Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Research Report
RR-92-44

Incorporating Graphics Design and Realization into the Multimodal Presentation System WIP

Thomas Rist, Elisabeth André

September 1992

Deutsches Forschungszentrum für Künstliche Intelligenz

Incorporating Graphics Design and Realization into the Multimodal Presentation System WIP

Thomas Rist, Elisabeth André

DFKI-RR-92-44

Zusammenfassung

In letzter Zeit werden verstärkt Anstrengungen zur Automatisierung der Arbeitsplanung unternommen. Insbesondere sollen wissensbasierte Methoden helfen, bisher offene Probleme zu lösen. So entstanden eine Reihe von Prototypen zur Arbeitsplanerstellung, über die in zahlreichen Veröffentlichungen berichtet wurde.

Nach einer Einführung in die Begriffswelt und Problematik der Arbeitsplanerstellung werden im ersten Teil dieser Arbeit eine Anzahl dieser Systeme untersucht und verglichen. Daraus werden Anforderungen abgeleitet, die ein System zur Arbeitsplanerstellung erfüllen sollte, aber teilweise noch nicht vorhanden sind.

In zweiten Teil wird das Konzept eines Systemes entwickelt, das versucht diesen Anforderungen gerecht zu werden. Eine Anforderung ist die Möglichkeit des Anschlusses externer Programme (wie z. B. CAD-Systemen und Datenbanken) an das Arbeitsplanungssystem.

Die Entwicklung und Implementierung einer Client Server Architektur zur Verwirklichung dieser Schnittstelle

This report has also been published in: Proc. of the International Workshop on 'Advanced Visual Interfaces', Rome, May 27 - 29 1992, World Scientific Press, Singapore.

This work has been supported by a grant from The Federal Ministry for Research and Technology (FKZ ITW-8901 8).

Table of Contents

Incorporating Graphics Design and Realization into the Multimodal Presentation System WIP

Thomas Rist, Elisabeth André

German Research Center for Artificial Intelligence (DFKI)
Stuhlsatzenhausweg 3, W-6600 Saarbrücken 11, Germany
Phone: (+49 681) 302-5252
E-mail: {andre, rist}@dfki.uni-sb.de

Abstract

Recently, there has been increasing interest in the design of user interfaces that take advantage of graphics when presenting information. Since it is impossible to anticipate the needs and requirements of each potential user in an infinite number of presentation situations, it is more reasonable to automatically design graphics on the fly in a context-sensitive way. In this paper, we present components for graphics design and graphics realization as parts of the multimodal presentation system WIP. After a short overview of WIP, we introduce our basic assumptions about how to describe surface aspects and the meaning of complex graphics. We then describe the graphics realization component and sketch the graphics design process. By means of a generation example we show how graphics design is interleaved with graphics realization.

1 Introduction

There is no doubt that graphics have a considerable potential as a way of presenting information and convey certain kinds of information more precisely and effectively than text. Although this seems to be a platitude, in today's man-machine communication, graphics are rarely used and play a subordinate role. Since graphics hard- and software has become more and more sophisticated and affordable, graphic-based communication has received significant interest in research on intelligent user interfaces. However, technology is only one step towards communication using graphics. Another important question is how to design a graphics so that it plays the intended role in a communication. Among other things, one has to design particular graphics for particular purposes and one should tailor the graphics to the individual audience and the presentation situation. Since it would involve immense effort to design and store graphics for each possible combination of relevant design parameters, it is more reasonable to automatically design graphics on the fly in a context-sensitive way.

Previous work on the automatic design of graphics can be distinguished in view of the kind of graphics to be generated and in view of the underlying design methods. The spectrum of graphics ranges from abstract presentation graphics such as pie- and bar charts (cf. [8], [11] and [13]), network diagrams (cf. [14]), symbol-based diagrams, e.g., for the visualization of process information in industrial control (cf. [4]), the presentation of electrical circuits (cf. [7]) and weather maps (cf. [12]), schematic line drawings, e.g., to describe chemical apparatuses (cf. [19]), up to 3D object depictions and environments (cf. [6]) and illustrations of 3D objects (cf. [5] and [17]).

Whereas several approaches rely on a pure selection of pre-defined graphical presentations (cf. [15] and [21]) and thus do not address design issues, others provide techniques in order to select and combine graphical elements. Such "compositional" approaches can be further distinguished in view of the primitives they use. Several approaches rely on predefined icons that are stored in a database, either as bitmaps (e.g., [12] and [19]) or as propositional descriptions (e.g., [6] and [7]). As an alternative, one can follow the approach of the graphics designer Bertin (cf. [3]) and describe a graphics as an implantation of spots (either points, lines or areas) in an empty 2D drawsheet. With respect to perceptible variations of a spot, Bertin distinguishes between eight visual variables (x- and y-position in the plane, size, intensity, pattern, color, direction and shape). A particular piece of information is then encoded by certain variations of visual variables. Although Bertin's view on graphics has proved to be quite useful for the automated design of abstract presentation graphics (cf. Mackinlay's APT system [13]), it is not clear how to transfer this approach to graphics including illustrations of material 3D objects. One may describe the depiction of an object as a configuration of spots together with their specific visual properties, but in general it will be very difficult and costly to make explicit which information is encoded by which variation of spots. E.g., to show an object, one can choose among numerous perspectives. Each choice affects the arrangement of corresponding spots as well as the shape and size of the spots.

Important work on the generation of depictions of 3D objects without relying on predefined icons has been done by Feiner and Seligman (cf. [17]). In their system IBIS, 3D objects are related to illustration objects on the picture level. When generating illustration objects, they consider both the underlying representation of the 3D object in the knowledge base and the purpose for which the illustration will be used. They use a generate and test approach in order to achieve a close relationship between the visual appearance of an object in the world and its appearance in the illustration.

Summing up, it can be said that a common theory of graphical communication has not yet crystallized, neither among the approaches mentioned above nor among the numerous contributions from disparate disciplines, such as graphics design, art history, pedagogy, philosophy, semiotics, and psychology. In this paper, we do not attempt to come up with a new and better theory of graphical communication. Rather, this approach should be understood as a starting point which is of practical use for the automatic synthesis of various kinds of pictures we want to generate in the context of WIP, a multimodal presentation system (cf. [20]).

2 Graphics Generation in WIP

Our goal is to develop a component for graphics generation to be incorporated in the multimodal presentation system WIP. This system has been designed as a unidirectional interface between an application system and a user. The information to be presented by WIP is provided by an application system, which may be a control panel, an expert system or a help system. Considering generation parameters, such as user characteristics and resource limitations, WIP generates multimodal presentations in which several modes are integrated. Currently, we focus on the generation of illustrated instructions explaining how to operate technical devices, such as an espresso machine or a lawn mower. Thus, WIP often has to graphically communicate information about physical objects, i.e., information about object properties (such as shape, material surface characteristics, constituent structure, function etc.), static relations to other objects (such as the location, the orientation and the distance of the objects) and dynamic relations that represent changes of object attributes and static relations (e.g., a change in the spatial position of an object). We presume that WIP has access to application-specific domain knowledge, which comprises an ontology of domain objects and actions, domain plans for problem solving and geometric models of 3D objects and object configurations.

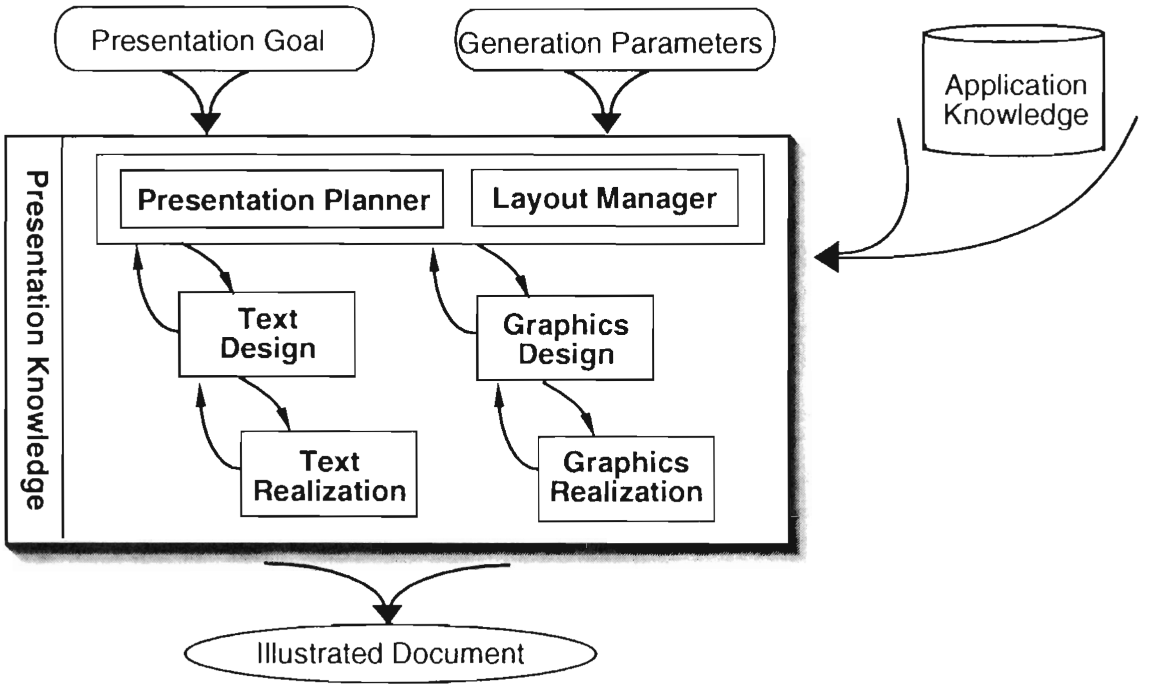

Fig. 1: Overview of the WIP system

The major components of the WIP system are: a presentation planner that is responsible for determining the contents and selecting an appropriate mode combination, mode-specific generators (currently for text and graphics) and a layout manager [9] that arranges the generated output in a document (cf. Fig. 1). Each generator consists of a design and a realization component. As with the text generator (cf. [10]), the graphics generator is driven by the presentation planner. The result of the presentation planning process is a hierarchically structured plan of the document to be generated (cf. Fig. 2). This plan reflects the propositional contents of the potential document parts, the intentional goals behind the parts as well as rhetorical relationships between them, for details see [1]. While the top of the presentation plan is a more or less complex presentation goal (e.g., introducing an object or explaining how to make coffee), the lowest level is formed by specifications of elementary presentation tasks (e.g., formulating a request or depicting an object) that are directly forwarded to the mode-specific design components. Thus, presentation tasks form the input of the graphics design component whereas its output should contain a sequence of instructions to be executed by the graphics realization component.

The WIP environment places certain requirements on the graphics generator.

A distinguishing feature of WIP is that the generators receive their input from the presentation planner in a piecemeal fashion. As soon as the presentation planner has decided which generator should convey a certain piece of information, this piece is forwarded to the respective generator. While the presentation planner selects the next parts to be communicated, the design components of the generators process in parallel previously selected parts and pass their results on to the realization components. This not only saves generation time, it is also necessary since content and mode selection as well as content realization may depend on each other and therefore have to be interleaved (see also [2]). As a consequence, the graphics generator must be able to process new input depending on what has been generated before. Among other things, this includes recognizing whether new information can be incorporated into already designed pictures or not.

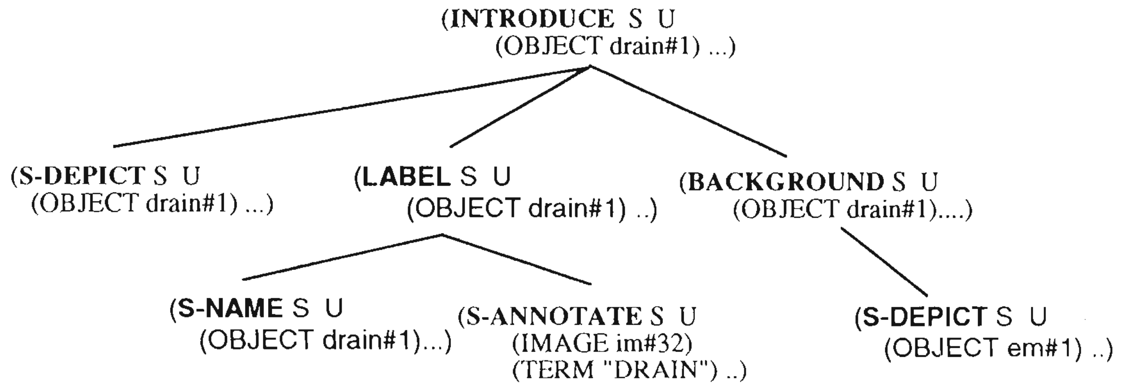

Fig. 2: Fragment of a Presentation Plan

To tailor graphics to document parts in other modalities, the presentation planner has to know how a certain piece of information is conveyed by the graphics generator. Among other things, this is necessary to generate crossmodal references to picture parts, e.g., via natural language or pointing gestures. Therefore, it is insufficient to integrate the graphics generator as a unidirectional front-end that only produces pictures. Instead, the graphics generator also has to provide an explicit description of how it has graphically encoded information in a picture.

Last but not least, the transfer to other applications should be supported. Thus, it should be possible to easily extend WIP's repertoire of graphical techniques.

3 Graphical Encoding of Information

Following a compositional approach to graphics design, we have to find appropriate graphical primitives. On the one hand, they should be elementary enough to allow for assembling various kinds of pictures including illustrations of physical objects. On the other hand, it must be possible to describe the semantic mapping between these primitives and the information they ought to convey.

3.1 Constituents of Graphical Presentations

When analyzing illustrations in printed instructions, we found that most pictures can be considered as an arrangement of constituents that belong to different categories. In our approach, we take the following view. A picture consists of a picture frame and a set of images located within this frame. A picture frame is regarded as a restricted 2D region that has specific attributes, such as shape, size and borders. To handle presentations with insets, a frame may also enclose smaller frames. Frames serve to carry images, i.e., images are added to and positioned and oriented within a frame. Each image is treated as an object that is characterized by a restricted 2D region and a set of attributes including visual properties, such as shape, color/gray-pattern. According to the underlying source from which an image is derived (cf. Fig. 3), we distinguish between several basic image types:

- Images of 2D concepts such as point, line, arrow, rectangle, etc., which are often used in 3D illustrations as metagraphical objects.
  These images are considered as instantiations of generic 2D concepts.
- Images that are created by typesetting strings of characters or symbols

  Images that result from mapping a 3D model of an object or an illustrational scene onto a plane 2D region

An illustrational scene is a spatial arrangement of several objects in a virtual 3D space. For the sake of uniformity, we assume that the depiction of a scene also involves images of all the objects constituting the scene. Consequently, if an object is occluded in a scene, its corresponding image will not be visible in the depiction of the scene. Although one may look at object- or scene depictions as a composition of 2D elements, it is useful to consider them as individual concepts. Our main argument is that in most presentations such depictions are only treated (i.e., perceived, referred to, etc.) as a whole and not in terms of low-level 2D constituents.

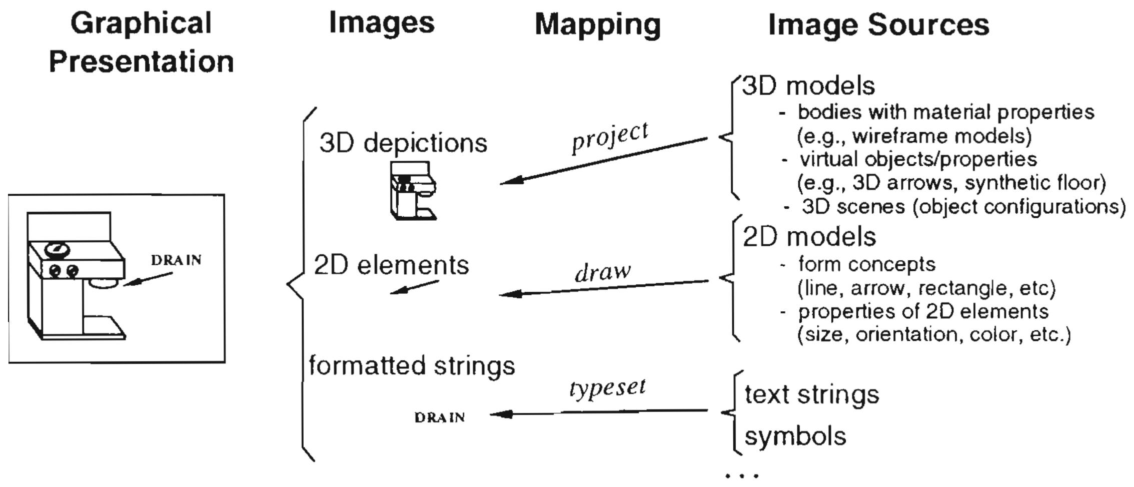
Graphical Presentation
DRAIN
Images
3D depictions
2D elements
formatted strings
DRAIN
Mapping
project
draw
typeset
Image Sources
3D models
- bodies with material properties (e.g., wireframe models)
- virtual objects/properties (e.g., 3D arrows, synthetic floor)
- 3D scenes (object configurations)
2D models
- form concepts (line, arrow, rectangle, etc)
- properties of 2D elements (size, orientation, color, etc.)
text strings
symbols
...

Fig. 3: Picture constituents

To characterize surface aspects of graphical presentations, we have not only to consider images and their attributes, but also various relations that can be identified between images. Indispensible are relations to characterize aspects of the spatial arrangement of images, e.g., to characterize the relative position of an image with respect to one or more other images or to indicate that two images are overlapping. Beside spatial relations, a surface description of a picture may also include relations to express similarities or differences of image properties (e.g., same-color, shape-congruency, etc.). Among other things, such relations are necessary to express the conditions under which two images can be discriminated by human viewers.

3.2 Describing the Meaning of Graphical Presentations

In section 2, we have argued that the graphics designer should explicitly represent how information is encoded in a picture. Inspired by Mackinlay's work (cf. [13]), we use a relation tuple of the form:

(Encodes <graphical-means> <information>).

to specify the semantic relationship between graphical means, i.e., picture constituents and relations between constituents, and the information they are to convey. Since in most cases a picture is composed of several constituents and conveys a collection of information, the semantics of a picture is described by a conjunction of tuples of the Encodes relation. E.g., the semantic description of the picture shown in Fig. 3 could include the following tuples:

(Encodes image#21 espressomachine#1)
(Encodes image#29 drain#5)
(Encodes 2D-rel-pos#24 3D-rel-pos#8)
(Encodes arrow-annotation#41 labeling-relation#2)
...

where image#21 and image#29 are depictions of espressomachine#1 and drain#5, 2D-rel-pos#24 is the relative position of the drain-image and the espressomachine-image whereas 3D-rel-pos#8 is the relative position of the drain and the espressomachine in the domain model. Arrow-annotation#41 represents the particular spatial arrangement of the 2D arrow, the drain-image and the string "DRAIN", and labeling-relation#2 represents a particular rhetorical relation between the string "DRAIN" and drain#5.

We can even go further and not only specify individual encoding relations between individual objects as above, but also specify encoding relations on a generic level. E.g., the relation tuple below expresses the fact that the object property of "being faulty" is encoded by the image property of "being red":

(Encodes λ(img).red(img) λ(obj).faulty(obj)).

In order to understand what a picture represents, a viewer has to recognize whether or not picture constituents and relations between them are meant to encode relevant information, and he has to know how to "read" these encodings. E.g., when color is used to indicate that an object has a certain non-visual property, say "red" for the property of "being faulty" as above, "red" must not be interpreted as object color. In general, not all object properties are encoded in a picture, e.g., a black and white picture normally provides no information about the color of an object. Vice versa, there may be picture properties that are not meant to encode specific object properties. E.g., each image has a particular size, but this size is often irrelevant since it is not meant to encode the absolute size of the corresponding object. Note that this view of graphically encoding properties of objects also fits into the more general framework of model theory (cf. [18]).

When looking at graphics, e.g., in textbooks or instruction manuals, one will realize that almost everything can be graphically presented in various ways. Although human designers and illustrators share conventions and fall back on approved techniques, they often make variations and even come up with new presentation forms. However, in case a picture is to depict a relation between world objects, two basic assumptions are made in WIP: a) For each object that occurs as an argument of the relation to be depicted the picture must contain at least one image as a representative. b) There must be an encoding of the relation-type, e.g., a spatial relation between the depictions of the arguments.

4 Realization of Graphical Encodings

In order to be able to synthesize pictures that satisfy a specified set of encoding relations, we have developed a graphics realization component. The major modules of the realization component are: a 3D studio, a mapping controller, a 2D clipboard and a module for handling data structures for images and pictures (cf. Fig. 4).The operators handled by the component fall into three classes:

- *Studio operators* to create and manipulate wireframe models of 3D objects. Examples are: adding an object to an illustrational scene, spatially separating object parts to construct exploded views and cutting away object faces to make obscured parts visible.
- *Mapping operators* that constrain projection parameters and project wireframe models onto images. E.g., we have the possibility to map models onto schematic line drawings or to produce more realistic looking depictions using rendering techniques.
- *Clipboard operators* that are defined on the picture level. E.g., annotating an object depiction with a text label, or scaling/framing/coloring picture parts.

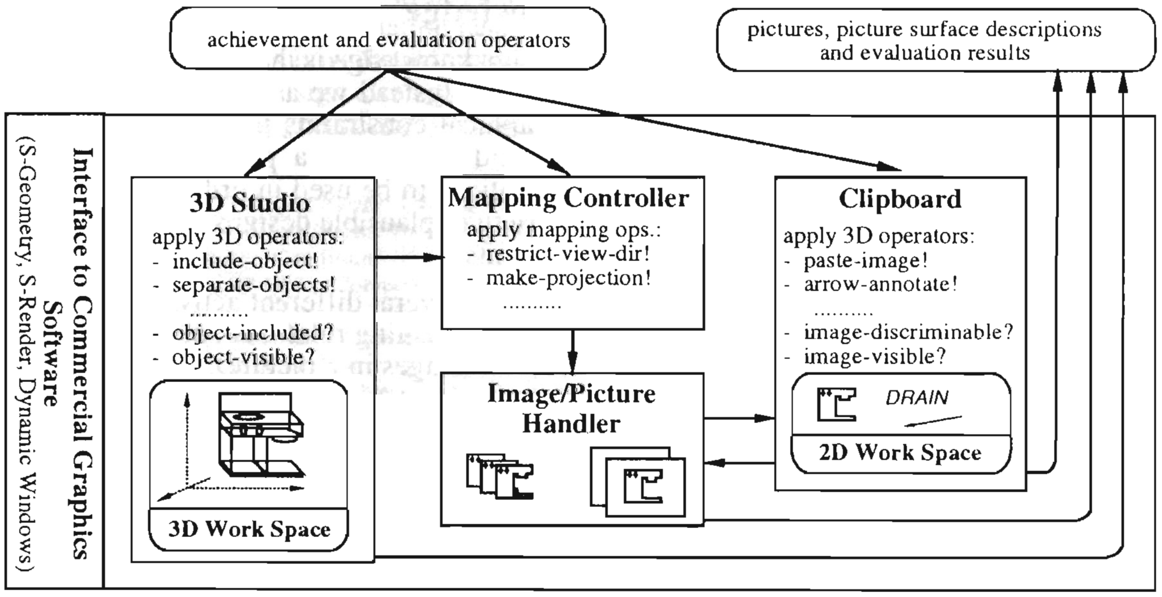

Fig. 4: Architecture of the Realization Component

Unfortunately, these *achievement operators* which produce effects either on models, mappings or pictures may have side effects which are hard to anticipate in advance. E.g., we know that the visibility of scene objects may be affected by adding further objects to the scene, but it is hard to predict how this influence will work out in a particular situation. For this reason, the functionality of the realization component not only encompasses achievement operators, but also *evaluation* operators[1] (e.g., for checking whether an object as part of an object configuration is visible from a given viewing specification, or for checking whether a picture part can be discriminated from other picture components). These evaluation operators serve to check whether graphical constraints have already been satisfied and whether they are still satisfied after having applied further achievement operators.

Summing up, the realization component can be compared with an object-oriented graphics editor providing various operations for composing graphical presentations. Its distinguishing features are that these presentations may include automatically designed images of 3D objects and that several evaluation procedures are supported. Furthermore, since the component is not driven by a human user, but by WIP's graphics design component, it provides a programmable interface that follows the paradigm of object-oriented programming.

5 Automating Graphics Design

The task of the graphics design component is to transform presentation tasks received from the presentation planner into a sequence of instructions which can be executed by the graphics realization component. For this transformation process, a plan-based approach has proven adequate. In the following, we describe how we represent design knowledge and how the graphics designer uses this knowledge to build up graphical presentations.

5.1 Representation of Design Knowledge

A basic idea underlying our representation of design knowledge is that we do not directly relate presentation tasks to graphical presentations. Instead we associate presentation tasks with a set of graphical constraints.[2] Graphical constraints place restricions on image sources (e.g. 3D models), mappings and images in a picture. Thus, they eventually constrain the set of graphical presentations to be used in order to achieve a presentation task. This enables us to cover a variety of plausible designs for one and the same presentation task with a single set of constraints.

Taking into account that graphics design comprises several different activities, such as creating entities (e.g., images and pictures) and establishing relations between entities (e.g., arranging 3D models in a scene, or arranging images in a picture), we distinguish between several types of graphical constraints:

> *Assignment*
> (:= <var> <term>)
> The value of the variable <var> must be the value of the term <term>.

[1] Achievement operators and evaluation operators are comparable to *style methods* and *style evaluators* in the IBIS system (cf. [17]).

[2] The so-called *style strategies* in the IBIS approach (cf. [17]) can also be regarded as a kind of constraints.

Relation
(<rel> $<arg>_1$ ... $<arg>_n$)
The relation <rel> must hold between the arguments $<arg>_1$ to $<arg>_n$

Weighted Relation
((<w-rel> $<arg>_1$ ... $<arg>_n$) <d>)
The weighted relation <w-rel> between the arguments $<arg>_1$ to $<arg>_n$ must be fullfilled at least to the degree[3] <d>. We presume that a weighted relation takes values from the interval [0 ... 1].

Compound
(FORALL ($<var>_1$ ... $<var>_k$) <cond> $<gc>_1$ ... $<gc>_n$)
The graphical constraints $<gc>_1$ ... $<gc>_n$ must be satisfied for all assignments of the variables $<var>_1$ ... $<var>_k$ satisfying the condition <cond>.

With each graphical constraint, we associate means to be used in order to evaluate whether it is fullfilled and means to be used in order to achieve them. Whereas the evaluation operators introduced in section 4, provide the basis for constraint evaluation, the achievement operators of the realization component are eventually the means to fullfill a graphical constraint. However, achieving a graphical constraint can be a complex task itself. Thus, if we want to keep the amount of design knowledge hidden in the procedurally defined achievement operators as low as possible and if we want to avoid redundant coding of design knowledge, we need a representation that allows for decompositional descriptions.

To bridge between presentation tasks, graphical constraints and realization operators, we use so-called *design strategies* . Each design strategy consists of a header, an applicability condition and a body. The header may be a presentation task or a graphical constraint. The applicability condition specifies when a strategy may be used and constrains the variables to be instantiated. The body contains a set of graphical constraints that have to be achieved in order to accomplish the goal indicated in the header.

Two examples of design strategies are shown below. The header of these design strategies refers to a presentation task. The partial picture description (Partial-Pic-Des) in the header indicates how the information to be communicated will be encoded. After the instantiation of a design strategy, this information can be accessed by the presentation planner, e.g., when generating references to parts of a picture.

```
[S1]  Header:
             (S-Depict System User (Object ?object)
                        (Partial-Pic-Des ((?picture ((Encodes ?image ?object)))))
      Applicability Condition:
             (3D-Object ?object)
      Body:
             (S-Includes ?object ?scene)
             (Type-of ?mapping 3D-Projection)
             ((S-Visible ?object ?scene ?mapping) 0.8)
             (:= ?image (Image-of ?object ?scene ?mapping))
             (P-Includes ?image ?picture)
             ((Discriminable ?image ?picture) 1)
             ((P-Visible ?image ?picture) 1)
```

3 The degree of a weighted-relation constraint corresponds to the achievement threshold of style strategies in the IBIS system (cf. [17]).

The strategy [S1] serves to depict a 3D object. This is accomplished by first applying operators of the 3D studio, namely including a model of a 3D object in a scene and ensuring that most[4] of the object is visible using a certain mapping function that must be a 3D projection. Then, the object depiction is instantiated by applying the mapping function. Finally, the object depiction is pasted into a picture, where it must be discriminable from other images. In addition, it must not be occluded by other images.

Strategy [S2] is to show the position of a 3D object ?x in an assembly ?group by spatially isolating this object from the assembly. In the underlying knowledge base, assemblies are modelled as tree structures. In these trees, a 1:n father-son relationship reflects the relationship between a base part and n attached parts. The strategy is applicable if object ?x is attached to another object ?y of the assembly. In the body, ?x is first separated from ?y and all objects of the group which are attached to ?x are then separated from ?x.

```
[S2]  Header:
              (S-Depict System User (Assembly-Position ?x ?group ?a-pos)
                      (Partial-Pic-Des ((?picture ((Encodes ?p-x ?x)
                                                   (Encodes ?p-group ?group)
                                                   (Encodes ?p-rel-pos ?a-pos))))))
      Applicability Condition:
              (And (Attached ?x ?y) (Part-of ?x ?group) (Part-of ?y ?group) (Assembly ?group))
      Body:
              (S-Includes ?group ?scene)
              (Type-of ?mapping 3D-Projection)
              ((S-Visible ?group ?scene ?mapping) 0.8)
              (Sep-Axis ?x ?y ?xy-axis)
              (Separate ?x ?y ?xy-axis)
              (:= ?view-dir (View-Dir ?scene))
              (View-Dir-Res ?view-dir (None-of (?xy-axis)))
              (For-all (?z) with (And (Attached ?z ?x) (Sep-Axis ?z ?x ?zx-axis))
                      (Separate ?z ?x ?zx-axis)
                      (View-Dir-Res ?view-dir (None-of (?zx-axis)))
              ((S-Visible ?x ?scene ?mapping) 1)
              (:= ?p-group (Image-of ?group ?scene ?mapping))
              (:= ?p-x (Image-of ?x ?scene ?mapping))
              (P-Includes ?p-group ?picture)
              ((Discriminable ?p-x ?picture) 1)
              ((Discriminable ?p-group ?picture) 1)
              ((P-Visible ?p-x ?picture) 1)
              (:= ?p-rel-pos (P-Relpos ?p-x ?p-group ))
```

To enable the graphics designer to flexibly combine design strategies, they should only embody a minimal set of graphical constraints associated with a given presentation task. E.g., since the presentation task in strategy [S2] does not prescribe how to encode particular object attributes (such as shape, size or surface structure) there are no corresponding graphical constraints included in the body. In some situations the set of graphical constraints put on a picture may be increased by further presentation tasks. In situations where several choices are left, the graphics designer uses heuristics to make necessary decisions. E.g., to choose among a set of possible view directions.

[4] An object is assumed to be fully (100%) visible if it is not clipped and if it is not obscured by other scene objects. In the example, "most" means that the object is visible to at least 80%. However, such thresholds have to be empirically validated.

5.2 Outline of the Graphics Design Process

Given a presentation task, the graphics designer has to find design strategies whose header matches the presentation task and checks for which variable bindings their applicability conditions hold. After the selection of a design strategy, the graphical constraints in the body of this strategy can be processed. If several strategies apply, a priority order has to be found employing selection heuristics.

When processing a graphical constraint, the graphics designer first checks whether it is already able to evaluate it. E.g., it cannot evaluate a predicate on the picture level, such as discriminable, if the corresponding mapping operations have not yet been carried out. In case a predicate is evaluable, the graphics designer constrains the possible values of the variables by applying the corresponding evaluation operator. If a predicate is not satisfied, the graphics designer attempts to achieve it by further expansion or by making the realization component apply an achievement operator. If no further operators or design strategies can be applied, the strategy fails. Fig. 5 gives an overview of the refinement process.

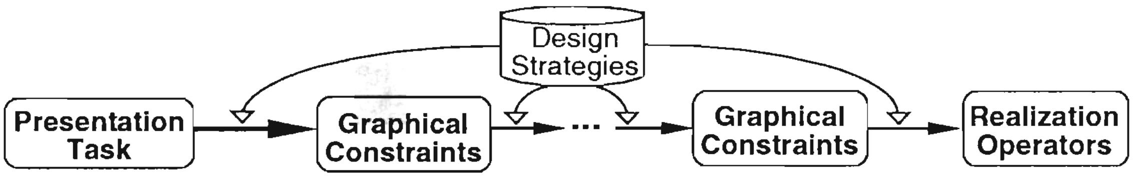

Fig. 5: Refinement Process

During the generation of a graphics, it may happen that already achieved goals are destroyed by later operations. Thus, after applying an achievement operator WIP's graphics designer has to check whether previously satisfied constraints are still fulfilled.

As mentioned in section 2, WIP's graphics designer receives new presentation tasks from the presentation planner in a piecemeal fashion. Here, the problem arises at which time the graphics designer should pass planned achievement operators on the realization component for execution. On the one hand, the graphics designer should process the input specification provided by the presentation planner immediately in order to make its results available to other system components as soon as possible. On the other hand, it does not make much sense to map a scene onto a new 2D depiction even though modifications of the scene are expected since this will waste computation time. Currently, the following strategy is used. Scene operators are applied as soon as they have been selected. Mapping operators and picture operations are only executed if a new scene has to be created or no further scene operations are expected. To decide on this, the graphics designer currently uses a rather simple criterion. It only checks whether new presentation tasks have been sent.

The graphics design process is triggered as soon as a presentation task has been written into its task queue. The graphics generation proceeds until all presentation tasks have been expanded, no further entries in the body of a design strategy have to be processed and all achievement operators have been executed.

5.3 Generation Example

In the following, we show by example how the graphics designer builds up a presentation starting from presentation tasks which it receives in a consecutive manner.

Let's assume the presentation planner has decided to show mower#1 and passes (S-Depict System User (Object mower#1) (Partial-Pic-Des ?part-pic-des)) on to the graphics designer. A strategy for achieving this goal is strategy [S1] (cf. section 5.1). Suppose the graphics designer selects this strategy and starts processing the entries in the body. (S-Includes mower#1 ?scene) is associated with an achievement operator of the 3D studio. Its application leads to the creation of a new illustrational scene scene#1 including the wire-frame model of the object mower#1. ((S-Visible mower#1 scene#1 ?mapping) 0.8) is a graphical constraint that may constrain, e.g., the perspective from which an object is shown. In our example, this is not the case since scene#1 contains only mower#1 at this time. The graphics designer adds this constraint to a list of already satisfied constraints. Fig. 6a shows a snapshot of the 3D studio after scene#1 has been created and the wireframe model of mower#1 has been added to the scene. Now let's assume that in the meantime the presentation planner has forwarded a second presentation task on to the graphics designer, namely (S-Depict System User (Assembly-Position ball-bearing#15 wheel-group#3 a-pos#123) (Partial-Pic-Des ?part-pic-des)). The consequence is that the graphics designer does not pass the next four operators of the selected design strategy on to the realization component because it assumes that further operations on the scene will be necessary. Instead, these operators will be registered as not yet processed entries. For the second presentation task, the graphics designer selects design strategy [S2] (cf. section 5.1). It first tries to instantiate ?scene with scene#1. When applying the evaluation operator for (S-Includes wheel-group#3 scene#1), the graphics designer finds out that wheel-group#3 has not to be added to scene#1 because it is part of mower#1. However, the

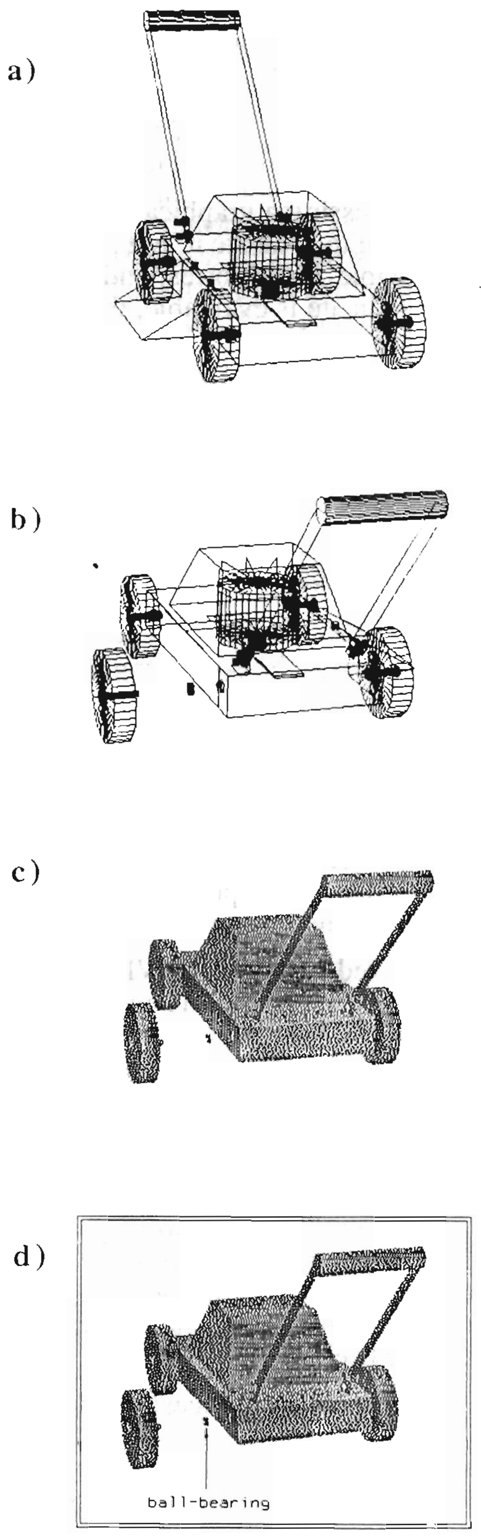

Fig. 6: Snapshots of the Generation Process

graphics designer has to check whether it is visible in the scene. In the example, it decides to constrain the view direction from which mower#1 can be shown in such a way that both mower#1 and wheel-group#3 are visible. The next operations serve to isolate ball-bearing#15 from other parts of wheel-group#3 to show where it is located in the assembly. This is done by separating parts of wheel-group#3 along a separation axis (cf. Fig. 6b). The operator (View-Dir-Res ?view-dir (None-of (?...-axis)) ensures that the view direction does not coincide with separation axes. This again leads to a further restriction of the view direction (cf. Fig. 6b). After each operation, the graphics designer has to check whether already satisfied constraints are still fulfilled, in this case ((S-Visible mower#1 scene#1 mapping#1) 0.8) and ((S-Visible wheel-group#3 scene#1 mapping#1) 0.8).

Let's assume for the purpose of this example that the graphics designer now expects no further scene operations and starts executing the mapping and picture operations. So far, only the projection type of the mapping function has been restricted. However, to do the mapping, a concrete mapping function has to be determined; e.g., by using default values for missing parameters. In the example, scene#1 is mapped onto the depiction image#1 using surface-rendering (cf. Fig. 6c). After that, a new picture picture#1 is created and image#1 is pasted into the picture. Note that since scene#1 only contains mower#1, the depiction of scene#1 coincides with the depiction of mower#1. After having executed the picture operators, the graphics designer checks whether all picture constraints are satisfied, i.e., whether the depictions of mower#1, wheel-group#3 and ball-bearing#15 are discriminable and visible.

In the meantime, the presentation planner has sent a new presentation task, namely (S-Annotate System User image#3 "ball-bearing" picture#1) where image#3 is the depiction of ball-bearing#15. Expanding this presentation task leads only to operations on the picture, namely inserting the image of text string ("ball-bearing"), connecting it to image#3 by means of an arrow image and checking whether already met constraints are still satisfied. The final picture is shown in Fig. 6d.

6 Conclusion and Outlook

In this paper, we have sketched a graphics generator as part of a multimodal presentation system. The generation of a graphics was split into a design and a realization part. In the design part, design strategies are used to relate presentation tasks to graphical constraints. Some of these graphical constaints are directly related to achievement operators to be applied by the realization part, others lead to the application of further design strategies. In addition, graphical constraints are associated with evaluation operators. They serve to check whether constraints are already satisfied and whether they are still satisfied after having applied further achievement operators. Since in most cases design decisions depend on whether a graphics meeting certain graphical constraints can be realized or not, the processes for graphics design and realization are interleaved. A distinguishing feature of the realization part is that we not only cope with 2D concepts, but also handle 3D models of objects and object configurations. These 3D models provide an important source for generating object depictions which in turn can be used to build up complex graphical presentations.

Future work will concentrate on both the design and realization component. First of all, we have to refine and add criteria for making design decisions. E.g., selection criteria are needed for finding a priority order if several design strategies apply or for completely specifying the arguments of achievement operators if several possibilities are left after the expansion of design strategies. Currently, selection heuristics have been only formulated for subproblems, e.g., for choosing an appropriate perspective (cf. [16]). The capacity of WIP's design process is also limited by the fact that design strategies embody more or less pre-compiled knowledge of how to encode a certain piece of information or how to satisfy graphical constraints. The set of graphical constraints and thus the sequence of operators to be applied by the realization component are found by means of a top-down refinement strategy with backtracking. As an alternative, one could also use bottom-up strategies. I.e., the composition of a graphics will be driven by encodings of elementary information units. Since in a presentation system a graphics design process is worth nothing without being able to produce the corresponding graphics, we also have to improve the functionality of the realization component. This comprises the improvement of the current operators which rely on various simplifications as well as the definition of further ones.

7 Implementation

The graphics designer, like the entire WIP system, is implemented in Symbolics Common Lisp under Genera 8.0 on a XL1200 and several MacIvory workstations. The design process takes advantage of software components we have already implemented for WIP's presentation planner, e.g., techniques for top-down planning and constraint processing. The realization component utilizes commercial software packages. The 3D studio and the mapping controller rely on the Symbolics interactive 3D editor S-Geometry. The clipboard uses the 2D graphics facilities of the Symbolics window system. Currently, we have no color option available and rendering is time consuming since it is not supported by our hardware. Representations of domain objects comprise wire-frame models which are based on the modelling primitives provided by S-Geometry. The quality of generated object depictions could, however, be improved by spending more effort on the specifications of their wire-frame models.

Acknowledgement

This work is supported by the German Ministry of Research and Technology under grant ITW8901 8. Thanks to Wolfgang Wahlster for his comments on an earlier version of this paper and to Andreas Butz, Bernd Hermann, Antonio Krüger, Daniel Kudenko, Georg Schneider, Frank Schneiderlöchner, Christoph Schommer and Detlev Zimmermann for implementing parts of the realization component.

References

[1] E. André, and T. Rist. Generating Illustrated Documents: A Plan-Based Approach. In: *InfoJapan 90*, Vol. 2, 163-170, 1990.

[2] E. André, and T. Rist. The Design of Illustrated Documents as a Planning Task. Submitted to AAAI Press.

[3] J. Bertin. Semiology of Graphics: Diagrams, Networks, Maps, University of Wisconsin Press, translated by W. J. Berg, 1983.

[4] P. Elzer, H. Siebert, and K. Zinser. New Possibilities for the Presentation of Process Information in Industrial Control, In: *Proc. of the 3rd IFAC/IFIP/IEA/IFORS Conference on Man-Machine Systems, Analysis, Design and Evaluation.* Oulu, Finland, Pergamon Press, 1988.

[5] S. Feiner. APEX: An Experiment in the Automated Creation of Pictorial Explanations. In: *IEEE Computer Graphics and Applications* **5(11)**, pp. 117-123, 1985.

[6] M. Friedell. Automatic Synthesis of Graphical Object Descriptions. In: *Computer Graphics (ACM)* **18(3)**, pp. 53-62, 1984.

[7] J. Geller, and C. Shapiro. Graphical Deep Knowledge for Intelligent Machine Drafting. In: *Proc. of the 10th IJCAI,* pp. 545-551, 1987.

[8] S. Gnanamgari. Information Presentation Through Default Displays. PhD thesis, University of Pennsylvania, 1981.

[9] W. Graf, and W. Maaß. Constraint-basierte Verarbeitung graphischen Wissens. In: W. Brauer and D. Hernández (eds.). *Verteilte Künstliche Intelligenz und kooperatives Arbeiten*, Proc. 4. Internationaler GI-Kongreß Wissensbasierte Systeme, München, 1991.

[10] K. Harbusch, W. Finkler, and A. Schauder. Incremental Syntax Generation with Tree Adjoining Grammars. In: W. Brauer and D. Hernández (eds.). *Verteilte Künstliche Intelligenz und kooperatives Arbeiten*, Proc. 4. Internationaler GI-Kongreß Wissensbasierte Systeme, München, 1991.

[11] K. Kansy. Leitbeispiel Graphikdesigner. Tasso Report Nr. 14, GMD, St. Augustin, Germany, 1990.

[12] S.M. Kerpedjiev, Automatic Generation of Multimodal Weather Reports from Datasets, in: *Proceeedings ANLP-92*, Trento, Italy, 1992.

[13] J. Mackinlay. Automating the Design of Graphical Presentations of Relational Information. In: *ACM Transactions on Graphics* **5(2)**, pp. 110-141, 1986.

[14] J. Marks. The Competence of an Automated Graphic Designer. In: *Proc. of the 1991 Long Island Conference on Artificial Intelligence and Computer Graphics*, NYIT, New York, pp. 53-61, 1991.

[15] P. Mertens. Expertisesysteme als Erscheinungsform der betrieblichen Expertensysteme. Arbeitspapiere Informatik-Forschungsgruppe VIII der Friedrich-Alexander-Universität Erlangen-Nürnberg, 1988.

[16] T. Rist, and E. André. Wissensbasierte Perspektivenwahl für die automatische Erzeugung von 3D-Objektdarstellungen. In: Klaus Kansy and Peter Wißkirchen (eds.). *Graphik und KI.* IFB 239, Berlin: Springer-Verlag, pp. 48-57, 1990.

[17] D.D. Seligmann, and S. Feiner. Automated Generation of Intent-Based 3D Illustrations. In: *Computer Graphics* **25(3)**, 1991.

[18] H. Stachowiak. Allgemeine Modelltheorie. Wien: Springer-Verlag, 1973.

[19] T. Strothotte. Pictures in Advice-Giving Dialog Systems: From Knowledge Representation to the User Interface. In: *Proc. of Graphics Interface 89*, pp. 94-99, 1989.

[20] W. Wahlster, E. André, S. Bandyopadhyay, W. Graf, and T. Rist. WIP: The Coordinated Generation of Multimodal Presentations from a Common Representation. In: Andrew Ortony, John Slack and Oliviero Stock (eds.). *Computational Theories of Communication and their Applications.* Heidelberg: Springer-Verlag, 1992.

[21] F. Zdybel, N. Greenfeld, and M. Yonke. An Information Presentation System. In: *Proc. of the 7th IJCAI,* pp. 978-984, 1981.

RR-92-34
Philipp Hanschke:
Terminological Reasoning and Partial Inductive Definitions
23 pages

RR-92-35
Manfred Meyer:
Using Hierarchical Constraint Satisfaction for Lathe-Tool Selection in a CIM Environment
18 pages

RR-92-36
Franz Baader, Philipp Hanschke:
Extensions of Concept Languages for a Mechanical Engineering Application
15 pages

RR-92-37
Philipp Hanschke:
Specifying Role Interaction in Concept Languages
26 pages

RR-92-38
Philipp Hanschke, Manfred Meyer:
An Alternative to Θ-Subsumption Based on Terminological Reasoning
9 pages

RR-92-43
Christoph Klauck, Jakob Mauss: A Heuristic driven Parser for Attributed Node Labeled Graph Grammars and its Application to Feature Recognition in CIM
17 pages

RR-92-44
Thomas Rist, Elisabeth André: Incorporating Graphics Design and Realization into the Multimodal Presentation System WIP
15 pages

RR-92-45
Elisabeth André, Thomas Rist: The Design of Illustrated Documents as a Planning Task
21 pages

RR-92-46
Elisabeth André, Wolfgang Finkler, Winfried Graf, Thomas Rist, Anne Schauder, Wolfgang Wahlster:
WIP: The Automatic Synthesis of Multimodal Presentations
19 pages

DFKI Technical Memos

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

TM-92-02
Achim Schupeta: Organizing Communication and Introspection in a Multi-Agent Blocksworld
32 pages

TM-92-03
Mona Singh
A Cognitiv Analysis of Event Structure
21 pages

TM-92-04
Jürgen Müller, Jörg Müller, Markus Pischel, Ralf Scheidhauer:
On the Representation of Temporal Knowledge
61 pages

TM-92-05
Franz Schmalhofer, Christoph Globig, Jörg Thoben
The refitting of plans by a human expert
10 pages

TM-92-06
Otto Kühn, Franz Schmalhofer: Hierarchical skeletal plan refinement: Task- and inference structures
14 pages

TM-92-08
Anne Kilger: Realization of Tree Adjoining Grammars with Unification
27pages

Zusammenfassung

In letzter Zeit werden verstärkt Anstrengungen zur Automatisierung der Arbeitsplanung unternommen. Insbesondere sollen wissensbasierte Methoden helfen, bisher offene Probleme zu lösen. So entstanden eine Reihe von Prototypen zur Arbeitsplanerstellung, über die in zahlreichen Veröffentlichungen berichtet wurde.

Nach einer Einführung in die Begriffswelt und Problematik der Arbeitsplanerstellung werden im ersten Teil dieser Arbeit eine Anzahl dieser Systeme untersucht und verglichen. Daraus werden Anforderungen abgeleitet, die ein System zur Arbeitsplanerstellung erfüllen sollte, aber teilweise noch nicht vorhanden sind.

In zweiten Teil wird das Konzept eines Systemes entwickelt, das versucht diesen Anforderungen gerecht zu werden. Eine Anforderung ist die Möglichkeit des Anschlusses externer Programme (wie z. B. CAD-Systemen und Datenbanken) an das Arbeitsplanungssystem.

Die Entwicklung und Implementierung einer Client Server Architektur zur Verwirklichung dieser Schnittstelle

Zusammenfassung

In letzter Zeit werden verstärkt Anstrengungen zur Automatisierung der Arbeitsplanung unternommen. Insbesondere sollen wissensbasierte Methoden helfen, bisher offene Probleme zu lösen. So entstanden eine Reihe von Prototypen zur Arbeitsplanerstellung, über die in zahlreichen Veröffentlichungen berichtet wurde.

Nach einer Einführung in die Begriffswelt und Problematik der Arbeitsplanerstellung werden im ersten Teil dieser Arbeit eine Anzahl dieser Systeme untersucht und verglichen. Daraus werden Anforderungen abgeleitet, die ein System zur Arbeitsplanerstellung erfüllen sollte, aber teilweise noch nicht vorhanden sind.

In zweiten Teil wird das Konzept eines Systemes entwickelt, das versucht diesen Anforderungen gerecht zu werden. Eine Anforderung ist die Möglichkeit des Anschlusses externer Programme (wie z. B. CAD-Systemen und Datenbanken) an das Arbeitsplanungssystem.

Die Entwicklung und Implementierung einer Client Server Architektur zur Verwirklichung dieser Schnittstelle

Incorporating Graphics Design and Realization into the Multimodal Presentation System WIP

RR-92-44
Research Report

Thomas Rist, Elisabeth André