when interacting with command oriented operating systems. InCome expands the scope of such help systems to window oriented interaction styles and Direct Manipulation Interfaces (DMI). Many applications offer a comfortable interface but show a deep complexity of implemented objects and actions without offering a suitable user assistance. To provide a sophisticated user support (to inform the user how to proceed in the actual task or how to resume a suspended dialog while working on different tasks in parallel) new approaches are required. To support a user in using a DMI, intelligent help systems must have knowledge about actions, plans and goals as well as the actual state of the interaction context between the user and the application. A sequence of actions must be deducible to reach intended goals of the user. Help information should be easily accessible without changing interaction style.

# 2 Concepts of InCome

InCome[1] provides a graphical visualization of the actual dialog context, the dialog history and possible future interactions. InCome gives the user a quick and helpful reminder on the system state to resume suspended applications. It supports the user in leaving system states unfamiliar to him and in exploring actions which would be executable next. InCome meets the following demands (cf. [5]):

- Adequate visualization of user interactions.
- Display of different levels of abstraction selected by user interactions.
- Graphical navigation services.
- Visualization of possible future interactions.
- Semantic undo/redo capabilities.
- Display of embedded, overlapped and interrupted plans.

An application system using InCome consists of at least the logical part of the application, a presentation manager for the communication between user and application, a plan recognizer/generator knowing predefined hierarchical plans and InCome itself (cf. fig. 1, InCome modules are shaded).

[1]InCome is developed as part of the joint project PLUS (Plan-based User Support) between the IBM Laboratory Böblingen, the IBM Deutschland GmbH and the DFKI

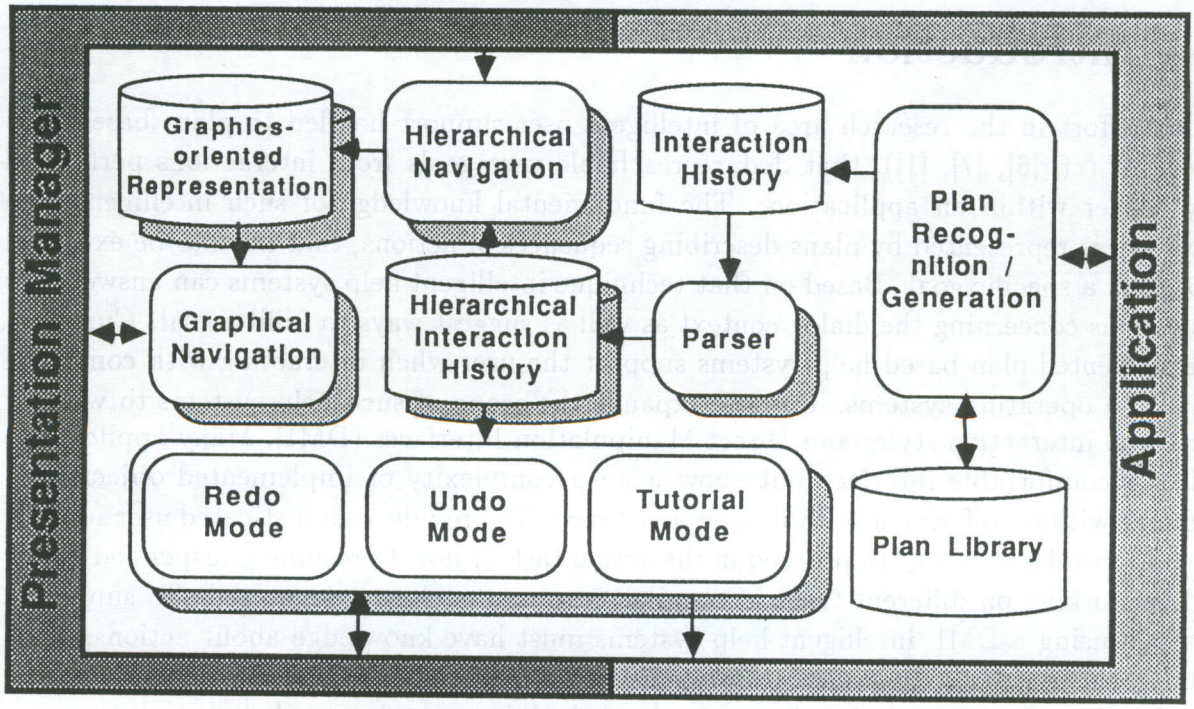


Figure 1: Architecture of InCome

# 3 User Interface of InCome

While the user interacts with the application the plan recognizer tries to map these actions to plans and makes assumptions about goals intended by the user. These assumptions are sent to InCome which builds up an internal representation of them and displays it as a graph structure on the screen. The instances of the two object classes plans and actions are visualized as nodes. To reflect the sequence of actions in a plan the objects belonging to the same plan are connected with arcs. A goal is visualized by a goal-banner. The displayed interaction structure involves different levels of the plan hierarchy. So the user can easily recognize which actions lead to a specific goal.

The overall structure resembles a directed graph from top to bottom to model the chronological order of the performed interactions. Additionally different colors are used for the nodes to distinguish interactions in the past and in the future.

The positioning of the different objects on the screen is performed by an incremental deterministic layout algorithm. This guarantees that the relative positions of the already visualized performed interactions are not influenced by adding new interactions to the graphical representation. This minimizes the cognitive stress of the user to recognize visualized interaction structures.

In DMI there is a high degree of plan interactions such as *plan-overlapping*, *plan-interruption*, and *plan-embedding*. For example, plan-embedding occurs in a plan hierarchy where frequently used sequences of actions are extracted and combined to a plan which can be later used as a subplan in higher-level plans, as for example when performing generic operations, such as choosing a menu selection or selecting a graphical object with the mouse.

InCome runs in its own window. The presented nodes are selectable via mouse clicks

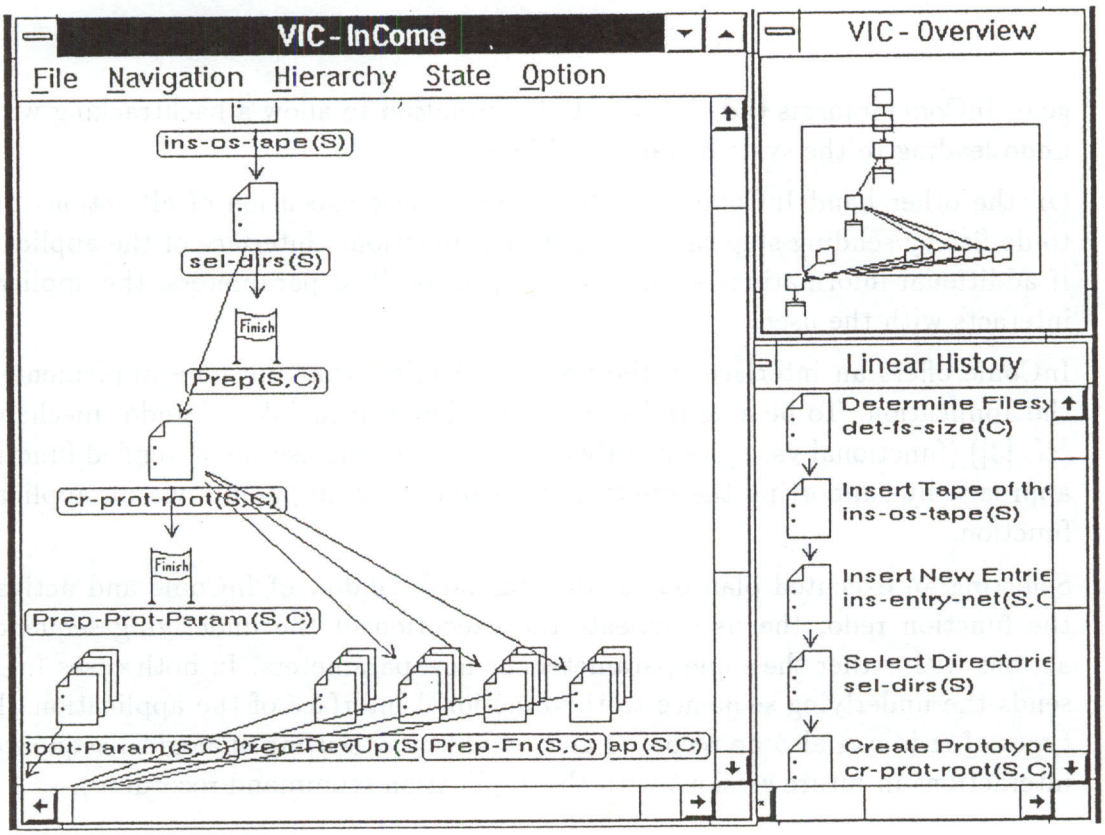


Figure 2: Windows of InCome

and actions are offered in pulldown menus following the interaction style standard implemented by the user interface management system. User actions possible in InCome are divided into three categories:

- Graphical navigation includes actions like overview, scrolling and searching specific nodes. The overview window contains the whole visualized interaction structure by performing operations to display it on a reduced scale. It supports several navigation aids (e.g., indirect and direct positioning of the standard window of InCome).

- Hierarchical navigation supports the user in viewing different levels of plans. A plan can be a subplan of a higher level plan. Plans can be opened to see the underlying sequences of actions or subplans. On the other hand, actions that are associated to the same goal can be combined and represented as one plan. In addition, InCome provides the ability to focus on plans comparable to a fish-eye view. Every object out of interest is abstracted to such a level that the focused plans are not effected.

- InCome allows the user to act on the application remotely by offering him a tutorial mode and access to undo-mechanisms and redo-mechanisms. In general, the user expresses reachable goals to be guided to by InCome. If the user selects a reachable goal and activates the tutorial mode, InCome requests a sequence of actions from the plan generator and visualizes this sequence in a to-do-list shown to the user in a separate window. Then InCome supervises the executed user actions by marking performed actions in the to-do-list and showing the next steps necessary to reach the selected goal. If the user has performed an action that compromises the selected

goal, InCome informs the user about this situation to allow a backtracking with an undo leading to the system state valid before.

On the other hand InCome provides an automatic execution of all actions in the to-do-list by sending stepwise actions to the functional interface of the application. If additional information is required, e.g., unspecified parameters, the application interacts with the user.

InCome offers an interface to the undo-mechanism that must be implemented by the application. To be able to handle two different principles of undo- mechanisms (cf. [3]) (functional vs. system-state oriented), InCome uses an extended functional approach by supporting the resetting to explicit freezing points via an application function.

Selecting an executed plan out of the standard window of InCome and activating the function redo, the user repeats the execution of the underlying sequence of actions with either the same parameters or new parameters. In both cases InCome sends the underlying sequence to the functional interface of the application. Both types of redo can also be recorded and reused by executing analogical sequences of interactions in future sessions with the application (command recording).

# 4 System Architecture

The system architecture of InCome is splitted into three components interacting with the user and one component building up the internal representation of the interaction history (cf. fig. 1). The parser generates the hierarchical interaction history from a syntactically formalized plan-based interaction history coming either from the plan recognizer/generator or from the interaction history. The hierarchical navigation module produces the graphics-oriented representation of the interaction history which is afterwards interpreted by the graphical navigation module to generate the several displays like the standard window, the overview window and the linear dialog history. This module also implements the graphical navigation aids.

Activating the tutorial mode causes the appropriate module either to extract a valid sequence of actions from the hierarchical interaction history or to request it from the plan recognizer/generator. The redo and undo modes generate the appropriate sequences of actions and send them to the functional interface of the application. InCome is a separate application and is implemented in Smalltalk/V PM running on an IBM PS/2 under OS/2.

# 5 Conclusions

InCome provides the user a quick and helpful reminder on the system state to resume suspended applications. It assists the user in leaving system states unfamiliar to him. The combination of undo- and redo-mechanisms on a high level of interaction abstraction supports him in exploring alternative actions which would have been executable next. The combination of the undo- / redo-mechanisms with the high level visualization of interactions gives the user the opportunity to perform an undo-redo on a semantic level

(semantic undo / semantic redo) because he can activate it on plans rather than on single actions.

The presented interaction control manager InCome can be integrated in a user interface design environment like UIDE (cf. [2], [4]) in such a way, that on user demand the tutorial mode of InCome presents an animation of the necessary user steps to reach the specified user goal.

# Bibliography


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

[2] 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.

[3] 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.

[4] 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.

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

[6] 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.

[7] R. Wilensky, Y. Arens, and D. Chin. Talking to UNIX in english: An overview of UC. *Communications of the ACM*, 27(6), June 1984.

# Plan-Based Graphical Help in Object-Oriented User Interfaces

Markus A. Thies and Frank Berger
German Research Center for Artificial Intelligence (DFKI)
Saarbrücken Site, Stuhlsatzenhausweg 3
DW-6600 Saarbrücken 11
Germany
{thies,berger}@dfki.uni-sb.de



**Abstract**

This paper describes the system PLUS, a plan-based help system for applications offering an object-oriented user interface. Our plan recognition process is based on a predefined static hierarchical plan base, which is modeled using a goal plan language. This language is designed to especially cope with the problems arising when plan recognition is performed in a graphical user interface environment whose interaction is based on a user-directed dialog by means of direct manipulation — so-called Direct Manipulation User Interfaces. The plan hierarchy is entered using the interactive graphics-oriented plan editor PlanEdit$^+$. The plan recognition module PlanRecognizer$^+$ builds a dynamic plan base by mapping user actions to plans stored in the static plan base. The dynamic plan base contains hypotheses about tasks the user is pursuing at the moment. These plan hypotheses serve as a basis to offer various kinds of assistance to the user. A central component of our graphical help is the module InCome$^+$. InCome$^+$ visualizes user actions previously executed in an application as a graph structure and enables the user to navigate through this structure. A higher level of abstraction on performed actions shows the dialog history, the interaction context, and reachable goals. InCome$^+$ offers special features like task-oriented undo und redo facilities and a context-sensitive tutor. A substantial extension of the graphical user assistance is the integration of animated help within PLUS. Animation sequences are generated in the context of the tasks the user is currently working on.

# Contents

# 1 Introduction

The overall objective of PLUS is the design and the implementation of a plan-based help system. Rather than doing basic research, the state-of-the-art methods of several fields in Artificial Intelligence like *Knowledge Representation* and *Plan-based Systems* should be incorporated. In the context of plan-based help systems, a plan is a sequence of actions that have to be executed to complete a given task, thereby achieving a specific goal. Unlike known help systems that where mostly developed for command-based interfaces (see, e.g., [Finin 83], [Fischer et al. 85], [Wilensky et al. 88], [Wahlster et al. 90], [Bauer et al. 91]), PLUS is designed to cope with applications that offer graphical user interfaces, whose main interaction principle is based on a user-directed dialog by means of direct manipulation (cf. [Shneiderman 83], [Shneiderman 87]) — so-called **D**irect **M**anipulation User **I**nterfaces (DMI).

A special feature of such a DMI-environment is the flexibility of the user in performing tasks within an application. There usually exists neither a definite action sequence for the execution of a plan nor a fixed number of actions required for accomplishing a given task. Moreover, in a DMI-environment it is possible to work on different tasks in parallel and to switch between them arbitrarily. For an expert user, this flexibility makes it easier and more comfortable to handle such a system. An unexperienced user, however, may easily get confused and encounter difficulties when working with the application. In such a situation, he may want to ask a human expert for help. The help system that is developed within PLUS, should replace such an expert. In contrast to manuals or static online help facilities usually supplied with an application, the PLUS help system is able to deal with the user's problems in a context-dependent manner by giving him advise related to the tasks he is performing at the moment.
In order to meet these demands, the following help strategies are designed within PLUS:

- Passive help:
  The user explicitly requests help.
  Context sensitive help information is generated.

- Active help:
  The user gets help without explicitly requesting it.
  E.g., the system offers the user an optimized interaction sequence to reach a specific goal.

- Cooperative help:
  The user gets help when he runs into an error condition.
  The system suggests possible corrections or recommends alternative solutions to the user.

- Implicit help:
  The system adapts itself by, e.g.,
  - changing the screen layout,
  - focusing the user's attention,
  - setting defaults.

The PLUS prototype has been developed using a Smalltalk system. The design of the prototype follows progressive methods in application development and concepts of object-oriented software development (cf. [Wisskirchen 90], [Booch 91]).

# 2 The PLUS System

The main module of a plan-based help system is a plan recognizer. While the user interacts with the application, the plan recognizer tries to map the performed actions to plans, thereby making assumptions about the goals intended by the user. These plan hypotheses form the basis for offering various kinds of help to the user.

There exist two different approaches for plan-based systems. On the one hand, there are systems that generate plans during run-time using a *plan generation system* (see, e.g., [Bauer et al. 91]). This approach is also called plan recognition from *first principles*. On the other hand, there are systems that use a predefined plan-base as an input for the plan recognition component (plan recognition from *second principles*). The plan recognizer used within PLUS is based upon the second approach.

Coping with different DMI-events, we use a two-level plan recognition approach. With this two-level approach we are able to process the low-level events without stressing the actual plan recognition process.

The first level processes low-level events like mouse-clicks and keystrokes. Upon the first level, we protocol the user's favourite interaction styles — does he mostly use the mouse or does he prefer 'short-paths' — and build up a simple user model to reflect the user's preferences (user modelling see, e.g., [Kobsa & Wahlster 89] [Rich 89]).

First, the user model can be employed for adapting help information to the user's habit by considering his prefered interaction styles, and second, it allows to detect alternative interaction principles that are unknown to the user. Moreover, while generating help sequences, the first level of the plan recognition is used to determine the most effective interaction technique to perform a specific action.

The results of this first plan recognition level are application-specific actions performed by the user by, e.g., selecting pulldown items. These actions are recorded within a *Dialog History* that serves as an input for the second level plan recognition process, realized through the module *PlanRecognizer*$^+$.

Using a *spreading activation* algorithm, PlanRecognizer$^+$ tries to map actions stored in the dialog history to plans in the predefined hierarchical plan base. We use a hierarchical plan base to offer a user-adequate assistance on a suitable abstraction level, and in order to guarantee an efficient plan recognition process. Logically coherent action sequences may be part of various plans. Therefore, it is obvious to combine them to an independent plan, and to define this plan as a subplan in the corresponding, more abstract plans. Thereby, we obtain a plan hierarchy with several layers. This so-called *static plan base* is interactively entered using the graphics-oriented plan editor PlanEdit$^+$ (cf. section 3).

The spreading activation algorithm builds up the so-called *dynamic plan base* at run-time. The dynamic plan base contains hypotheses about plans and goals the user is pursuing at the moment. Together with a knowledge base containing common help strategies extended by rules and facts about generic interface concepts, these hypotheses serve as a basis for the help component and for InCome$^+$ (cf. figure 1).

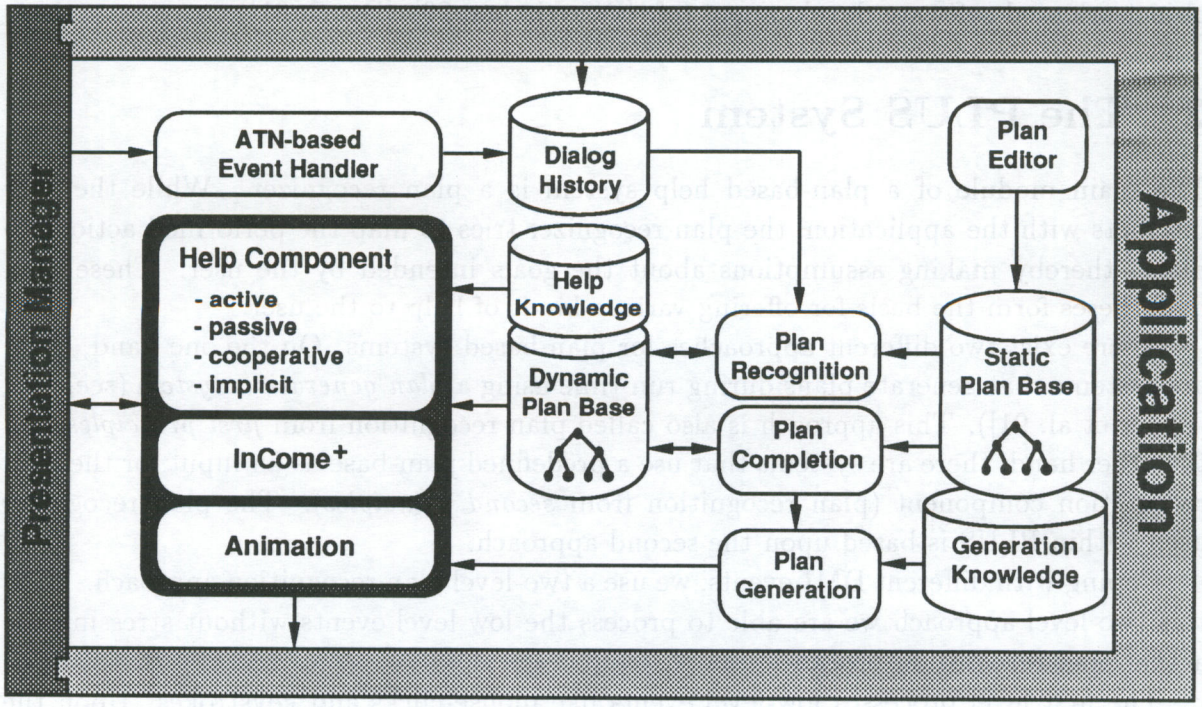


Figure 1: PLUS System Architecture

DMI-environments enable users to perform tasks in a very flexible way. Due to this, the number of considered plan hypotheses may increase without additional mechanisms to reduce their number. Therefore, we provide the following focussing methods:

- We introduced a *time frame* concept that splits plan hypotheses into different classes, considering the time stamps of their last activation. This classification serves as an instrument to decide, which plans are presented to the user if he requests help.
- For each plan a list of *cancel actions/goals* may be specified. The execution of a cancel action or the achievement of a cancel goal immediately dismisses the respective plan hypothesis. E.g., if the user explicitly removes an object used within a plan, this plan is dismissed.
- For each plan a list of *views* may be specified. A plan is only activated, if it is performed within one of its associated views. This concept supplies an additional cancellation mechanism. If a view is closed, all plans associated with that view can be dismissed.

The term *view* describes windows having a specific type. E.g., a window representing hierarchical information like father-child-relations about some data (hierarchical view) or a window showing vertical information like predecessor and successor relations of the same data (vertical view). By introducing *views*, specific actions may only be possible in specific views. E.g., connecting nodes that are on the same level of abstraction within a hierarchy is possible in the *vertical view*. In contrast, building father-child-relations is

only possible in the *hierarchical view*. In addition to the restriction of actions to views, plans may also be restricted to be performed completely within a specific view.

To enter the static hierachical plan base, the interactive graphics-oriented plan editor *PlanEdit*$^+$ is used, which is described in more detail in the next section.

# 3 The Plan Editor

The plan hierarchy comprises three types of objects, *actions*, *plans*, and *goals*. It is organized as follows (cf. figure 2):

- The lowest layer consists of the *actions* the user can perform when interacting with the application. Actions may be parts of plans.
- A *plan* is defined by a set of actions and/or subgoals. Each plan is associated with exactly one goal that is reached, if the plan is completed.
- A *goal* may be achieved in different ways, each of them describing an alternative plan. Some of them may be suboptimal or wrong. Goals may be contained as subgoals within more abstract plans.

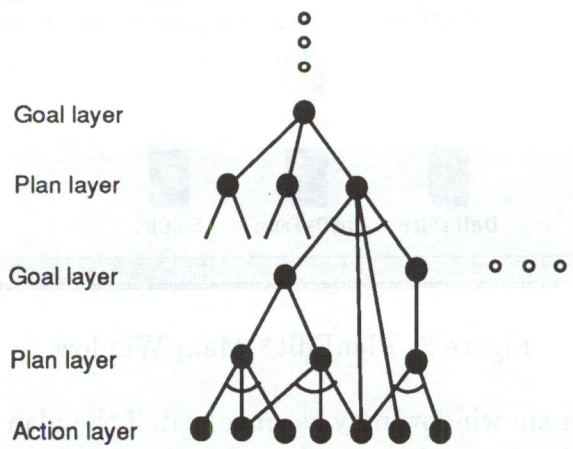


Figure 2: Plan Hierarchy

For each type of object in our planbase, certain properties may be defined. Therefore, we developed the plan description language GPL$^+$ (**G**oal **P**lan **L**anguage). By means of GPL$^+$ we are able to model several problems closely related to DMI environments like *optionality*, *multiple selection*, *iteration*, *parallelity*, and multiple *views* on objects. Besides, features common to plan recognition like parameter and temporal constraints, plan cancellation and plan interactions can be modeled. In addition, the elements of the plan base may be supplied with hierarchy information, thereby defining the structure of the plan hierarchy.

Grounded on the structure of our plan base — an object hierarchy with specific properties per element — we decided to chose an object-oriented internal representation for

the plan base. In order to offer the plan designer a comfortable input tool, we developed the graphics oriented plan editor PlanEdit$^+$, that allows to build up the plan base interactively within a DMI environment.

Figure 3 shows the PlanEdit$^+$ main window, in which the biggest part of the interaction takes place. The elements of the plan base are displayed as graphical objects. Each object consists of an icon representing the element's type and the element's name. The *Type Box* in the lower left corner of the main window contains icons for the three types of elements contained in the plan base: actions, plans and goals. These icons can be used to generate new elements of the respective types. The properties of the elements may be defined within a series of dialog boxes.

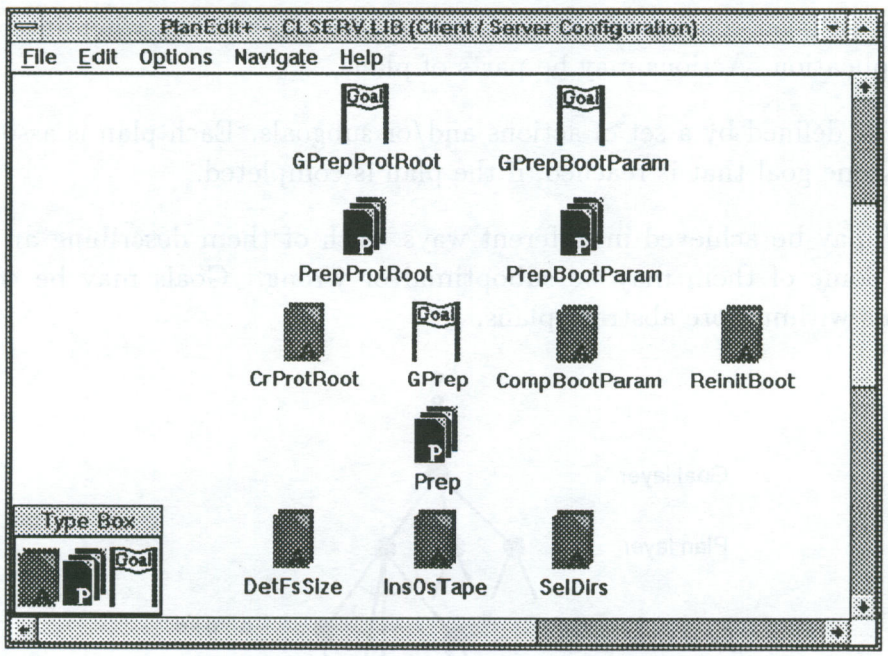


Figure 3: PlanEdit$^+$ Main Window

The contents of the main window may get involved, if the plan base grows. Therefore, we added a second window type that enables us to separately examine the structure of previously defined plans and goals, and to easily modify their properties. Figure 4 shows an example of a plan window.

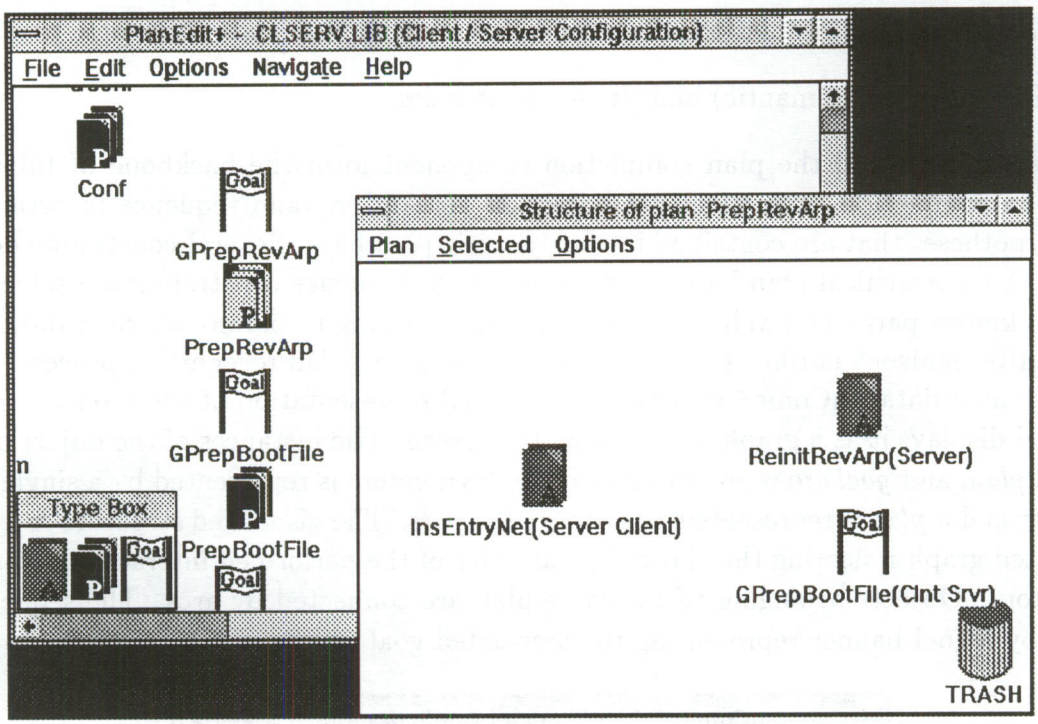


Figure 4: Different Window Types

In the main window, the elements of the plan base may be arranged arbitrarily without considering the structure of the plan base. Within a plan window, however, the layout of the objects corresponds to the logical sequence of the elements within the plan, as defined through the sequence constraints.

After the plan base is completely specified by using PlanEdit$^+$, a corresponding Smalltalk module modeling the internal object-oriented representation of the static plan base is generated.

# 4 The Module InCome$^+$

One of the central components for graphical help within the system PLUS is the *Interaction Control Manager* InCome$^+$ (cf. [Thies 90], [Fehrle & Thies 91]). It provides a graphical visualization of the current dialog context, the dialog history, and possible future interactions. InCome$^+$ gives the user a quick and helpful reminder on the system state to resume suspended tasks. It supports the user in leaving system states unfamiliar to him and in exploring actions (cf. [Paul 89]) that can be executed next to complete unfinished tasks. InCome$^+$ meets the following demands:

- Adequate visualization of user interactions,
- Display of different levels of abstraction selectable by the user,
- Visualization of possible future interactions,
- Graphical navigation services,

- Display of plan interactions, like embedded, overlapping, and interrupted plans, and
- Task-oriented (semantic) undo/redo capabilities.

PlanRecognizer$^+$ and the plan completion component form the backbone of InCome$^+$. The plan completion component generates, on demand, a valid sequence of actions for plan hypotheses that are contained in the dynamic plan base. Several constraints defined within the hierarchical plan base are satisfied. E.g., sequence constraints are solved and already known parameter values are propagated according to parameter constraints.

PlanRecognizer$^+$ notifies InCome$^+$ about the ongoing plan recognition process. Upon the incoming data, InCome$^+$ generates an internal representation of the interaction context and displays it as a graph structure on the screen. The instances of the object classes *action*, *plan*, and *goal* are represented as nodes. An *action* is represented by a single sheet of paper and a *plan* is represented by a stack of papers. The visualized structure resembles a directed graph reflecting the chronological order of the performed interactions from top to bottom. Objects belonging to the same plan are connected by arcs. The sequence is ended by a goal banner representing the associated goal (cf. figure 5).

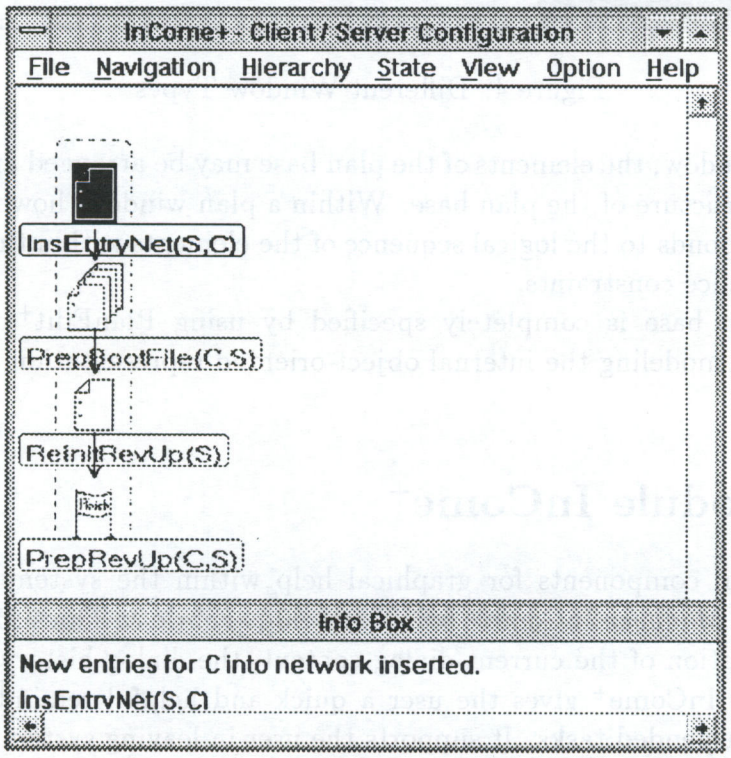


Figure 5: Visualized Elements within InCome$^+$

InCome$^+$ runs in its own window. The presented nodes are selectable via mouse clicks. User actions provided by InCome$^+$ can be divided into four categories:

- Graphical navigation,
- Hierarchical navigation,
- Tutor activation, and

- Remote application interaction.

**Graphical Navigation** includes actions like *scrolling, overviewing,* and *searching* for specific nodes. The overview window (cf. figure 6) contains the whole visualized interaction structure by performing operations to display it on a reduced scale. Within the overview window several additional navigational actions like direct and indirect positioning of the standard window of InCome$^+$ are possible.

**Hierarchical navigation** supports the user in viewing plans in different abstraction levels. InCome$^+$ is able to generate a visualization of the different plan interactions that are handled during the plan recognition process.

Figure 7 shows an interaction context where plan *PrepBootParam(C,S)* and *PrepProtRoot(S,C)* overlap each other with the plan *Prep(C,S)*. Plan *PrepRootFile* is embedded within plan *PrepRevUp* as shown in figure 8.

InCome$^+$ supports actions for *expanding* and *collapsing* plans. Expanding equals to a movement down in the hierarchy and collapsing equals to a movement up in the hierarchy. Expanding and collapsing of plans are realized within InCome$^+$ by grouping sequences of actions together to plans or by replacing plans with their sequences of actions. Besides the step-wise vertical movement within the interaction hierarchy an additional feature is provided that works comparable to *Fish-Eye Lenses.* In the *Fish-Eye mode*, every object out of interest is abstracted to such a level that the focused plans are not effected.

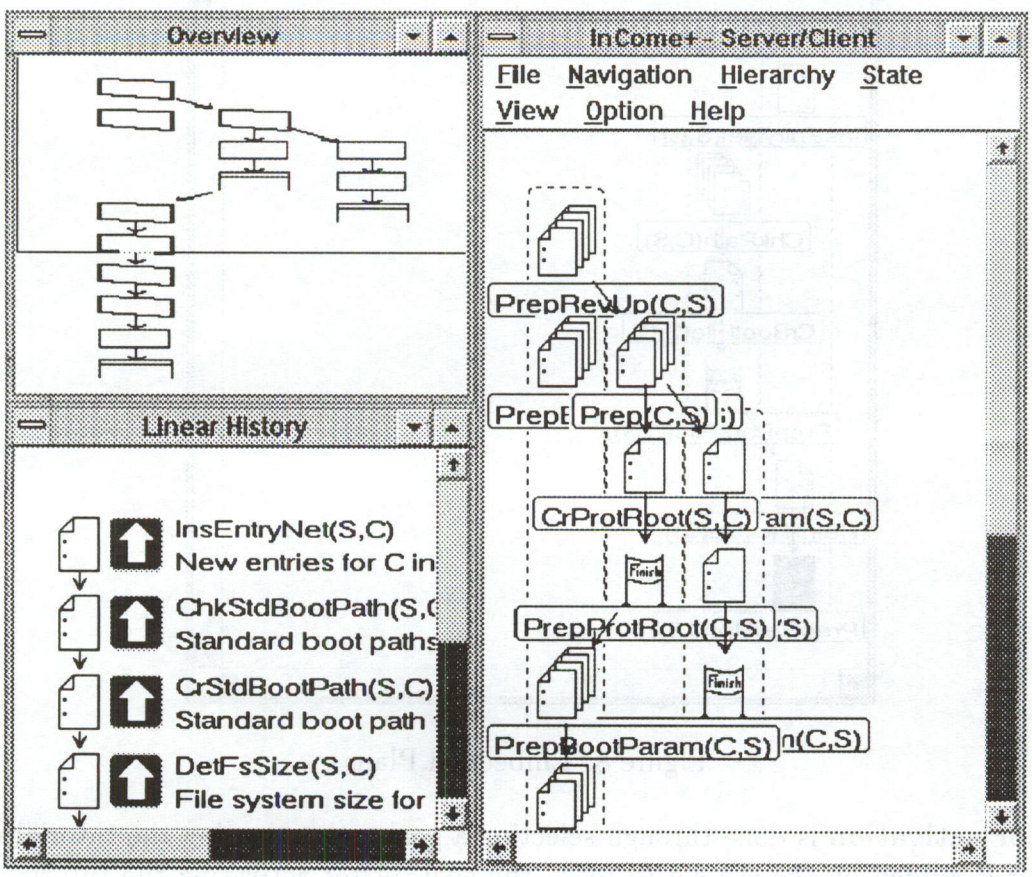


Figure 6: Linear History and Overview

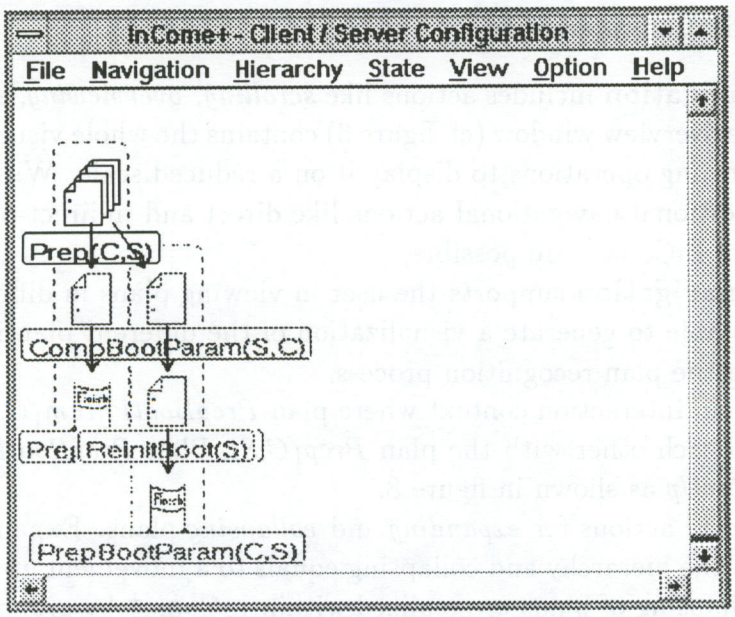


Figure 7: Overlapped Plans

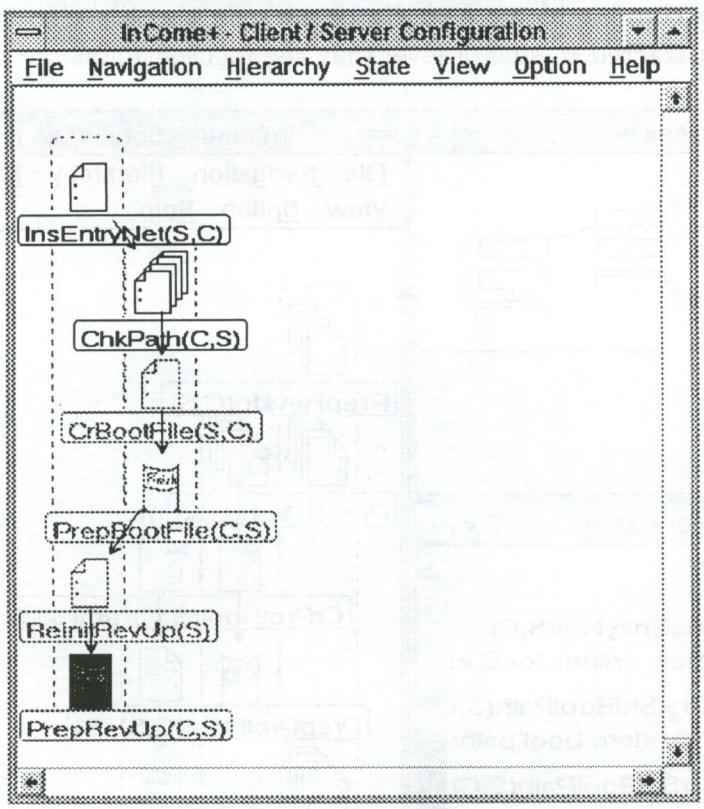


Figure 8: Embedded Plans

**Tutor activation** is done through selecting a goal and activating the tutorial mode. The user is guided by the system to reach the goal. After activating the tutorial mode InCome$^+$ requests an optimal sequence of actions for reaching the selected goal. In this context *optimal* means the most efficient sequence of actions to reach a goal. The plan completion component generates this sequence by considering the different constraints

defined in the hierarchical plan base. Already known argument values are propagated. The sequence of actions is textually represented in a separate window like a *to-do-list* (cf. figure 9). The tutor of InCome$^+$ supervises the executed user actions by marking performed actions in the *to-do-list* and showing the next steps necessary to reach the selected goal (cf. figure 10). If the user has performed an action that compromises the selected goal, InCome$^+$ informs the user about this situation to allow a backtracking by performing undo operations leading to the previously valid system state.

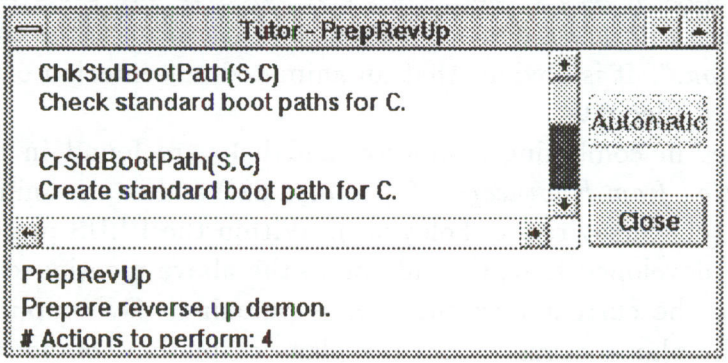


Figure 9: Tutorial Mode

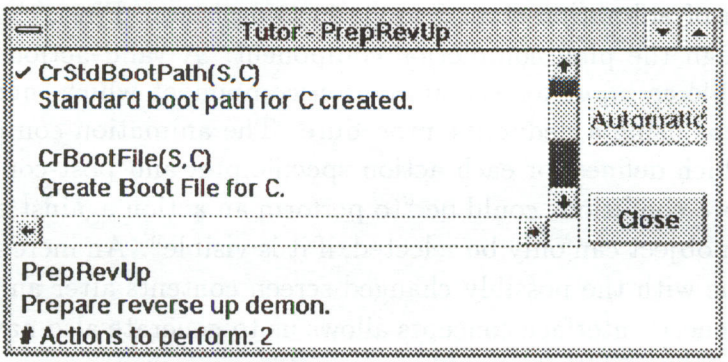


Figure 10: Tutorial Mode and Performed Actions

**Remote application interaction** is provided through the access to *undo-* and *redo-mechanisms* of the application. InCome$^+$ provides an interface to these mechanisms. To be able to deal with two different undo principles (*function-oriented* vs. *state-oriented*; see also [Rathke 87], [Rathke 89], [Yang 90]), InCome$^+$ uses an extented function-oriented approach by handling *freezing-points* (cf. [Paul 89]). Freezing-points are snap-shots of system states that are saved within the application. It is possible to reset the application state to one of these freezing-points by activating the corresponding application function. By representing the interaction context in a more abstract way than by the linear dialog history the user can perform undo-actions and redo-actions on plans rather than on actions. We call this *undoing on a semantic level.* An undo of tasks without reversing tasks following them is not supported (*'freies undo'*, cf. [Rathke 87]).

The linear dialog history is presented in a separate window. The visualization emphasizes *reversible actions* and *freezing-points* that are set within the application. The lower left window in figure 6 represents the linear dialog history. The arrows on the right side of the actions mark *reversible actions.*

# 5 Animated Help

An outstanding claim in providing graphical user assistance is the coupling of animation and help. Conventional help systems and knowledge-based help system reach their limits as soon as the user needs assistance in performing interactions. E.g., the user addresses questions of the form: *"How do I include object A into container-object B?"* or *"Please show me, how do I get objects X, Y, and Z visualized."*. A generated textual help could possibly sound: *"Move mouse to the position of object A and press left mouse button. Now move the mouse with pressed mouse button to the position of the container object B. Release mouse button."*. It is obvious that an animated presentation of these interaction steps would be more adequate.

First approaches in combining animation and help are found in the system *GAK* (*Graphical Animation from Knowledge*, cf. [Neiman 82]) and in the animated help extention of *Cartoonist* (cf. [Sukaviriya & Foley 90]). Within the PLUS project an animation component will be developed that, in contrast to the above mentioned systems, reaches a closer relation to the current task the user is pursuing. By coupling the animation component with the plan recognizer and the plan completion component of PLUS, the animation component can generate a sequence of animated interactions for a specific plan hypothesis.

The generation of animation steps is done in two phases. First, the plan hypothesis is completed through the plan completion component. A valid action sequence for the plan is now available as input for the animation component which incrementally generates animation steps using a deductive procedure. The animation component accesses a knowledge base which defines for each action specific pre- and post-conditions. Informal examples of such pre-conditions could be "to perform an action against an object, it must be selected" or "an object can only be selected, if it is visible". An incremental generation is necessary to cope with the possibly changed screen contents after an animation step.

Representing generic interface concepts allows us to generate also navigational animation steps (e.g., steps to scroll the visible area within a window).

An animation is initiated in the PLUS system by selecting a goal and activating the corresponding menu function. The animation steps are done by imitating user actions and by sending them to the user interface in such a way that it responds to these 'animation inputs' exactly as if they were performed by the user.

# 6 Acknowledgements

The research presented here has been carried out in the PLUS project which is conducted cooperatively by the IBM Laboratory Böblingen, the IBM Germany GmbH, and the DFKI. Following research scientists are involved in the project: Prof. Dr. Wolfgang Wahlster (DFKI), Frank Berger (DFKI), Markus A. Thies (DFKI), Dr. Thomas Fehrle (IBM Laboratory), and Volker Schölles (IBM Laboratory).

Special thanks to Wolfgang Wahlster and Thomas Fehrle for their valuable remarks on this paper.

# Bibliography


[Bauer et al. 91] 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 (eds.), Verteilte Künstliche Intelligenz und kooperatives Arbeiten. 4. Internationaler GI-Kongress Wissensbasierte Systeme, Berlin, Heidelberg, 1991. Springer. Also DFKI Research Report RR-91-26.

[Booch 91] Grady **Booch**. *Object Oriented Design with Applications.* Redwood City, California, USA: The Benjamin/Cummings Publishing Company, 1991.

[Fehrle & Thies 91] Th. **Fehrle** and M.A. **Thies**. *InCome: A System to Navigate through Interactions and Plans.* In: H.-J. Bullinger (ed.), Human Aspects in Computing: Design and Use of Interactive Systems and Information Management, Amsterdam, London, New York, Tokyo, 1991. Elsevier Science Publishers B.V.

[Finin 83] T. W. **Finin**. *Providing Help and Advice in Task Oriented Systems.* In: Proc. IJCAI-83, p. 176–178, Karlsruhe, Deutschland, 1983.

[Fischer et al. 85] G. **Fischer**, A. **Lemke**, and T. **Schwab**. *Knowledge-based Help Systems.* In: Proceedings CHI-85, San Francisco, CA, 1985.

[Kobsa & Wahlster 89] A. **Kobsa** and W. **Wahlster** (eds.). *User Models in Dialog Systems.* Symbolic Computation. Berlin, Heidelberg, New York: Springer, 1989.

[Neiman 82] D. **Neiman**. *Graphical Animation from Knowledge.* In: Proceedings of AAAI, 1982.

[Paul 89] H. **Paul**. *Exploratives Agieren in interaktiven EDV-Systemen.* In: B. Endres-Niggemeyer, T. Herrmann, A. Kobsa, and D. Rösner (eds.), Interaktion und Kommunikation mit dem Computer. Informatik Fachbericht 238. Berlin: Springer Verlag, 1989.

[Rathke 87] M. **Rathke**. *UNDO/REDO - Szenarien und Anforderungen für eine anwendungsneutrale Implementierung.* In: M. Paul (ed.), GI - 17. Jahrestagung Computerintegrierter Arbeitsplatz im Büro, Berlin, Heidelberg, New York, London, Paris, Tokyo, 1987. Springer.

[Rathke 89] M. **Rathke**. *Erweiterung interaktiver Anwendungen um Undo-Mechanismen.* In: Software Ergonomie: Aufgabenorientierte Systemgestaltung und Funktionalität, GI Band 32, Stuttgart, 1989. Teubner.

[Rich 89] E. **Rich**. *Stereotypes and User Modeling.* In: Kobsa and Wahlster [Kobsa & Wahlster 89], p. 35–51.

[Shneiderman 83] B. **Shneiderman**. *Direct Manipulation: A step beyond programming Languages.* IEEE Computer, 16, 1983.

[Shneiderman 87] B. **Shneiderman**. *Designing the User Interfaces: Strategies for effective Human-Computer Interaction.* Massachusetts: Addison Wesley, 1987.

[Sukaviriya & Foley 90] 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), p. 152–166, Snowbird, Utah, October 1990.

[Thies 90] M. A. **Thies**. *Interaction Control Manager: Ein System zum Navigieren durch Interaktionen und Pläne.* Master's Thesis, Fakultät Informatik, Universität Stuttgart, Deutschland, 1990.

[Wahlster et al. 90] W. **Wahlster**, D. **Dengler**, M. **Hecking**, and C. **Kemke**. *SC: The SINIX Consultant.* In: P. Norvig, W. Wahlster, and R. Wilensky (eds.), Intelligent Help Systems for Unix - Case Studies in Artificial Intelligence. Heidelberg: Springer, 1990.

[Wilensky et al. 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.

[Wisskirchen 90] P. **Wisskirchen**. *Object-Oriented Graphics. From GKS and PHIGS to Object-Oriented Systems.* Symbolic Computation. Berlin, Heidelberg: Springer-Verlag, 1990.

[Yang 90] Y. **Yang**. *Current Approaches & New Guidlelines for Undo Support Design.* In: H.-J. Bullinger and B. Shackel (eds.), 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:* $\Pi_{ODA}$: 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

Intelligent User Support in Graphical User Interfaces:

**RR-92-14**
Research Report

## 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