

SUPPORTING THE DESIGN OF OFFICE PROCEDURES IN THE DOMINO SYSTEM

Frank Victor, Edgar Sommer

Gesellschaft für Mathematik und Datenverarbeitung (GMD)
P.O.B. 1240
D-5205 St. Augustin 1
West Germany

ABSTRACT

In this paper, we show how a graphical interactive planner is used to support the design of specifications for an office procedure system. In a first phase, a plan is designed relying on a knowledge base which describes the given organization. This plan is represented as a modified Petri net and can be tested and revised using simulation. In the second phase, the plan is transformed automatically into a formal specification of an office procedure, in our case a CoPlan-S procedure for the DOMINO system.

In the first part of the paper we briefly describe the planning system VIPS and the office procedure system DOMINO. Then we explain the transformation process and illustrate the generation of a procedure specification with the help of an example.

1. Introduction

The description of activities in an office environment is an active field of research. In recent years, several models for the support of cooperative work have been proposed, for example the Amigo Activity Model [PaDa89] or SDL [BoCh88]. Our group at GMD is concerned with conversational systems, a model which views cooperative work as a formalized exchange of messages [KrWo88]. The office procedure system DOMINO [KrWo86] and the date planning system TVS [PrWo88] belong to this class. These systems manage the execution of communication oriented procedures in an office environment.

We stipulate that the design of such procedures in general, and the generation of descriptions in particular (for example in CoPlan-S, the specification language for office procedures in DOMINO), pose some problems and are worthy of computer support. Naturally, before such a procedure description becomes a fixed regulation in an organization, discussion among the departments involved is necessary. But we believe a planning support system to be useful in the initial design phase. The

specification languages themselves do not facilitate quick human interpretation. Translation of the textual description into a visual one would allow a better overview. Additional support in the form of information about the organization could also simplify the design phase and protect from inconsistencies with previously designed procedures. We have therefore decided on a combined graphical knowledge-based approach.

In our support system VIPS, the design of a DOMINO office procedure is treated as a process of interactive planning. A plan is a modified Petri net, representing activities, actors and objects. An office procedure is designed by constructing a plan and translating it into a CoPlan-S specification.

The construction of a plan relies on access to a knowledge base which describes the organization [MaVi87, MaVi89]. We discern between *object-knowledge*, which describes the hierarchical structure of the organization, the objects that make up the organization (e.g. forms, purchase objects), the roles of employees, and *procedural knowledge*, which describes permitted actions in the organization and relationships between them. The planning process is described in [MaVi88a].

The plan constructed in VIPS contains more information than is relevant to a DOMINO procedure, which is concerned only with documents, roles and certain kinds of actions. During the translation phase, the plan is transformed into a CoPlan-S specification by filtering out unnecessary information, introducing CoPlan-S alternative actions to resolve conflicts and determining the initiator of the office procedure.

We have made the experience that it is simpler and more intuitive to formulate an office procedure visually with VIPS than to write code in a specification language. The graphical representation offers a general view of the structure of the plan at every point in the design process. The knowledge base supports the designer of a CoPlan-S procedure by providing access to the rules and regulations of his organization.

2. The planning system VIPS

VIPS is an interactive graphical planning system. It is used for planning procedures in the office domain. The creation of a plan is interpreted as a cooperative design process between user and planning system. One of the main advantages of this approach is that the planning goal does not have to be specified at the beginning of the planning process but is worked out by the user and the system, i.e. is made more concrete during the planning process. This problem solving paradigm - which we call *support of planning* - is well suited for open systems like the office domain, in which not all situations that may occur are predictable [MaVi88a].

VIPS uses a knowledge base that contains knowledge about problem domains in which the planning process can be executed. We have represented knowledge about our own organization, the GMD, using the

LUIGI development environment [Kind88]. LUIGI facilitates the acquisition of object structures and rules. Figure 1 shows a part of the object hierarchy of our knowledge base. The main object classes we use are: purchase objects, forms, the organizational structure of GMD, actions and roles.

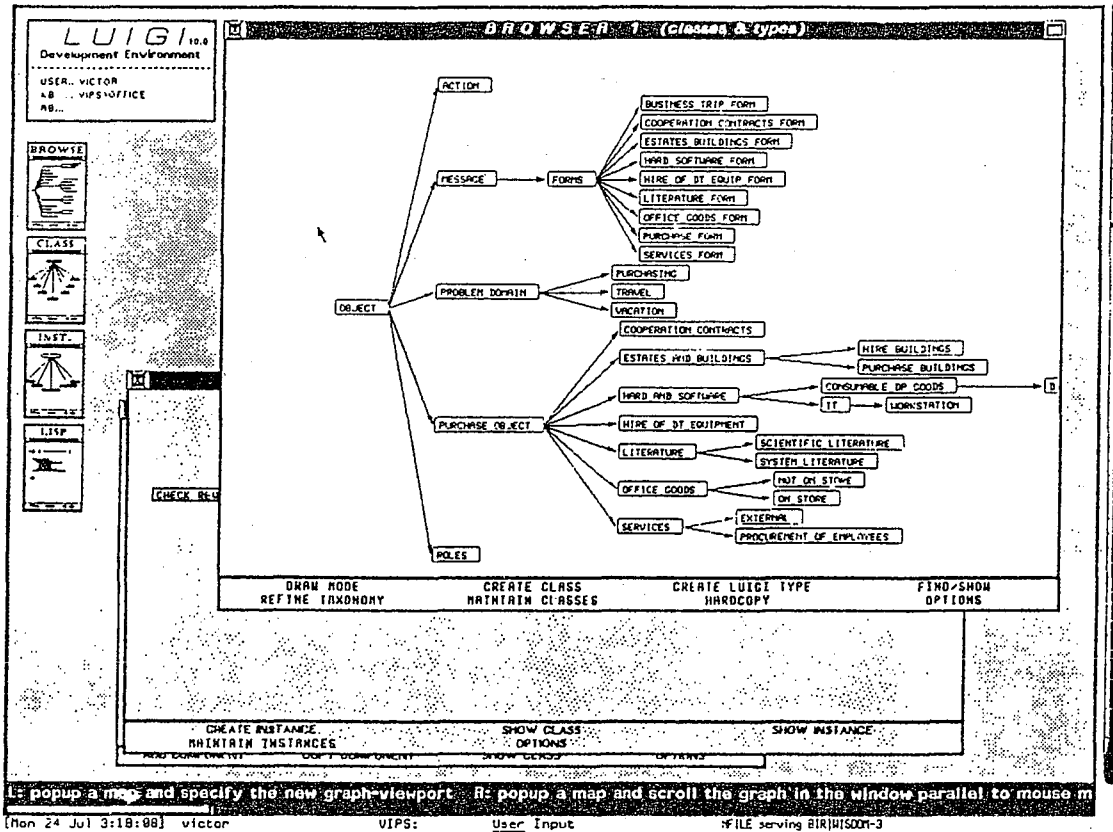


Figure 1: A part of the knowledge base used in VIPS

Actions contain information about related objects, i.e. objects that are needed for or produced by the action, preconditions for the applicability of an action and details about the inner structure of complex actions. Roles specify the responsibility of persons and organizational units for actions.

We use a graphical language for the representation of plans: *modified Petri nets* that model objects, actions and their relations. A plan is a *system* in which concurrency, conflicts and causal dependencies are modeled. Petri nets can be simulated, i.e. beginning with an initial situation, all possible goal situations of the underlying system can be calculated. The advantage of simulating a plan is that the user gains an overview over all possible action sequences that might occur during the execution of the plan.

The planning process in VIPS can be divided into two steps: the synthesis phase and the simulation phase. In the synthesis phase, a plan is constructed cooperatively by the planning module and the user. In the second phase, the plan is simulated and can be tested by the user.

Figure 2 shows the architecture of VIPS. Plans are created and displayed in the *Visual Planning Interface*. The actual plan, specified as Prolog facts, is stored in the *Plan File* which is accessed by the *Planning Module*. The Planning Module uses knowledge stored in a LUIGI knowledge base. The left branch in figure 2 shows the possibility of creating a DOMINO procedure. DOMINO procedures are expressed in a special language, the CoPlan-S specification language. The *CoPlan-S file* serves as input for the DOMINO office procedure system.

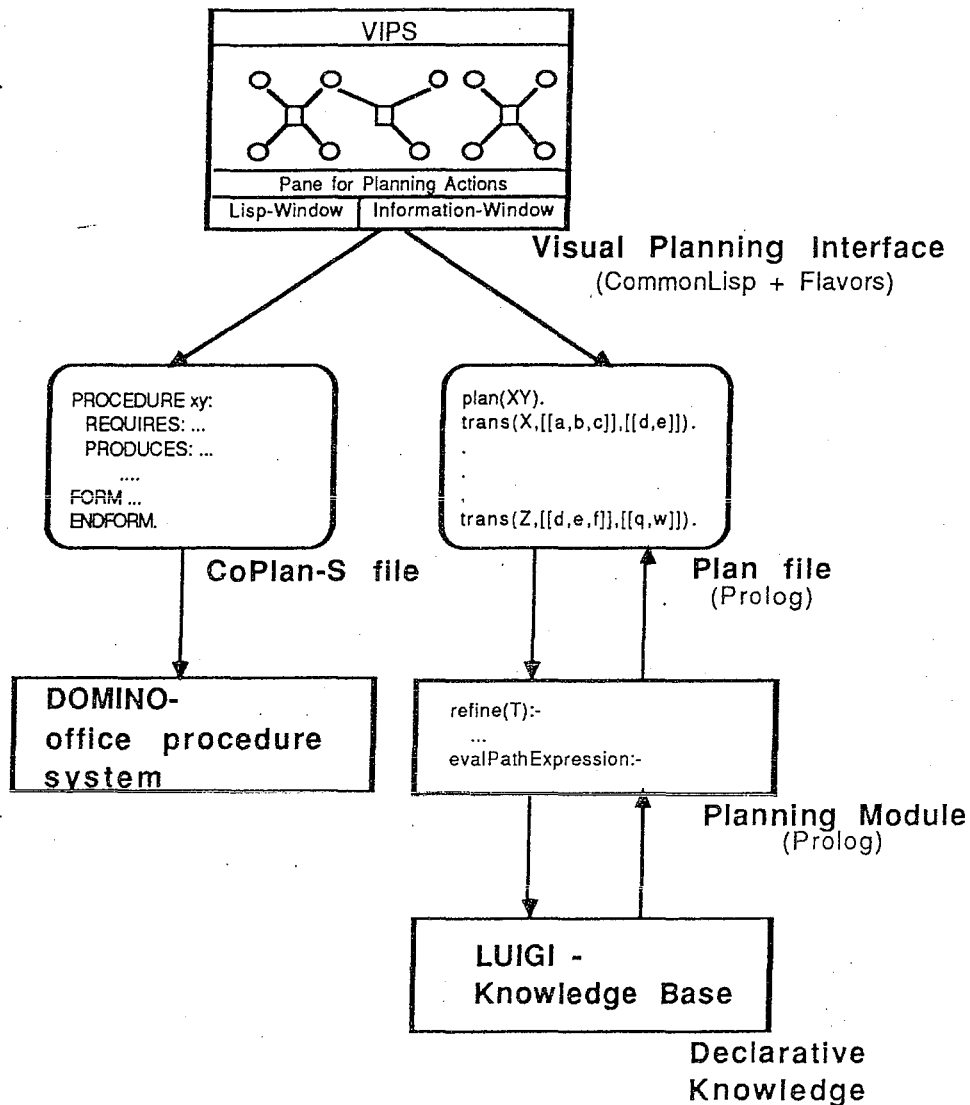


Figure 2: Architecture of VIPS

We now summarize the functionality of VIPS. There are four categories of functions that can be called by clicking items in pull-down menus or buttons (see figure 5): editing facilities, functions of the history component, functions for simulation and planning functions.

The Visual Planning Interface provides the functionality of a graphical editor for creating modified Petri nets, e.g. creating, deleting, moving and renaming arcs, places and transitions. All created plans are stored and can be accessed in the history component. The simulation component provides facilities for simulating a plan.

There are several planning functions. The most important is the function *Refine*, which substitutes a more detailed subnet for a single action. There are two possibilities for the refinement of an action: Substitution of a subnet that is stored in the knowledge base, or creation of a subnet using a path expression [ViSo89]. Other planning actions are: *Check Precondition*, where a necessary predecessor action for an action is integrated into the plan, and *Search Actors* which determines responsible actors for the actions in the plan. The information is again extracted from the knowledge base, i.e. from the organizational hierarchy. Information about objects that are contained in the knowledge base can be shown by the function *Help*, which provides menu techniques for specializing and generalizing objects.

Conflict and decision situations may be generated in the course of plan construction. A *conflict situation* occurs in a plan when a single place serves as input for several transitions (see figure 3). This can be interpreted as a situation in which a resource is prerequisite for several actions but only available once, such as a form that has to be filled out by several office workers. The function *Resolve Conflict* deals with this situation by attempting to duplicate the place in question and integrate the copies into the plan.

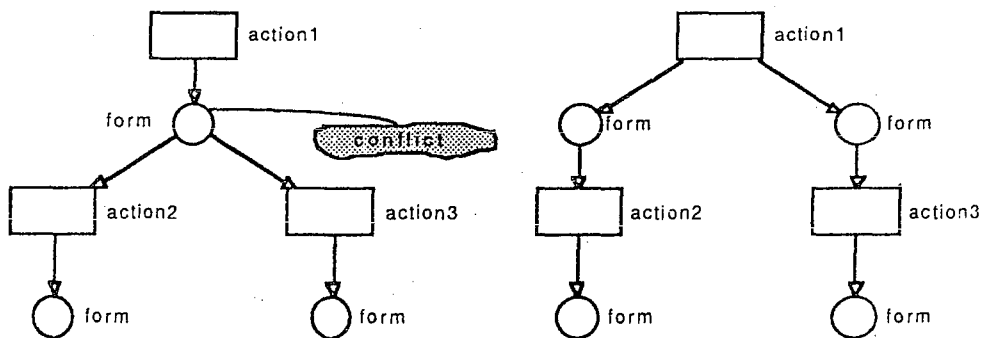


Figure 3: A conflict situation and its resolution

A *decision situation* is structurally similar, but emerges from a different context in the knowledge base. Here, the two related transitions are explicitly marked as being *alternative*, i.e. only one may fire. The VIPS-function *Resolve Decision* deals with this situation by creating a distinct branch in the plan for each alternative transition.

3. Creating DOMINO office procedures using VIPS

3.1 CoPlan-S specifications for DOMINO office procedures

When a plan has been designed and the user has tested it by simulation the function *create CoPlan-S* can be invoked. This function calls a special module of VIPS, the *CoPlan-S generator*. This component is implemented in Prolog and realizes a set of transformation rules that transform a Petri net created by the planning module into a CoPlan-S specification.

DOMINO is an office procedure system in which a procedure is viewed as a series of coordinated information-exchange actions between several persons, with the goal of fulfilling a certain task. Examples of such procedures are the application for a business trip or the processing of a purchase order.

Every action in a procedure can be described via its input/output behaviour, i.e. by the documents it requires (input channels) and the documents it produces (output channels). An action can have input and output alternatives which consist of input and output channels. This means that an action does not have to process all of its input documents and may produce only a subset of its output documents.

CoPlan-S (Coordination Procedure Language) is intended for the specification of office procedures in the DOMINO system. Let us now informally discuss the example of a generated specification in figure 6 to briefly explain the language components of CoPlan-S (for a complete definition see [WoKr87]).

Example (compare figure 6):

The CoPlan-S specification defines an office procedure with the name *business_trip*. The procedure requires the documents *trip2* and *trip1* and produces two alternative results: a document *decision1*, if the business trip is not approved, otherwise two documents *decision2* and *business_trip_form1*. Numbers following the names for documents denote copies of the same document and are generated automatically by VIPS. The actions are *or_APPROVE_YES__APPROVE_NO* and *plan_trip_details* with the actors *sup* (superior) and *td* (travel department). For every action it is specified, as for the complete procedure, what documents are required and produced. The keyword *FORM* specifies a form that occurs in the procedure, the keyword *TEXT* a document which is not a form. For every form, its list of attributes is specified, e.g. *business_trip_form1* has attributes *kind-of-message*, *form-number*, *for*, *destination* and *dates*. For the text documents a default text is specified. The last part of the CoPlan-S procedure are the roles. Roles are the actors of actions that occur in the procedure, in our case *td* and *sup*.

The DOMINO system checks whether a CoPlan-S procedure is (semantically) correct. The procedure can be viewed as a Petri net. A CoPlan-S specification is correct if the corresponding net has the following structural properties:

- The net must be connected and without cycles.
- Every action has at least one input and one output channel.
- Every channel must occur as an input channel of the procedure or as an output channel of an action.
- Every channel must occur as an output channel of the procedure or as an input channel of an action.

Beside these properties there are some demands to the dynamic behaviour of a procedure, e.g. the absence of deadlocks and actions that can never be executed (for further details see [WoKr87]).

3.2 The CoPlan-S generator of VIPS

The CoPlan-S generator transforms a VIPS plan into a CoPlan-S specification. The plan that has been generated contains more information than is relevant for an office procedure. This information has to be filtered out. On the other hand, information is provided that has to be integrated into the procedure in the right form. We now explain briefly the transformation rules that are implemented in Prolog.

The following steps are executed in creating a CoPlan-S specification:

- generation of the PROCEDURE entry
- processing of forks
- generation of the procedure

Generation of the PROCEDURE entry

The PROCEDURE entry is the first one in a CoPlan-S specification (compare figure 6). It denotes the name of the procedure and the documents which are required for or produced by the execution of the procedure. The PROCEDURE entry combines the start and end actions of the office procedure which produce and need these documents, respectively. These actions are carried out by the initiator.

In VIPS a plan is always generated for a special application domain, e.g. the purchase or the business trip domain. For every domain there are initiator actions in the plan that can be interpreted as the start and the end of the procedure. In the case of the purchase domain, the initiator is the person who demands and gets goods. The start action is *demand* and a possible end action is *get_good*. In the case of the business trip domain the initiator is the person who applies for the business trip and executes it.

Input to the start action of a VIPS plan and output of the end action have no corresponding parts in an office procedure. Therefore, these places are inserted into the PROCEDURE entry of the CoPlan-S specification as comments starting with EXTERNAL PROCEDURE INPUT or EXTERNAL PROCEDURE OUTPUT. This information is not strictly necessary for the execution of the procedure by DOMINO.

Processing of forks

First, the plan is checked for *fork situations* (see figure 4). If the transitions involved have different actors, the fork represents a conflict situation (compare section 2) and the plan is not yet transformable into a CoPlan-S specification. The conflict must be resolved on the VIPS side, using the planning function *Resolve Conflict*. If the transitions have the same actor, a CoPlan-S action of type *alternative* is generated. The conflicting actions are integrated into one which models the decision over which course of action is to take place.

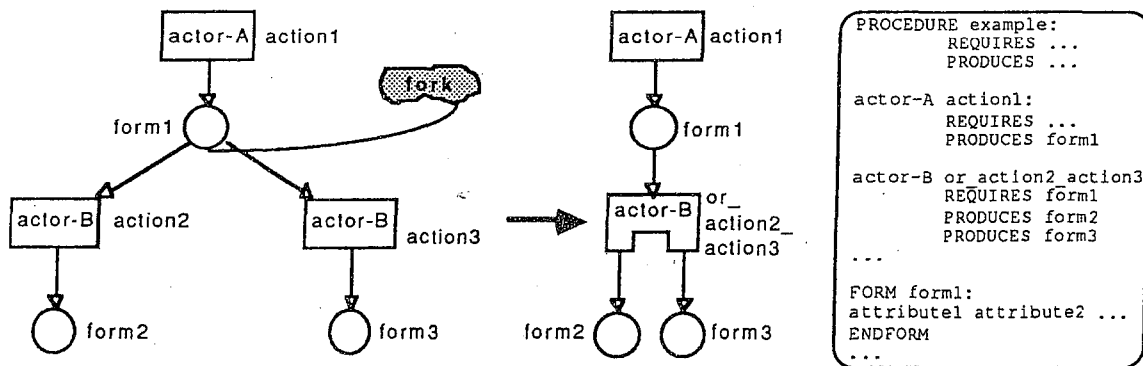


Figure 4: Fork situation, transformation into an alternative action and corresponding CoPlan-S specification

Generation of the procedure

The PROCEDURE entry of the CoPlan-S specification is generated. The start places of the procedure are noted in the REQUIRES attribute and the end places in the PRODUCES attribute. Similarly, the input/output places for every action are specified. The external inputs and outputs of the procedure are listed as comments.

For every place in the net, the knowledge base is queried whether it denotes a form. In this case, the fields of the form are inserted as attributes in the FORM entry corresponding to that place. Otherwise the place is interpreted as a text document. If the knowledge base contains information about this document it is inserted as an attribute of the corresponding TEXT entry. Then a ROLE entry is created for every actor in the net.

The result of applying these transformation rules is a CoPlan-S procedure that may be executed by the DOMINO office procedure system. We should stress that in a real application, the created office procedure should only be viewed as a proposal. Before executing it via DOMINO, discussion between the people involved is absolutely necessary. Here again, VIPS can be helpful in providing editing facilities to modify the plan graphically.

4. Example:

The design of a DOMINO procedure for the business trip domain

Figure 5 shows a plan that has been designed using VIPS. We do not explain this process in this paper, for further details see [ViSo89]. The plan models the application for a business trip. The details of the trip are planned by the travel department, here abbreviated by *td*. The superior (*sup*) can approve the trip or not. If the trip has been approved and the details of the trip have been worked out, the trip can be executed. In the other case, it is not necessary to involve the travel department and the person named *self* receives the message to the effect that his trip has not been approved.

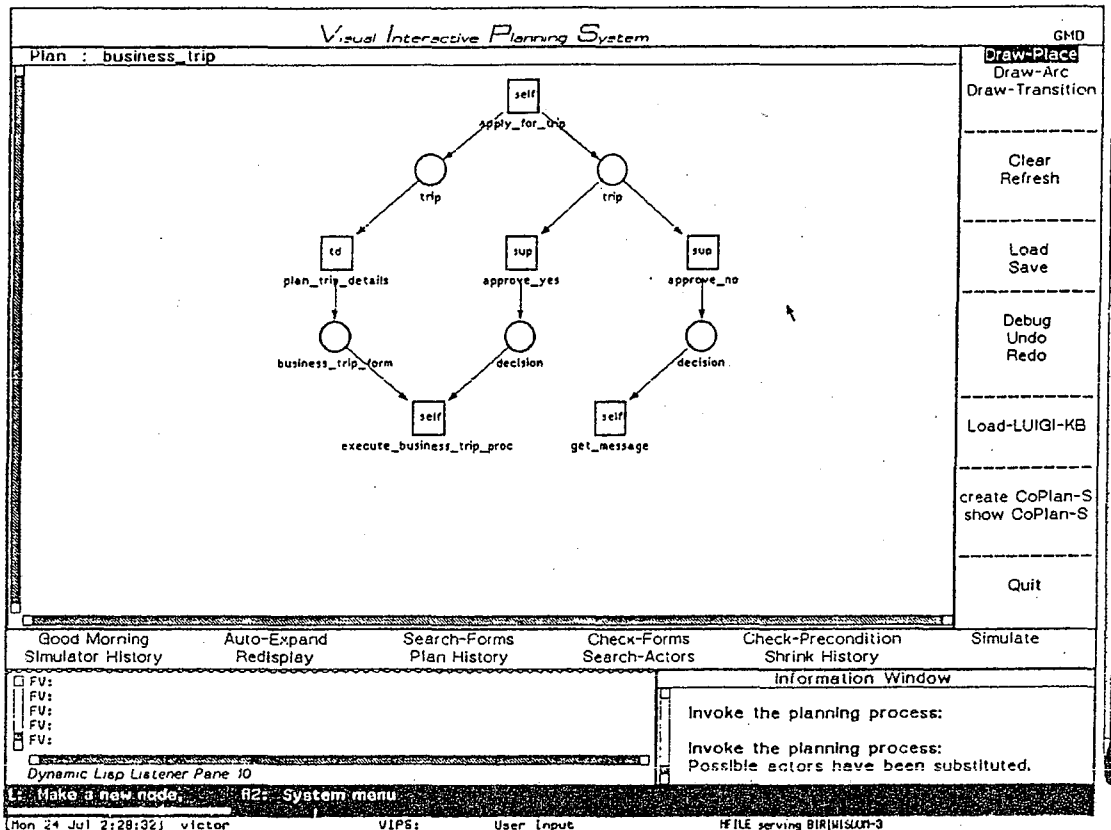


Figure 5: Plan describing the application for a business trip

We assume that the plan has been tested. Now the CoPlan-S generator is invoked via the function *create CoPlan-S*. The steps we have described in section 3.2 are executed. The result is shown in figure 6. First the initiator-actions *apply_for_trip*, *execute_business_trip_proc* and *get_message* are removed. Then the plan is searched for forks. The two actions *approve_yes* and *approve_no* have the same role. Together with the place *trip* they form a fork situation. This is resolved and an alternative action *or_APPROVE_YES_APPROVE_NO* is substituted for the two actions. Next the knowledge base is checked for information about the documents in the plan. The result of this step is the FORM entry *business_trip_form1* listing its attributes.

5. Conclusion and future work

We have presented a component of the planning system VIPS, the CoPlan-S generator, which transforms a plan into a specification that can be executed by the office procedure system DOMINO. The creation of the plan is viewed as a cooperative design process, the transformation into a CoPlan-S specification is done automatically. Graphical representation can be understood more easily and quickly than code in a specification language. Its use facilitates the design of a procedure and the discussion among the departments involved, while the knowledge-based approach ensures consistency with existing regulations in the organization.

In the future we plan to develop an additional component for VIPS which allows the acquisition of new knowledge during the design phase and its consistent integration into the knowledge base. A further aim is to lend a (yet) more intuitive quality to the Petri net formalism, possibly through the use of icons and more flexible knowledge base browsing capabilities.

```

Plan : business_trip

PROCEDURE business_trip:
  REQUIRES trip2 trip1
  PRODUCES decision2 business_trip_form1
  PRODUCES decision1 .

sup or APPROVE_YES_APPROVE_NO:
  REQUIRES trip2
  PRODUCES decision2
  PRODUCES decision1 .

cd plan_trip_details:
  REQUIRES trip1
  PRODUCES business_trip_form1 .

FORM business_trip_form1:
  kind-of-message form-number for destination dates
  ENDFORM

TEXT decision2:
  default text yet to be filled in
  ENDTXT

TEXT decision1:
  default text yet to be filled in
  ENDTXT

TEXT trip2:
  default text yet to be filled in
  ENDTXT

TEXT trip1:
  default text yet to be filled in
  ENDTXT

ROLE id: TRAVEL DEPARTMENT.
Dynamic Window 4

Dynamic Lisp Listener Pane 10

```

Dynamic Lisp Listener Pane 10

A CoPlan-S Office Procedure has been generated. Please inspect file: "LOCAL\VIPS\domino.lisp.1"

(Mon 24 Jul 2:35:47) victor VIPS: User Input FILE saving DIR\DISOUT-3

Figure 6: CoPlan-S specification generated on the basis of the plan in figure 5

Acknowledgements

We would like to thank Thomas Kreifelts for his help with the CoPlan-S generator and Frank von Martial for many helpful ideas.

This work was conducted as part of the joint project WISDOM and was supported by the German Federal Ministry of Research and Technology under grant ITW 8404B and TA Triumph Adler AG, Nuremberg.

References

- [Barb83] G. Barber, "Supporting Organizational Problem Solving with a Work Station", *ACM Transactions on Office Information Systems*, Jan. 1983, 45-67
- [BaJo83] G. Barber, P. de Jong, C. Hewitt, "Semantic Support for Work in Organizations", R.E.A. Mason (ed.), *Information Processing 83*, North-Holland, Amsterdam, 1983
- [BoCh88] J. Bowers, J. Churcher, "Local and Global Structuring of Computer Mediated Communication: Developing Linguistic Perspectives on CSCW in COSMOS", *Proc. of the Conf. on Computer-Supported Cooperative Work*, Portland OR, Sept. 1988
- [Elli83] C.A. Ellis, "Formal and Informal Models of Office Activity", R.E.A. Mason (ed.), *Information Processing 83*, North-Holland, Amsterdam, 1983
- [ElNa87] C.A. Ellis, N. Naffah, "Design of Office Information Systems", *Surveys in Computer Science*, Springer, 1987
- [Kind88] M. Kindler, "User Manual: LUIGI Object System" (in German), WISDOM Research Report FB-TA-88-24, TA Triumph Adler AG, Nuremberg, July 1988
- [KrWo86] Th. Kreifelts, G. Woetzel, "Distribution and Error Handling in an Office Procedure System", *IFIP WG8.4 Working Conf. on Methods and Tools for Office Systems*, Pisa, Italy, Oct. 1986, 33-41
- [KrWo88] Th. Kreifelts, G. Woetzel, "Conversational Systems: A Conceptual Model for Off-Line Group Support Systems", *Proc. of the Australian Computer Conference*, Sydney, Australia, Sept. 1988
- [MaVi87] F. von Martial, F. Victor, "The Electronic Organizational Handbook: Requirements and Specification" (in German), WISDOM Research Report FB-GMD-87-16, Gesellschaft für Mathematik und Datenverarbeitung, St. Augustin, July 1987
- [MaVi88a] F. von Martial, F. Victor, "An Interactive Planner for Open Systems", *Proc. of the Fourth IEEE-Conference on Artificial Intelligence Applications*, San Diego, 1988
- [MaVi88b] F. von Martial, F. Victor, "Collaborative Construction of an Office Knowledge Base: Experiences and Suggestions", *Proc. of the 2nd European Knowledge Acquisition for Knowledge-Based Systems Workshop*, Bonn, June 1988
- [MaVi89] F. von Martial, F. Victor, "Knowledge Acquisition for an Electronic Organizational Handbook", *Proc. of the Fifth Australian Conf. on Applications of Expert Systems*, Sydney, May 1989
- [PaDa89] U. Pankoke-Babatz (ed.), T. Danielsen, A. Patel, P.A. Pays, W. Prinz, R. Speth, "Computer-Based Group Communication: The Amigo Activity Model", Ellis Horwood Limited, Chichester, 1989

- [PrWo88] W. Prinz, M. Weitass, "The Date Planning System - Example of a Cooperative Group Activity", Proc. of the IFIP 6.5 Working Conference on Message Handling Systems and Distributed Applications, Costa Mesa CA, October 1988
- [ViSo89] F. Victor, E. Sommer, F. von Martial, "The Planning Support System VIPS: Creating and Analyzing Office Procedures Using an Electronic Organizational Handbook" (in German), to be published in Conf. Proc. of the GI-89 in: M. Paul (ed.), Reihe "Informatik-Fachberichte", Springer, Heidelberg
- [WiFi86] T. Winograd, F. Flores, "Understanding Computers and Cognition", ALEX Press, Norwood NJ, 1986
- [WoKr87] G. Woetzel, Th. Kreifelts, "The Office Procedure Language CoPlan" (in German), WISDOM Research Report FB-GMD-87-34, Gesellschaft für Mathematik und Datenverarbeitung, St. Augustin, Dec. 1987
- [WoLo86] C.C. Woo, F.H. Lochowsky, "Integrating Procedure Automation and Problem Solving Approaches to Supporting Office Work", IFIP WG8.4 Working Conf. on Methods and Tools for Office Information Systems, Pisa, Italy, Oct. 1986