Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Technical Memo
TM-91-03

Clamping, COKAM, KADS, and OMOS: The Construction and Operationalization of a KADS Conceptual Model

Otto Kühn, Marc Linster, Gabriele Schmidt

February 1991

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

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

Stuhlsatzenhausweg 3
D-6600 Saarbrücken 11
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 by the shareholder companies ADV/Orga, AEG, IBM, Insiders, Fraunhofer Gesellschaft, GMD, Krupp-Atlas, Mannesmann-Kienzle, Nixdorf, Philips and Siemens. 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

Clamping, COKAM, KADS, and OMOS: The Construction and Operationalization of a KADS Conceptual Model

Otto Kühn, Marc Linster, Gabriele Schmidt

DFKI-TM-91-03

Submitted to EKAW91: 5th European Knowledge Acquisition for Knowledge-Based Systems Workshop.

CONTENTS

Clamping, COKAM, KADS, and OMOS:
The Construction and Operationalization of a KADS Conceptual Model

Otto Kühn*, Marc Linster+ and Gabriele Schmidt*

*German Research Center for Artificial Intelligence
PO Box 2080
D-6750 Kaiserslautern
e-mail: kuehn,schmidt@dfki.uni-kl.de

+GMD
Expert System Research Group
PO Box 1240
D-5205 St. Augustin 1
e-mail: linster@gmdzi.gmd.de

Abstract. For a simplified version of the clamping tool selection problem in mechanical engineering, the knowledge acquisition tool COKAM is applied to obtain an informal knowledge base and explanation structures from technical documents and previously solved cases. The output of COKAM is used to construct a three layered KADS conceptual model, which is then transformed into an operational model in the language OMOS. The OMOS formalization allows to verify the informal KADS conceptual model and to check the completeness of the domain knowledge. The results of this analysis are utilized in the next knowledge elicitation session with COKAM.

1. MOTIVATION

This paper describes how three tools and approaches, each representing a different contribution to knowledge acquisition, have been used in one application to obtain qualitatively new knowledge acquisition potential. COKAM (Schmidt & Schmalhofer 1990) is a knowledge elicitation tool. KADS (Breuker & Wielinga 1989) is a systematic knowledge engineering approach. OMOS (Linster 1990) is a representation language for operational models of problem-solving. The interaction of the tools and approaches (see figure 1.1) looks as follows:

- The output of the knowledge elicitation tool COKAM, which produces informal knowledge units and explanation structures from technical documents and from prior case solutions can be transformed into a KADS conceptual model.
- The KADS conceptual model serves as an explanation framework in the ensuing analysis of COKAM-generated explanations. Thus the conceptual model is tested against the next cases. This validates the model in the continuing elicitation.
- The structure-preserving operationalization of the KADS conceptual model in the representation language OMOS enforces an unambiguous definition of the knowledge of the conceptual model. A reduction of ambiguity, vagueness and a better understanding of concepts can be achieved that way (Akkermanns, Balder, van Harmelen, Schreiber & Wielinga 1990, p. 9).
- The cases, that were the basis for the COKAM explanation structures can be used to validate the OMOS model.
- The OMOS analysis features point out loopholes in the OMOS knowledge base. Because of the structure-preserving transformations they correspond to loopholes in the KADS conceptual model, which again are due to loopholes in the COKAM case and explanation base.

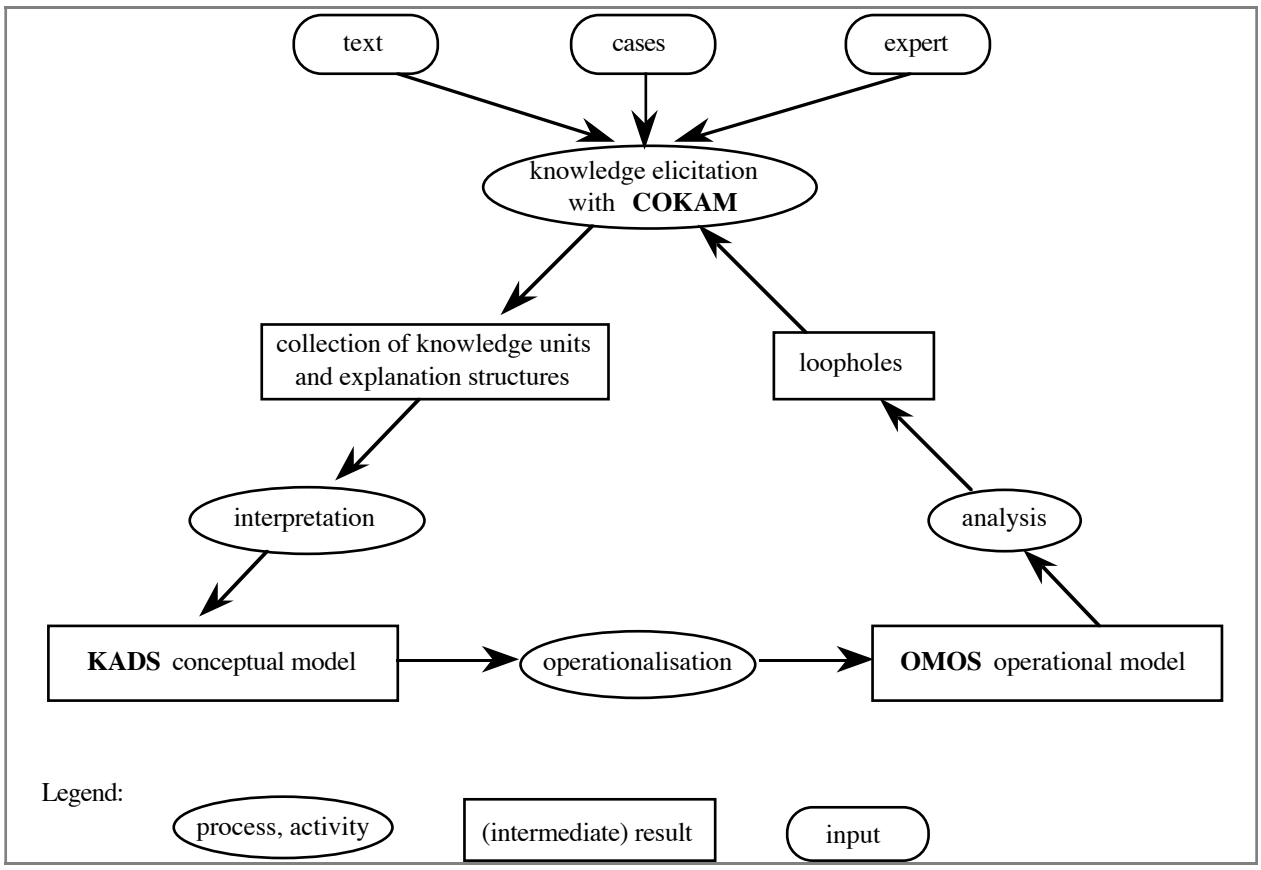

Figure 1.1: Knowledge Acquisition with COKAM, KADS and OMOS

Section 2 of this paper describes our application: the selection of an appropriate clamping tool to fix a workpiece for lathe turning. The third section describes how the tool COKAM is applied to elicit knowledge and explanation structures from texts and prior case solutions. In section 4 we will show how the output of COKAM is used to develop an adequate KADS conceptual model. Section 5 demonstrates how the KADS conceptual model is operationalized with the representation language OMOS, and Section 6 how the analysis tools of OMOS provide useful hints for the use of COKAM to complete the elicited knowledge. Section 7 summarizes and discusses the COKAM-KADS-OMOS approach.

2. OUR APPLICATION DOMAIN: CLAMPING TOOL SELECTION FOR LATHE TURNING

COKAM was developed in the ARC-TEC project (**A**cquisition, **R**epresentation and **C**ompilation of **TEC**hnical Knowledge; Richter, Boley & Wetter 1989). The goal of this project is the development of a domain specific shell for the construction of expert systems that solve various tasks in mechanical engineering. One prototypical task is the generation of a production plan for a rotational part. The production plan specifies:

- the clamping tool with which the workpiece is fixed in the lathe
- the cutting tools with which the material is removed from the mold
- the sequence of cuts
- the cutting parameters such as feed and revolutions per minute.

Figure 2.1 shows a typical rational part (a drive shaft) overlaid with the mold from which it is to be manufactured. The numbers indicate the sequence of cuts by which the material is removed.

They also refer to concrete cutting tools. Since the whole task of production planning is rather complex, we will only consider some aspects of the clamping tool selection problem to demonstrate our approach.

The clamping tool serves to center the workpiece in the lathe and to transmit the rotation. In Figure 2.1, it is depicted by the two black parts labeled left fixture and right fixture. In this particular case the left fixture, which transmits the rotation is a lathe dog and the right fixture is a lathe center. Other types of clamping tools are clamping jaws and collet chucks.

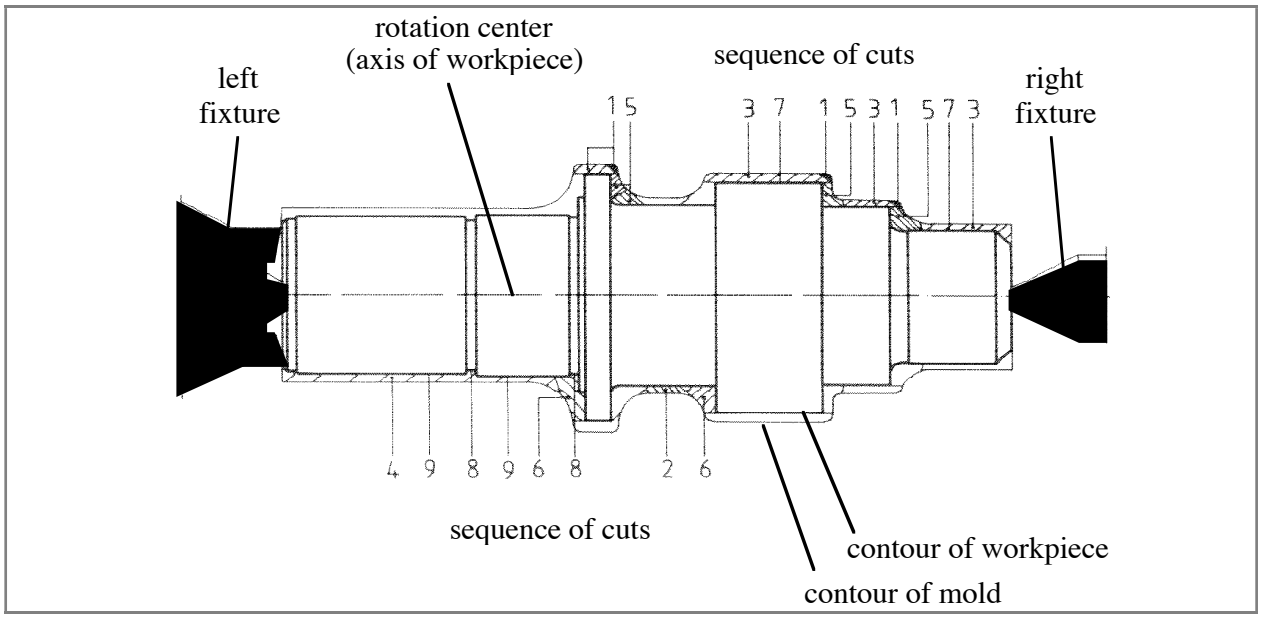

Figure 2.1: A rotational part with its clamping tool and a partial production plan (after: Example for application, SPK-Feldmühle Werkzeuge, undated)

Lathe dogs hold the workpiece from the side and allow free access to all surfaces except the left and the right vertical plane. Clamping jaws and collet chucks, on the other hand, do not use the vertical planes to fix the workpiece but use instead parts of the outside plane (represented by horizontal lines in figure 2.1). Depending on the turning requirements, i.e., which surfaces of the workpiece are to be manufactured, either a lathe dog or collet chucks are more advantageous. Besides the accessibility of different sections of the workpiece surface there are several other criteria that determine the selection of a clamping tool. In our application we will only consider the set-up time, i. e., the time that is needed to mount the clamping tool on the lathe and the clamping time, the time it takes to close the clamping tool when a new mold is inserted into the lathe.

The selection of an appropriate clamping tool depends on three different kinds of data about the manufacturing problem:

- the workpiece data which specify the geometry and the technological requirements (e.g., surface quality and rotational accuracy) of the workpiece to be manufactured,
- the job data (e.g., lot size and delivery deadlines) which pertain to the whole set of workpieces that were ordered by a customer.
- the workshop or shop floor data that specify which machines, clamping tools and cutting tools are available and which clamping tools are already mounted on the individual machines.

In the next two sections of the paper we will develop a KADS conceptual model for this simplified version of the clamping tool selection problem. We construct a KADS conceptual model with the help of the knowledge acquisition tool COKAM. We will only describe the subset of the functionality of COKAM which is relevant for the construction of the conceptual model. A more comprehensive description is given in Schmidt & Schmalhofer (1990) and Schmalhofer, Kühn & Schmidt (1990).

3. KNOWLEDGE ACQUISITION WITH COKAM

3.1. Basic Principles

COKAM (**C**ase-**O**riented **K**nowledge **A**cquisition **M**ethod from Text) is based on the idea that different traces of expertise, which serve as information sources should be used together for knowledge acquisition. In the domain of mechanical engineering there are three different traces of expertise, namely texts, records of previously solved cases, and expert's know-how. These three traces of expertise complement each other since to some extent they provide different information. Even when they contain the same information the knowledge may be obtained more easily from one information source than the other. Therefore, integrated knowledge acquisition (Schmalhofer, Kühn, Schmidt 1990) in which different traces of expertise are used, offers significant advantages. Since the traces supply different and possibly overlapping knowledge the completeness and consistency of a constructed knowledge base can more easily be established. Furthermore the relevant knowledge can be elicited more efficiently when several information sources are used.

COKAM combines the three traces of expertise as follows: Texts are the primary information source since they contain general and well structured knowledge in explicit form. Segments that are extracted from the text by an expert constitute the building blocks or knowledge units of the initial informal knowledge base. Although texts are used first, records of previously solved cases are just as important for knowledge acquisition with COKAM. Cases are used to obtain information of how the mostly declarative knowledge that was extracted from the text is applied by the expert in practice. In COKAM this is accomplished by having the expert explain solutions of different cases using the knowledge units in the informal knowledge base. Cases are used together with their solutions, since it is less time consuming to explain than to generate a solution of a particular case. The explanations of individual cases can be analyzed to check the relevance and sufficiency of the informal knowledge base. It is even more interesting, however, that the obtained explanation structures can be used to construct a task and an inference structure of a conceptual model.

3.2. Extracting an Initial Informal Knowledge Base from Text

Figure 3.1 shows how COKAM supports the construction of an initial informal knowledge base from a technical text. The text (which can be scanned in) is presented on the left of the screen. The text browser, which is located directly above the text window, helps the expert navigate through the text. The right side of the screen shows the informal knowledge base. Small windows display the knowledge units, which are arranged in the form of a card stack. The user can move up and down the stack to select a particular knowledge unit. The selected unit is always located next to the dialog window in the center of the screen. COKAM supports copying text segments to a knowledge unit, editing a knowledge unit, and assigning it to various categories. It records the history of each knowledge unit so that the user can at any time go back to the piece of text from which it originated.

The text that was used in our application contained different types of information. It described the different types of clamping tools, the function of a clamping tool, the criteria for selecting an appropriate clamping tool, and how these criteria are applied in practice. The expert was told to select all the text passages that contain relevant information for the solution of the target task, since only then a detailed explanation at different levels of abstraction can be given. The knowledge units that were thus constructed usually consisted of single text sentences, of partial sentences or combinations of two sentences. These text fragments were modified so that they could be understood without their context, i.e., anaphora and other references were resolved. Table 3.1 shows a selection of knowledge units that were obtained from our text. The expert then used these knowledge units to construct explanations for individual cases.

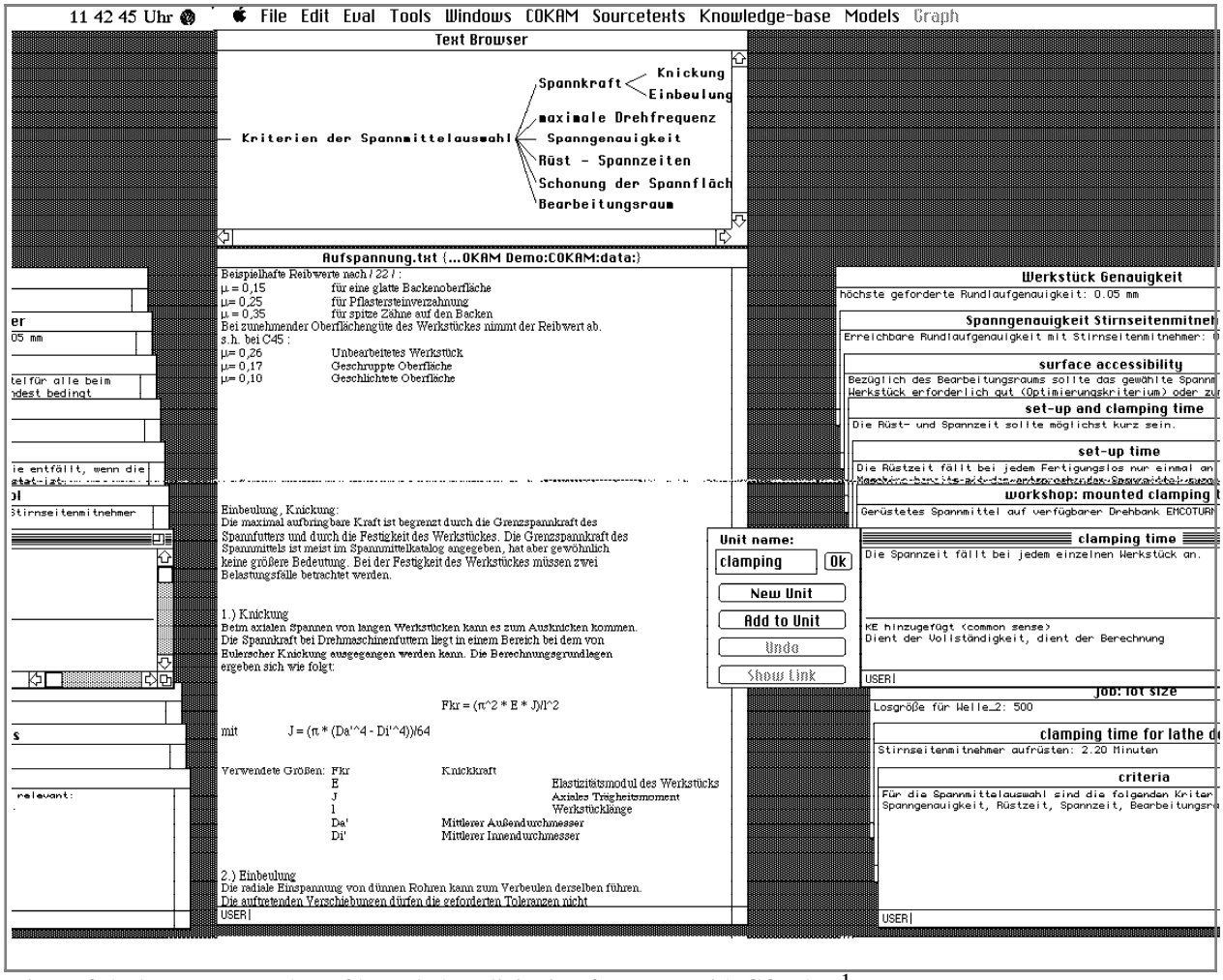

Figure 3.1: A screen snapshot of knowledge elicitation from text with COKAM[1].

1. "The purpose of the clamping tool is to center the workpiece in the lathe and to transmit the rotation."
2. "For the selection of a chucking tool the criteria accessibility of workpiece surfaces, clamping time and set-up time must be considered."
3. "A lathe dog is ideal for transverse turning but axial turning is also possible."
4. "The set-up time for a lathe dog is 3 Minutes."
5. "The set-up time of a selected clamping tool should be as short as possible."

Table 3.1: A sample of knowledge units of the informal knowledge base.

[1]A German text was used for knowledge elicitation, as our experts are German mechanical engineers.

3.3. Obtaining Explanations for Individual Cases

The screen set-up for constructing explanations is similar to figure 3.1, except that instead of the text window an explanation window is displayed, which initially contains a case description. We build an explanation tree by selecting knowledge units from the informal knowledge base and attaching them to the case description or to other units in the explanation window. When constructing the explanation, one can add new units to fill gaps in the informal knowledge base. Existing knowledge units can be reformulated, so that they can be better understood. Constructing an explanation for a case thus also helps to get a more precise formulation of the units.

Figure 3.2 shows a part of a typical explanation tree. The knowledge unit directly below the graphical representation of the case names the criteria for clamping tool selection (see knowledge unit 2 of table 3.1). The subsequent units show how these criteria are satisfied for the particular problem. The leaves of the explanation tree always refer to concrete elements of the case description, i.e., the workpiece data, the job data, the workshop data or the given solution. The links between the knowledge units should be interpreted as "explain" links, i.e., the units below a particular unit together explain that unit.

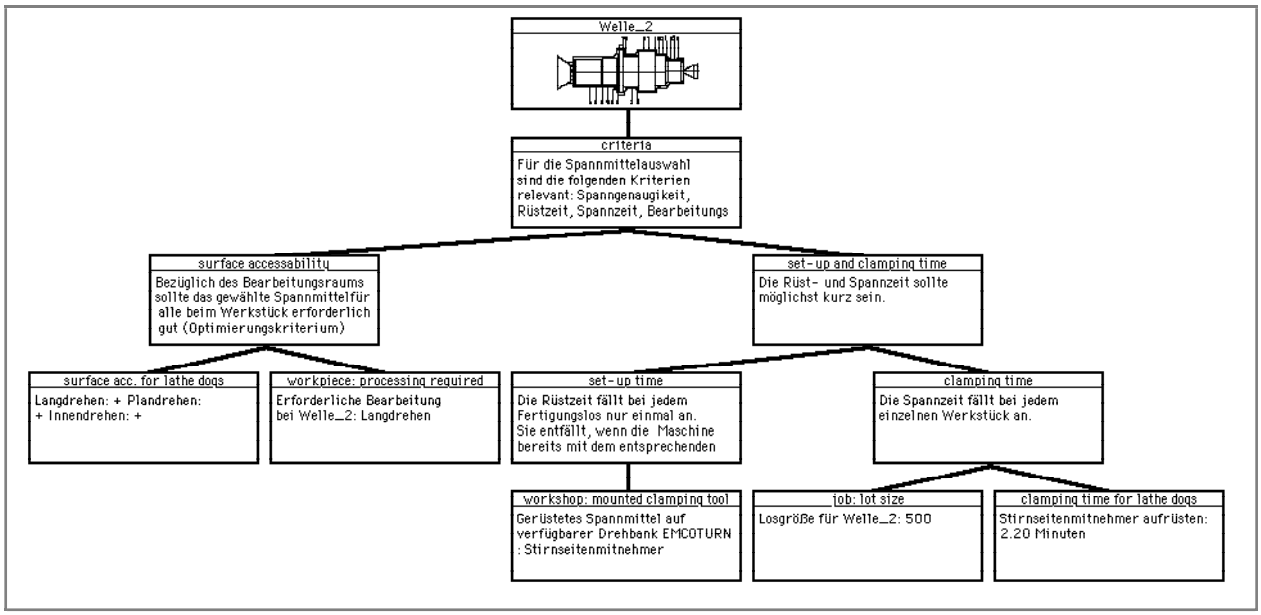

Figure 3.2: Part of an explanation tree justifying the use of a lathe dog for manufacturing the workpiece "Welle_2."[2]

4. CONSTRUCTING THE KADS CONCEPTUAL MODEL

4.1. The Subset of KADS that We Consider

KADS (Knowledge Acquisition and Design Structuring; Breuker et al. 1987, Schreiber et al 1988, Wielinga & Breuker 1989) is a methodology for the structured and systematic development of knowledge-based systems. KADS divides the development process of a knowledge-based system into a suite of models (in particular the conceptual model, the design model, the logical model; Schreiber et al 1989) that represent the transformation of knowledge from the initial, non-

[2]For the reader's convenience, the knowledge units containing German text are given English labels which characterize their contents.

formal phases into an operational system. Within the models, knowledge is represented on several layers to stress the different types of knowledge of a knowledge-based system.

Our major concern is the development of the conceptual model. According to our interpretation the conceptual model is a model of the expert's problem-solving process. It should also be the basis for the design of the system. For the construction of the conceptual model we will rely on the informal knowledge base and the explanation structures collected with COKAM. According to the KADS terminology we will distinguish a domain layer, an inference layer and a task layer in the conceptual model. The domain layer describes the static domain knowledge, which can be used for various tasks. Entities in the domain layer are concepts and relations. The inference layer specifies what inferences can be made in terms of knowledge sources and meta-classes. Knowledge sources are functional descriptions of inference making and meta-classes describe the role that the domain layer concepts can play. The inference layer is neither domain-specific nor task-specific since the knowledge sources and meta-classes may occur in different domains and different tasks. The inference structure for a specific task, however, specifies only that part of that inference layer that is relevant for the solution of this task. The task layer describes how the knowledge sources of the inference layer are applied to solve a particular task. The task structure describes one particular way to solve a task. The strategy layer controls the selection of a task solution method. We do not consider this layer since we did not observe different strategies in our application task.

4.2. The Domain Layer

The concepts and relations that describe the static knowledge of the application domain can be identified from the elements of the informal knowledge base. Since the knowledge units contain decontextualized pieces of information, which were considered to be relevant by the expert, it is easier to discover concepts and the relations in the knowledge units than in complete texts or in interview protocols.

The informal knowledge base that was constructed with COKAM contains many knowledge units that describe relations between domain concepts. For example the knowledge unit 3 of table 3.1, which states that a *lathe dog* is an *optimal clamping tool* for *transverse turning* and a possible clamping tool for *axial turning*, or more technically, the relation between *lathe dog* and *transverse turning* is *optimal clamping tool* and the relation between *lathe dog* and *axial turning* is *possible clamping tool*. Another example is knowledge unit 4 of table 3.1, which states that the *set-up time* for a *lathe dog* is 3 minutes.

The relevant domain concepts for clamping tool selection that can be identified from the units of the informal knowledge base are:

- clamping tool and the various kinds of clamping tools (e.g., clamping jaw, lathe dog and collet chuck),
- surface accessibility which depends on the turning requirements: transverse turning, axial turning, inside turning,
- clamping time, set-up time,
- workpiece.

The relevant domain relations are:

- required processing and possible processing,
- has clamping time and has set-up time,
- mounted clamping tool,
- ideal clamping tool and feasible clamping tool.

4.3. The Inference and Task Layers

We will not separate the discussion of the inference and task layer. They both describe the problem-solving method and they both depend strongly on each other. The identification of the expert's problem-solving method is a major difficulty in knowledge acquisition. One approach to overcome this difficulty is the collection of all known problem-solving methods in libraries that the knowledge engineer can consult. The KADS interpretation model library (Breuker et al 1987), Generic Tasks (Chandrasekaran 1986) and the collection of role-limiting methods (McDermott 1988) pursue this goal. Even when comprehensive libraries exist, the selection of an appropriate model still constitutes a problem. Furthermore, we believe that such libraries can never be both exhaustive and provide sufficiently detailed models. This means, that only a global model can be selected from the library, which must be refined to fit the specific task. The knowledge acquisition method COKAM supports the knowledge engineer in the selection and refinement of an appropriate problem-solving model.

The problem-solving model that we will suggest for the task of clamping tool selection was not part of the KADS library of interpretation models. We had to build a new one. Psychological research on human decision making (Huber, 1982; Gertzen, 1990) provides the foundation for our problem-solving method.

When people are confronted with a task of selecting one alternative from a large number of given alternatives based on several criteria, they first reduce the set of alternatives by eliminating some obviously bad alternatives to obtain a smaller, more manageable set of alternatives, which are then examined in more detail. In order to obtain a manageable set of alternatives the selection criteria are applied more or less rigorously. In a second step, the subjective costs of the remaining alternatives are computed on the basis of optimization criteria and the best one is selected. The described strategy has been observed when people select an apartment to rent or a car to buy.

The problem of selecting a clamping tool is similar to the above problems in that there are a large number of solution alternatives (not illustrated in the example of this paper) and several criteria that make an elimination process necessary. The above problem-solving method, which might be termed *selection by elimination and optimization* is thus a plausible candidate. However this must be supported by data collected with COKAM.

4.3.1. The Inference Structure

A closer inspection of the units in the informal knowledge base shows that they not only describe relations of domain concepts but also refer to inference layer elements. The inference layer units can be identified by searching the knowledge units for keywords that are typically used to name meta-classes and hardly ever refer to domain concepts. Such keywords are: criterion, solution, hypothesis, alternative, feature, etc. For example, knowledge unit 2 of table 3.1 contains the word *criteria*, and it indeed relates the domain concepts surface accessibility, clamping time and set-up time to the meta-class *criterion*.

Besides the individual knowledge units that give some hints for possible meta-classes and knowledge sources, the construction of the inference layer is facilitated by the explanation collected with COKAM. Whereas the knowledge units at the leaves of the explanation tree refer to the domain layer, the inner knowledge units pertain to the inference layer. There are three such knowledge units in the explanation tree shown in figure 3.2, namely the units directly below the problem description.

When the explanations that were generated with COKAM for different cases are compared to each other, it can be seen that the criteria are usually applied in the same order, i.e., the criterion *surface accessibility* is considered before the criteria *set-up* and *clamping time* (see the explanation structure in figure 3.2). Therefore it is likely that *surface accessibility* is a *selection criterion* whereas *set-up time* and *clamping time* are *optimization criteria*. An inspection of the corresponding knowledge units confirms this interpretation. The selection criterion *surface accessibility* can be instantiated in two ways so that either only the ideal clamping tools (those clamping tools that are

well suited for the required types of processing) or all feasible clamping tools (those clamping tools that are less suitable, but that still do the job) are selected.

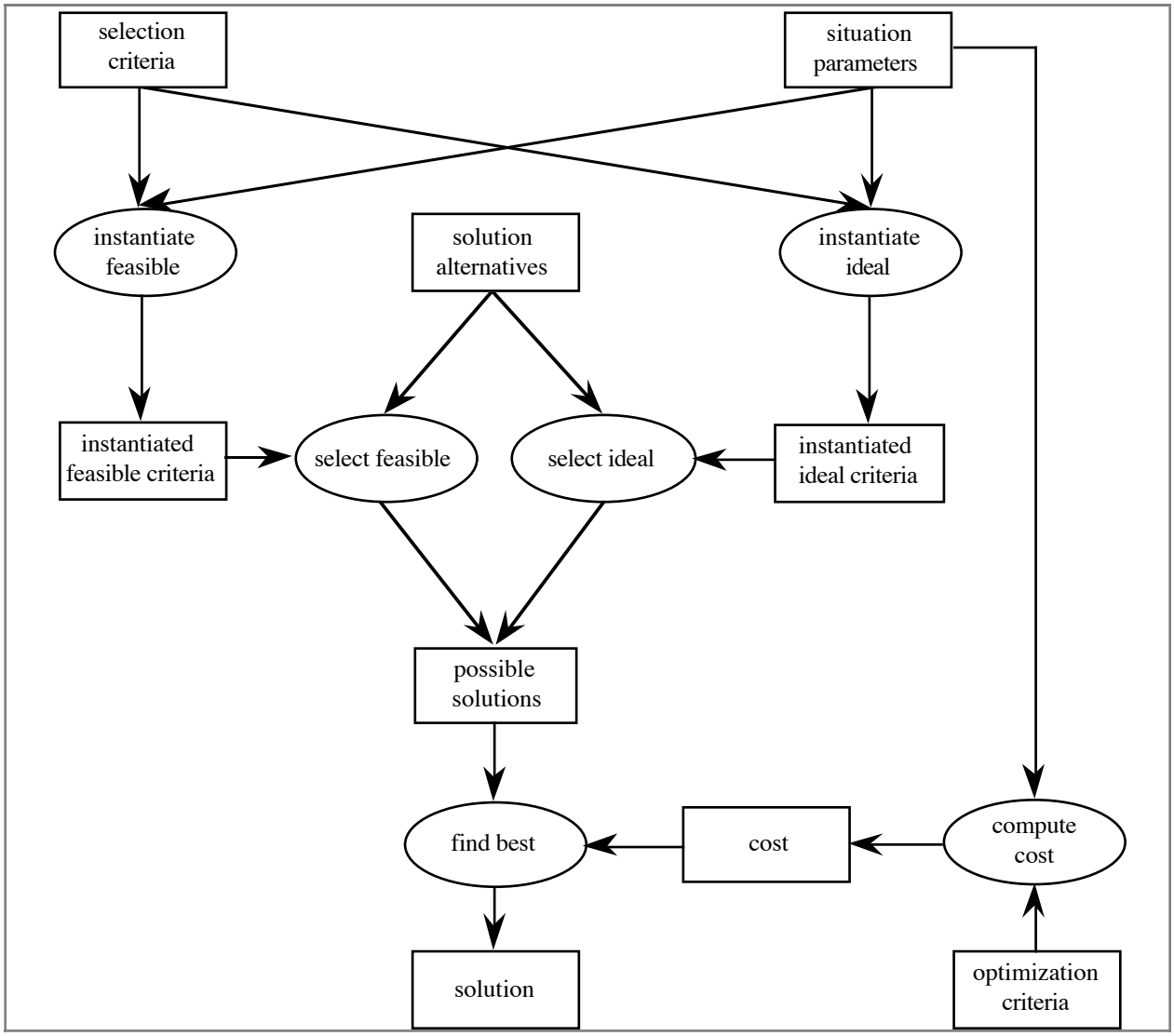

Figure 4.1 Inference structure for the problem-solving method *selection by elimination and optimization* that we use for clamping tool selection.

Figure 4.1 shows the inference structure for clamping tool selection. There are two ways to select the possible solutions from the set of solution alternatives (e.g., the available clamping tools specified in workshop data). The selection criteria (surface accessibility) can be instantiated with the situation parameters (the problem description except the solution alternatives) so that either all feasible or only the ideal possible solutions are selected. From the set of possible solutions the best solution is the one with the lowest cost. The cost for each possible solution is computed based on the optimization criteria (set-up and clamping time) and the relevant situation parameters (lot size and mounted clamping tool).

4.3.2. The task structure

The task structure describes the sequence in which the knowledge sources of the inference structure are used to solve the task. There are no units in the informal knowledge base that contain such information. This is not surprising since texts usually do not describe in detail how to solve par-

ticular tasks. To some extent the explanations that were generated with COKAM provide information about the sequence of knowledge sources as already pointed out in the previous section.

Figure 4.2 shows the task structure for clamping tool selection. The ideal instantiation of the selection criteria is used first since it allows a more extensive reduction of the number of solution alternatives. If the ideal instantiation does not produce any solutions, then we resort to feasible, sub-optimal instantiations of the selection criteria. In our application, the selection criteria are descriptions of the surface accessibility required to turn a workpiece: inside, transverse or axial turning. For the solutions of the selection process we compute costs. Cost computation uses optimization criteria, which in our application correspond to set-up time and clamping time.

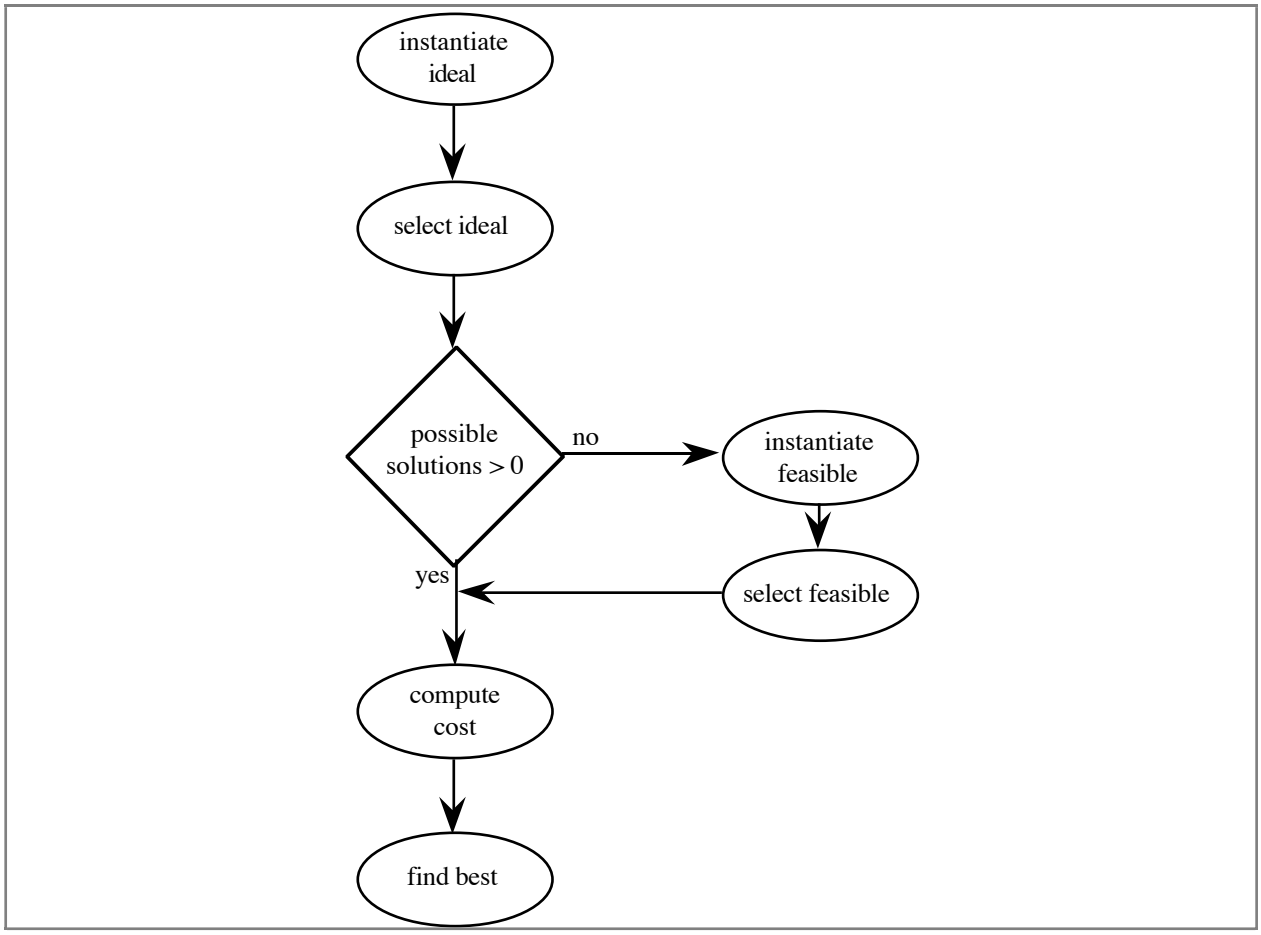

Figure 4.2: Task structure for the problem-solving method *select by elimination and optimization* that we use for clamping tool selection.

5. OPERATIONALIZING THE CONCEPTUAL MODEL WITH OMOS

This section of the paper describes how the KADS conceptual model of the clamping tool selection task is represented in the operational language OMOS and which transformation processes are necessary to do so. We will limit ourselves to a brief description of the language OMOS and of the implemented clamping-tool-selection system, as this article focusses on the feedback that a structure-preserving implementation of an explicit model of expertise provides for the continuing knowledge elicitation and analysis (see figure 1.1).

5.1. The Language OMOS

OMOS (Operational Models of Problem-Solving) is a representation language for operational problem-solving models for knowledge-based systems (Linster 1990). OMOS models consist of a generic part that describes the problem-solving method and of an application-specific part that describes the domain knowledge that is used by the problem-solving method. OMOS tries to combine the advantages of KADS (i.e., flexibility of modelling for a wide range of tasks, reusability of the generic aspects of the models) with the advantages of operational approaches to the modelling of problem-solving as we find them in PROTEGE/p-OPAL (i.e., use of operational data-structures to represent detailed domain-knowledge, automatic analysis capabilities and strong guidance in the acquisition and representation of the application-specific knowledge; Musen 1989). (Wetter 1990) and (Voß et al. 1990) describe related work.

OMOS represents knowledge in a two-layered KADS-like framework, consisting of a domain layer and of a layer for the problem-solving method.

5.1.1. The Language Constructs for the Domain Layer

On the domain layer OMOS provides language facilities for the definition of concept hierarchies and for the definition of relations between concepts.

Frames, attribute-descriptions for the frames and instances of the frames are the building blocks for the concept hierarchies. In OMOS, frames are used to define domain types (e.g., the type *clamping-tool)*. Frames can be included in multiple-inheritance hierarchies. Attributes, defined for the frames, provide uniform descriptions for the instances of the frames, such as the attribute *set-up-time* for the frame *clamping-tool*. Concepts of the domain layer are represented as instances of frames with values for the attributes.

Relations express information that relates to several concepts. The relation *optimal-clamping-tool* relates clamping tools (i.e., instances of the frame *clamping-tool*) to turning requirements (i.e., instances of the frame *turning-requirement*). Relations are lists of tuples or predicates, whose extension corresponds to a finite list of tuples. The arguments of the relations correspond to instances of frames. The relation *optimal-clamping-tool* has instances of the frame *clamping-tool* as a first argument whereas the second argument refers to instances of the frame *turning-requirement*.

5.1.2. The Language Constructs for the Layer of the Problem-Solving Method

On the layer of the problem-solving method OMOS provides knowledge sources, meta-classes, inference structures and control-structures to represent a problem-solving method. OMOS uses a KADS-like terminology on this layer, but the implementation of the terms is different. Important differences exist for the implementation of the knowledge sources and for the inference structure. The latter one is a data-dependency diagram in KADS, whereas in OMOS many elements of directed and conditional data-flows are included. KADS uses the inference and task layer elements to conceptualize problem-solving behavior. OMOS is an implementation language.

Meta-classes describe roles that domain layer concepts play in a problem-solving process. Initially the meta-class *solution-alternative* describes the concept *collet-chuck*. In case it is selected because the workpiece could be turned using a *collect-chuck* then it changes its role and becomes a *possible-solution*. If it is the best solution as far as some cost-factor is concerned, then it plays the role *solution*. Thus these roles are dynamic.

Knowledge sources are inference functions that work on domain layer concepts. They can modify the role-assignment of a concept or they can change value-assignments of attributes of a concept. An OMOS knowledge source has one input meta-class, one output meta-class and several control meta-classes. A knowledge source moves concepts from the input meta-class to the output meta-class. It can assign values to concepts that currently play the role that is denoted by the output meta-class. Conditionals are expressed through the elements of the control meta-class. Knowledge

sources use the knowledge that is expressed in the domain-layer relations. The inference functionality of a knowledge source is defined declaratively by stating the type of value-assignment it implements (none, initial or modification), its type of role-assignment (none or transfer), the domain-layer relation it uses and the input, output and control meta-classes.

The inference structure is defined implicitly through the meta-classes and knowledge sources. It is not an explicit construct of OMOS.

The control structure specifies the sequence in which the knowledge sources are called. Conditionals may only refer to knowledge sources and meta-classes. The control structure corresponds to the fixed task structures of KADS. OMOS does not provide an equivalent for the KADS strategy layer.

5.1.3. The Interpretation of the Language OMOS

OMOS is implemented in BABYLON (Guesgen, Junker & Voß 1987, DiPrimio & Wittur 1987). The domain layer elements of OMOS are BABYLON language constructs: frames and instances for the types and the concepts; PROLOG for the tuples of the relations. The constructs of the problem-solving layer are compiled into BABYLON constructs. The meta-classes are dynamic prolog predicates, that hold for pairs of domain-concepts and roles. The knowledge sources are transformed into forward-rule schemes, which implement the value-assignment and role-assignment description and that use the underlying domain relation. When a knowledge source is called, the rule-scheme executes for all the tuples of the domain relation. Thus all the constructs of OMOS are transformed into BABYLON constructs, and we only need a rudimentary interpreter.

5.2. Application Limitations for OMOS

The current implementation of OMOS is limited to finite domains. The problem-solving methods must obey several characteristics. The inference steps must be represented as selection processes. The problem-solving method must be a pre-defined iteration through a limited number of inferences. Problem-solving roles, i.e., meta-classes, may only refer to domain-concepts, and not to attributes of concepts or to tuples of relations. It is not clear yet how these limitations restrain the applicability of OMOS. It has been used successfully for the implementation of more complex systems such as K-ONCOCIN, a re-modelling of large parts of ONCOCIN with KADS terminology (Linster & Musen 1991).

5.3. Formalizing the Conceptual Model of Clamping Tool Selection

5.3.1. The Domain Layer

The domain language features of OMOS allow for a straight-forward implementation of the domain layer knowledge of the KADS conceptual model described in section 4.2.

5.3.1.1. The Concepts

Figure 5.1 shows the hierarchical definition of types and instances for the domain layer concepts. Not all the concepts described in section 4.2 appear in figure 5.1, as not all of them have been defined explicitly in OMOS (e.g., *lot-size* is a concept of the conceptual model, but an attribute with values in the implementation). Several of the relations described in the conceptual model were implemented as attributes of concepts too, for example *has clamping time*, *mounted clamping tool*, *required processing*. Thus they do not appear in figure 5.2.

In the implementation we had to introduce the type *workpiece-cost-element* (see figure 5.1) to provide concepts that can play the roles defined by meta-class *optimization-criterion*. This is because OMOS cannot apply meta-classes to attributes, only to concepts. The type *turning-requirement* is a more restrictive, and within the context of our system much more precise term for *surface-accessibility*. Its instances describe precisely which kind of turning operation is needed to model the surface of a workpiece, i.e., how we have to access the surface with our tool.

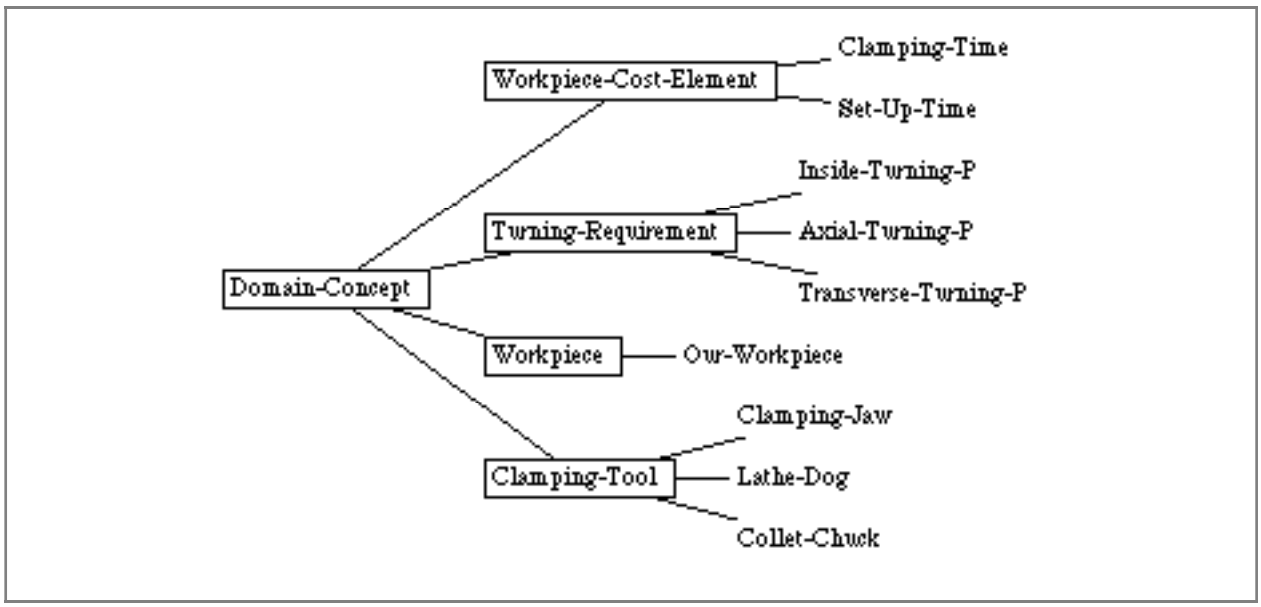

Figure 5.1: The hierarchical definition of concepts. Types are boxed, instances are not. The lines between types denote inheritance. The lines between types and instances describe instantiation.

5.3.1.2. The Relations

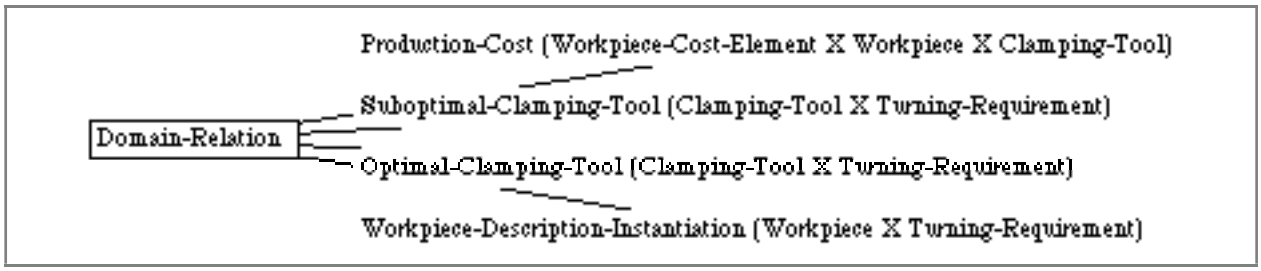

Figure 5.2: The relations are defined on the instances of the domain layer types. The tuples of the relation *production-cost* have instances of the type *workpiece-cost-element* in the first argument position, instances of *workpiece* in second position and the last argument is of type *clamping-tool*.

```
(DEFINE-DOMAIN-RELATION Optimal-Clamping-Tool
 WITH ARITY = 2
      TYPE-SEQUENCE = (((INSTANCE Clamping-Tool)
                        (EXTENSION-* Turning-Requirement)))
      ARGUMENT-SEQUENCE =
                      ((Clamping-Tool Important-Turning-Requirements))
      TUPLES = ((
                ((Optimal-Clamping-Tool
                          Collet-Chuck
                          ((Transverse-Turning-P Required-Value = Yes)
                           (Inside-Turning-P Required-Value = No))))

                ((Optimal-Clamping-Tool
                          Lathe-Dog
                          ((Axial-Turning-p Required-Value = Yes)
                           (Inside-Turning-P Required-Value = No))))
…
```

Table 5.1: The definition of the domain layer relation *optimal-clamping-tool* and some of the tuples of the relation.

```
(DEFINE-DOMAIN-RELATION Suboptimal-Clamping-Tool
 WITH ARITY = 2
      TYPE-SEQUENCE = (((INSTANCE Clamping-Tool)
                        (EXTENSION-* Turning-Requirement)))
      ARGUMENT-SEQUENCE =
                      ((Clamping-Tool Important-Turning-Requirements))
      TUPLES = ((
                      ((Suboptimal-Clamping-Tool
                           Collet-Chuck
                           ((Inside-Turning-P Required-Value = No))))

                      ((Suboptimal-Clamping-Tool
                           Lathe-Dog
                           ((Inside-Turning-P Required-Value = No))))
…
```

Table 5.2: The definition of the domain layer relation *suboptimal-clamping-tool* and some of the tuples of the relation.

The relation *optimal-clamping-tool* (see table 5.1) describes which clamping tool is a first choice for certain turning requirements, for example a *collet chuck* is the optimal choice if the workpiece requires transverse turning and inside turning is not needed. The relation *suboptimal-clamping-tool* (see table 5.2) describes a relaxation of the relation *optimal-clamping-tool*. For example the second tuple *(Suboptimal-Clamping-Tool Lathe-Dog ((Inside-Turning-P Required-Value = No)))* is a formal representation of the third knowledge unit of table 3.1: *a lathe dog is ideal for transverse turning but axial turning is also possible*. In the formal representation of the relaxed choice criteria, we only mention that a lathe dog can definitely *not* be used for inside turning.

5.3.2. Formalization of the Problem-Solving Method *Selection by Elimination and Optimization*

The problem-solving method of the operational system is built from the inference and task layers of the conceptual model.

5.3.2.1. Modifications to the Inference Structure of the Conceptual Model

The formalization of the underlying domain layer knowledge has shown us that the knowledge sources *instantiate-feasible*, *instantiate-ideal*, *select-ideal* and *select-feasible* are redundant in the problem-solving method *selection by elimination and optimization*. It is sufficient to instantiate the selection criteria once and then use them in two different ways. Furthermore we noticed that the meta-classes *cost* and *possible-solutions* cannot be separated, as they both relate to different aspects of the same domain concepts[3]. Thus in the OMOS inference structure the knowledge source *compute-cost* operates on the meta-class *possible-solution*. In OMOS meta-classes describe the roles that concepts play and they do not refer to a set of concepts currently playing that role. Thus the set-like descriptors of the conceptual model, such as *possible-solutions* become role-descriptors such as *possible-solution,* which will be used in the OMOS inference structure (see figure 5.3), which is a simplification of the inference structure of the conceptual model (see figure 4.1).

[3] This is due to a limitation of the language OMOS because only concepts can play roles and not attributes of concepts.

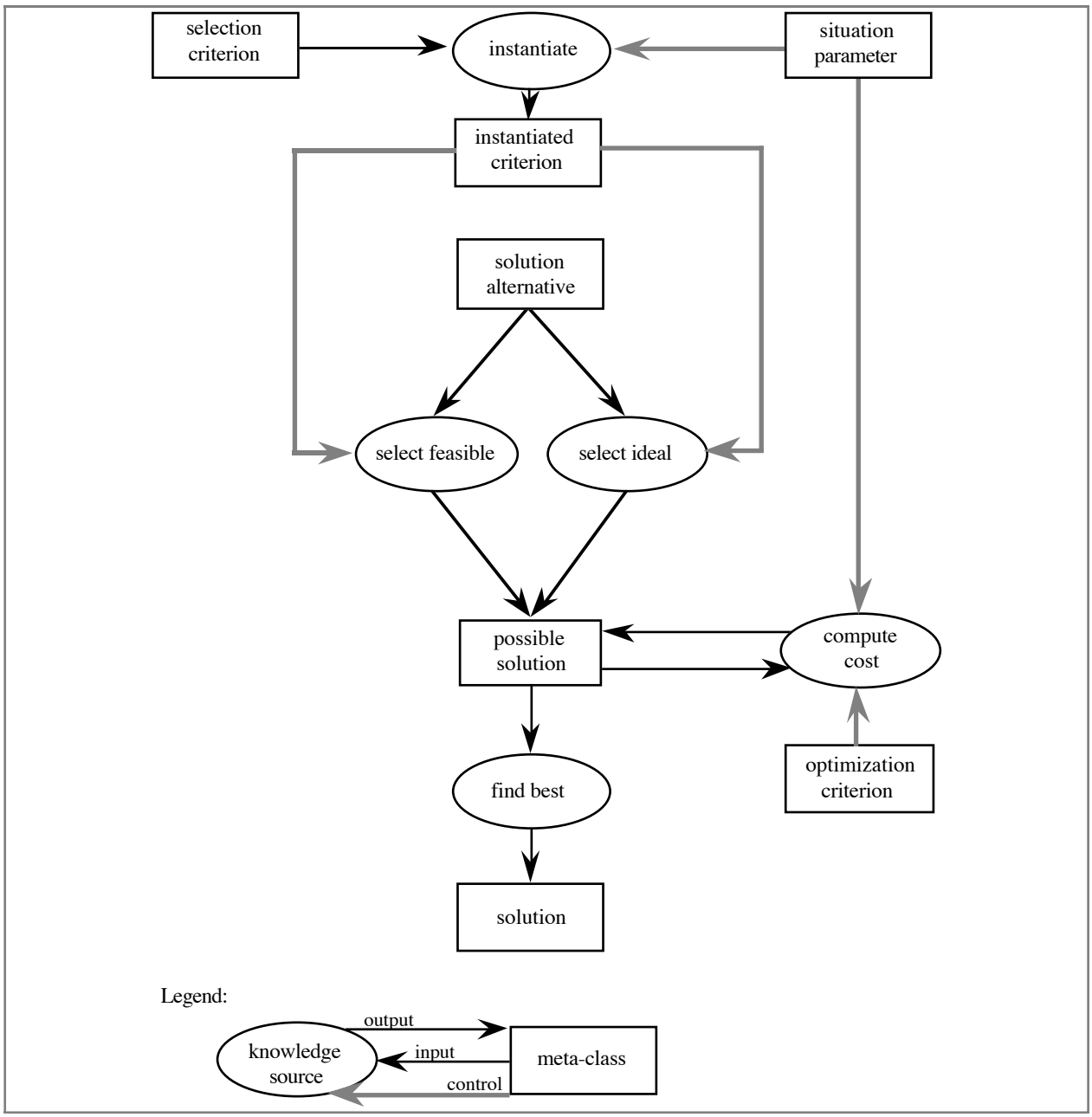

Figure 5.3: The OMOS inference structure for the problem-solving method *select by elimination and optimization.*

5.3.2.2. The Knowledge Sources & Meta-Classes

Meta-classes describe dynamic roles that domain concepts play. Some concepts have initial roles; for example all the instances of the type *clamping-tool* have the initial role assignment *solution-alternative*.

```
(DEFINE-META-CLASS Solution-Alternative
  WITH INITIAL-ROLE-ASSIGNMENT = ((Clamping-Tool _Tool)))
```

Table 5.3: The definition of the meta-class *solution-alternative*, with its initial role assignment referring to all the instances of the domain type *clamping-tool.*

```
(DEFINE-KNOWLEDGE-SOURCE Select-Ideal
 WITH INPUT-META-CLASS = Solution-Alternative
      OUTPUT-META-CLASS = Possible-Solution
      CONTROL-META-CLASSES = ((Instantiated-Criterion))
      VALUE-ASSIGNMENT = FALSE
      ROLE-ASSIGNMENT = TRANSFER
      DOMAIN-RELATION = Optimal-Clamping-Tool
      REL-ARG-TYPE = (((0 . INSTANCE)(1 . EXTENSION-*)))
      PROJ-MCS-REL-ARGS = (((INPUT-META-CLASS . 0)
                            (OUTPUT-META-CLASS . 0)
                            ((CONTROL-META-CLASSES . 0) . 1))))
```

Table 5.4: The definition of the knowledge source *select-ideal* with its corresponding meta-classes, the value- and role-assignment descriptions, the domain layer relation and the mapping of the meta-classes onto the arguments of the domain layer relation.

In OMOS knowledge sources are defined through their meta-classes, the domain relation they use, the value- and role-assignment descriptions and the mapping of the meta-classes onto the argument positions of the underlying domain relation. A compiler translates this definition (see table 5.4) into a forward rule-schema. Table 5.5 shows how the first tuple of the relation *optimal-clamping-tool* (see table 5.1) is used as a forward chaining rule to implement the knowledge source *select-ideal* (see table 5.4).

```
IF   (CURRENT-META-CLASS Solution-Alternative Collet-Chuck)
     (CURRENT-META-CLASS Instantiated-Criterion Transverse-Turning-P)
     (CURRENT-META-CLASS Instantiated-Criterion Inside-Turning-P)
     (Transverse-Turning-P Required-Value = Yes)
     (Inside-Turning-P Required-Value = No)
THEN (TRANSFER-ROLE Solution-Alternative Possible-Solution
                                                          Collet-Chuck)
```

Table 5.5: A beautified version of the forward-chaining rule that the compiler generates from the definition of the knowledge source and from the underlying domain relation.

5.3.2.3. The Control Structure

The control structure for the problem-solving method *selection by elimination and optimization* (see figure 5.4) is identical to the task structure of the KADS conceptual model (see figure 4.2), except for the changes described for the OMOS inference structure. In OMOS control structures, conditionals may only refer to elements of the inference structure. In table 5.6 the statement `(CALL-KS Instantiate)` is an activation of the knowledge source *instantiate*. `(ROLE-P Possible-Solution)` is a test that evaluates to true if at least one concept currently plays the role denoted by the meta-class *possible-solution*.

```
(DEFINE-CONTROL-STRUCTURE Select-by-Elimination-and-Optimization
      CALL-SEQUENCE = (((CALL-KS Instantiate)
                        (CALL-KS Select-Ideal)
                        (WHEN (NOT (ROLE-P Possible-Solution))
                           (CALL-KS Select-Feasible))
                        (CALL-KS Compute-Cost)
                        (CALL-KS Find-Best))))
```

Table 5.6: The control structure consists of a procedural definition of activations of knowledge sources and tests of meta-classes.

5.4. Insights through Formalization

As described previously, the formalization of the inferences has shown us, that several knowledge sources of the inference structure of the conceptual model (*instantiate-feasible*, *instantiate-ideal*, *select-ideal* and *select-feasible)* were redundant, even though our analysis of the explanation structures seemed to reveal that experts used different selection criteria in different selection processes. In retrospect though, this simplification is not a contradiction to the human problem-solving process described in section 4.3.

6. FEEDBACK FROM THE OPERATIONAL MODEL FOR THE KNOWLEDGE ELICITATION

6.1. The Use of the OMOS Analysis Capabilities for Focussed Knowledge Acquisition

OMOS provides a series of analysis tools to check the domain layer knowledge and the use of the domain layer knowledge by the inference layer elements. Completeness checks of the domain layer relations provide feedback that tells us wether all concepts of a domain-type are used in those domain relations that refer to elements of that type. Table 6.1 shows the result of the analysis of the relation *suboptimal-clamping-tool*. The first two paragraphs of the analysis text tell us that the domain concepts *axial-turning-p* and *transverse-turning-p* are not used in that relation, even though they are potential values for the first argument position of the tuples of that relation (see table 5.2 for the definition of the relation *suboptimal-clamping-tool*).

The same analysis text (table 6.1) tells us in the third paragraph that *clamping-jaw* is not mentioned. *Clamping-jaw* is mentioned again in table 6.2. This table shows the results of analyzing the use of the domain layer knowledge in a meta-class. The analysis tells us that the concept *clamping-jaw* can play the role denoted by the meta-class *solution-alternative*. This meta-class is the input meta-class of the knowledge sources *select-ideal* and *select-feasible* (see figure 5.3). None of the forward chaining rules that represent the implementation of the these knowledge sources (see table 5.5) refer to the concept *clamping-jaw* in their preconditions. This points to an open end in the knowledge base, which should be the topic of a focussed knowledge acquisition session with COKAM.

```
The domain layer concept AXIAL-TURNING-P is not used in any tuple of the relation
SUBOPTIMAL-CLAMPING-TOOL it should appear in 1. position (called IMPORTANT-TURNING-
REQUIREMENTS)  with the argument description (EXTENSION-* TURNING-REQUIREMENT).

The domain layer concept TRANSVERSE-TURNING-P is not used in any tuple of the
relation SUBOPTIMAL-CLAMPING-TOOL it should appear in 1. position (called
IMPORTANT-TURNING-REQUIREMENTS)  with the argument description (EXTENSION-*
TURNING-REQUIREMENT).

The domain layer concept CLAMPING-JAW is not used in any tuple of the relation
SUBOPTIMAL-CLAMPING-TOOL it should appear in 0. position (called CLAMPING-TOOL)
with the argument description (INSTANCE CLAMPING-TOOL).
```

Table 6.1: The results of an analysis of the relation **suboptimal-clamping-tool**.

```
The domain concept CLAMPING-JAW can be assigned to the meta-class SOLUTION-
ALTERNATIVE, but it is not used by any of the knowledge sources SELECT-IDEAL,
SELECT-FEASIBLE that use SOLUTION-ALTERNATIVE as control- or as input-meta-class.

To change this do AT LEAST ONE of the following:

    Add a tuple to the relation OPTIMAL-CLAMPING-TOOL, which is used by the
    knowledge source SELECT-IDEAL. CLAMPING-JAW must be mentioned in the 0.
    argument of the tuple, which is of type (INSTANCE CLAMPING-TOOL) and is called
    CLAMPING-TOOL.

    Add a tuple to the relation SUBOPTIMAL-CLAMPING-TOOL, which is used by the
    knowledge source SELECT-FEASIBLE. CLAMPING-JAW must be mentioned in the 0.
    argument of the tuple, which is of type (INSTANCE CLAMPING-TOOL) and is called
    CLAMPING-TOOL.
```

Table 6.2: The results of the analysis of the meta-class *instantiated-criterion*.

```
The condition (Meta-Class-Applicable Possible-Solution Clamping-Jaw) derived from
the tuple (PRODUCTION-COST SET-UP-TIME OUR-WORKPIECE (CLAMPING-JAW PRODUCTION-COST
= (+ (<- CLAMPING-JAW :GET 'PRODUCTION-COST) (<- CLAMPING-JAW :GET 'SET-UP-TIME))))
of the relation PRODUCTION-COST used in the knowledge source COMPUTE-COST can never
be fired, as none of the knowledge sources COMPUTE-COST, SELECT-FEASIBLE, SELECT-
IDEAL assign the role POSSIBLE-SOLUTION to the domain concept CLAMPING-JAW.

To change this do AT LEAST ONE of the following:

    Add a tuple to the relation PRODUCTION-COST, which is used by the knowledge
    source COMPUTE-COST. CLAMPING-JAW must be mentioned in the argument nbr. 2,
    which is of type (EXTENSION CLAMPING-TOOL).

    Add a tuple to the relation SUBOPTIMAL-CLAMPING-TOOL, which is used by the
    knowledge source SELECT-FEASIBLE. CLAMPING-JAW must be mentioned in the
    argument nbr. 0, which is of type (INSTANCE CLAMPING-TOOL) and is called
    CLAMPING-TOOL.

    Add a tuple to the relation OPTIMAL-CLAMPING-TOOL, which is used by the
    knowledge source SELECT-IDEAL. CLAMPING-JAW must be mentioned in the argument
    nbr. 0, which is of type (INSTANCE CLAMPING-TOOL) and is called CLAMPING-TOOL.
```

Table 6.3: The results of the analysis of the knowledge source *compute-cost*.

The analysis of the knowledge sources produces similar results (see table 6.3). The knowledge source *compute-cost* uses the domain relation *production-cost*. The rule-schema that the compiler produces to evaluate the knowledge source (see table 5.5) will contain a precondition testing wether *clamping-jaw* currently plays the role defined by the meta-class *possible-solution*. The knowledge base analyzer determines that this can never be the case, as none of the knowledge sources that use *possible-solution* as output-meta-class assign that role to that concept. The analyzer proposes several ways to remedy this. This information, coined in the terms of the operational model can easily be traced back to the conceptual model, as the operational model is a structure-preserving implementation of the conceptual model.

6.2. Using the Feedback in the Next Session With COKAM

The results of the OMOS analysis, which indicate gaps in the so far acquired knowledge, are used in various ways in the next session of COKAM. In order to check whether the missing knowledge was overlooked when extracting knowledge units from the text, the text can be searched for the respective keywords (e.g., *axial-turning* or *clamping-jaw*). Since COKAM records which text passages contributed to the informal knowledge base, it can be easily determined whether the

keywords occur in text passages which contain potentially novel information. In this case the expert has to judge whether these text segments contain information which should be added to the informal knowledge base.

If the uncovered knowledge gaps can not be filled from the initially selected text, the expert is asked for additional literture which covers these topics. The knowledge gaps mentioned in table 6.1 are due to the fact that our initial text dealt mostly with clamping tool selection for outside turning and only contained a few references to inside turning.

Furthermore it may be advisable to select a new case for which the missing knowledge is supposedly needed and have the expert generate an explanation. This should be done in particular when the knowledge gaps can not be easily filled from the initial or the additional text or when the correctness of the constructed conceptual model is in question and must be reexamined. Since the OMOS analysis indicates that the knowledge concerning clamping jaws is fragmentary, a case in which a clamping jaw is applied to fix a workpiece should be explained by the expert. If the obtained explanation structure substantially differs from the other explanation structures this may indicate that the conceptual model must be modified. For example, the original conceptual model may be too simple, since the relevance of certain criteria only becomes obvious, when cases are encountered for which these criteria cannot be neglected.

7. SUMMARY & OUTLOOK

The combination of COKAM, KADS and OMOS is an experiment that we started to analyze the interaction of different contributions to knowledge acquisition. We spanned the bridge from natural language texts and experts' explanations of previously solved cases (COKAM) via systematic models of problem solving (KADS) to an operational system (OMOS). In these transformation processes we focused on the following questions:

- how can a KADS model be build from explanation structures, and how does the KADS model influence the generation of explanations?
- how can we formalize the KADS model, and what does the formalization tell us about the model?
- what does the formal model propose for the next knowledge acquisition session?

In the present paper we used a toy version of a real world problem to analyze and illustrate our approach. The future work will concentrate on full-blown versions of the process planning problems for rotational parts.

BIBLIOGRAPHY

Akkermanns, H., Balder, J., van Harmelen, F., Schreiber, G. & Wielinga, B.: Formal Specifications of Knowledge Level Models, REFLECT Report, Esprit Basic Research Project P3178, Task 1.4, 1990

Breuker, J., Wielinga, B., van Someren, M., de Hoog, R., Schreiber, G., de Greef, P., Bredeweg, B., Wielemaker, J., Billaut, J.P., Davoodi, M., and Hayward, S. Model-Driven Knowledge Acquisition: Interpretation Models. *Tech. Rept. Esprit Project 1098*, Deliverable Task A1, University of Amsterdam, Social Sciences Informatics, Amsterdam, 1987.

Breuker, J. and Wielinga, B. Models of Expertise in Knowledge Acquisition. In *Topics in Expert System Design, Methodologies and Tools*. North-Holland, Guida, G. and Tasso, C., pp. 265 - 295, Amsterdam, 1989.

Chandrasekaran, B. Generic Tasks in Knowledge-Based Reasoning: High-Level Building Blocks for Expert System Design. *IEEE Expert* (Fall 1986), 32 - 30.

DiPrimio, F. and Wittur, K.: BABYLON, a Meta-Interpretation Model for Handling Mixed Knowledge Representations, in: Proceedings of the 7th International Workshop on Expert Systems and Their Applications, Avignon 1987, pp. 821 - 833.

Gertzen, H. Entscheidungen bei sequenzierter Informationsdarbietung am Bildschirm. Münster; New-York: Waxmann, 1990.

Guesgen H.W., Junker, U. and Angi Voß: Constraints in a Hybrid Knowledge Representation System, in: Proceedings of IJCAI87, 1987, pp. 30 - 33.

Huber, O. Entscheiden als Problemlösen. Bern: Huber, 1982.

Linster, M. Declarative Problem Solving Procedures as a Basis for Knowledge Acquisition: A First Proposal. Tech. Rept. 448, Arbeitspapiere der GMD, GMD, St. Augustin, June, 1990.

Linster, M. and Musen M. Use of the KADS Knowledge Acquisition Method to Model the ONCOCIN Task of Cancer-Chemotherapy Administration, submitted to EKAW91, Crieff, 1991

McDermott, J. Preliminary Steps Toward a Taxonomy of Problem-Solving Methods. In *Automating Knowledge Acquisition for Expert Systems*. Kluwer Academic, Marcus, S., Ch. 8, pp. 225 - 256, Boston, 1988.

Musen, M.A. *Automated Generation of Model-Based Knowledge Acquisition Tools,* Pitman Publishing, London , Research Notes in Artificial Intelligence 1989 .

Richter, M., Boley, H., Wetter, T., and Warnecke, G. ARC-TEC: Acquisition, Representation and Compilation of Technical Knowledge. *Tech. Rept. Projektantrag, German Research Center for Artificial Intelligence, Postfach 2080,D-6750 Kaiserslautern*, 1989.

Schmalhofer, F., Kühn, O. and Schmidt, G., 1990: Integrated Knowledge Acquisition from Text, Previously Solved Cases and Expert Memories; *Research Report RR-90-14, German Research Center for Artificial Intelligence, Postfach 2080,D-6750 Kaiserslautern*, 1990.

Schmidt, G. and Schmalhofer, F. Case-Oriented Knowledge Acquisition from Texts. In *Current Trends in Knowledge Acquisition*, Wielinga, B., Boose, J., Gaines, B., Schreiber, G., and van Someren, M., IOS Press, May 1990, pp. 302-312.

Schreiber,G., Akkermans, H. Wielinga, B.: On Problems with the Knowledge Level Perspective,In *Proceedings of the 5th Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, November 1990, pp. 30-1-30-14.

SPK-Feldmühle Werkzeuge, Examples for Application (Turning and Milling), Feldmühle AG, Produktbereich SPK-Werkzeuge, D-7333 Ebersbach a d.. Fils, Gottlieb-Häflel-Str. 7.

Voß A., Karbach, W., Drouven U., Lorek D, and Schuckey R.: Operationalization of a Synthetic Problem, ESPRIT Basic Research Report Project P3178 REFLECT, Task 1.2.1, 1990.

Wetter, T. First Order Logic Foundation of the KADS Conceptual Model, Wielinga, B., Boose, J., Gaines, B., Schreiber, G., and van Someren, M., IOS Press, May 1990, pp. 356 - 375.

Wielinga, B., Akkermans, H., Schreiber, G., and Balder, J. A Knowledge Acquisition Perspective on Knowledge-Level Models. In *Proceedings of the 4th Knowledge Acquisition for Knowledge-Based Systems Workshop*, October 1-6 1989, pp. 36-1-36-22.

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

DFKI
-Bibliothek-
PF 2080
67608 Kaiserslautern
FRG

DFKI Publikationen

Die folgenden DFKI Veröffentlichungen sowie die aktuelle Liste von allen bisher erschienenen Publikationen können von der oben angegebenen Adresse oder per anonymem ftp von ftp.dfki.uni-kl.de (131.246.241.100) unter pub/Publications bezogen werden.
Die Berichte werden, wenn nicht anders gekennzeichnet, kostenlos abgegeben.

DFKI Research Reports

RR-93-16
Gert Smolka, Martin Henz, Jörg Würtz: Object-Oriented Concurrent Constraint Programming in Oz
17 pages

RR-93-17
Rolf Backofen:
Regular Path Expressions in Feature Logic
37 pages

RR-93-18
Klaus Schild: Terminological Cycles and the Propositional μ-Calculus
32 pages

RR-93-20
Franz Baader, Bernhard Hollunder:
Embedding Defaults into Terminological Knowledge Representation Formalisms
34 pages

RR-93-22
Manfred Meyer, Jörg Müller:
Weak Looking-Ahead and its Application in Computer-Aided Process Planning
17 pages

RR-93-23
Andreas Dengel, Ottmar Lutzy:
Comparative Study of Connectionist Simulators
20 pages

RR-93-24
Rainer Hoch, Andreas Dengel:
Document Highlighting —
Message Classification in Printed Business Letters
17 pages

DFKI Publications

The following DFKI publications or the list of all published papers so far are obtainable from the above address or via anonymous ftp from ftp.dfki.uni-kl.de (131.246.241.100) under pub/Publications.
The reports are distributed free of charge except if otherwise indicated.

RR-93-25
Klaus Fischer, Norbert Kuhn: A DAI Approach to Modeling the Transportation Domain
93 pages

RR-93-26
Jörg P. Müller, Markus Pischel: The Agent Architecture InteRRaP: Concept and Application
99 pages

RR-93-27
Hans-Ulrich Krieger:
Derivation Without Lexical Rules
33 pages

RR-93-28
Hans-Ulrich Krieger, John Nerbonne, Hannes Pirker: Feature-Based Allomorphy
8 pages

RR-93-29
Armin Laux: Representing Belief in Multi-Agent Worlds viaTerminological Logics
35 pages

RR-93-30
Stephen P. Spackman, Elizabeth A. Hinkelman:
Corporate Agents
14 pages

RR-93-31
Elizabeth A. Hinkelman, Stephen P. Spackman:
Abductive Speech Act Recognition, Corporate Agents and the COSMA System
34 pages

RR-93-32
David R. Traum, Elizabeth A. Hinkelman:
Conversation Acts in Task-Oriented Spoken Dialogue
28 pages

RR-93-33
Bernhard Nebel, Jana Koehler:
Plan Reuse versus Plan Generation: A Theoretical and Empirical Analysis
33 pages

RR-93-34
Wolfgang Wahlster:
Verbmobil Translation of Face-To-Face Dialogs
10 pages

RR-93-35
Harold Boley, François Bry, Ulrich Geske (Eds.):
Neuere Entwicklungen der deklarativen KI-Programmierung — *Proceedings*
150 Seiten
Note: This document is available only for a nominal charge of 25 DM (or 15 US-$).

RR-93-36
Michael M. Richter, Bernd Bachmann, Ansgar Bernardi, Christoph Klauck, Ralf Legleitner, Gabriele Schmidt: Von IDA bis IMCOD: Expertensysteme im CIM-Umfeld
13 Seiten

RR-93-38
Stephan Baumann: Document Recognition of Printed Scores and Transformation into MIDI
24 pages

RR-93-40
Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, Werner Nutt, Andrea Schaerf:
Queries, Rules and Definitions as Epistemic Statements in Concept Languages
23 pages

RR-93-41
Winfried H. Graf: LAYLAB: A Constraint-Based Layout Manager for Multimedia Presentations
9 pages

RR-93-42
Hubert Comon, Ralf Treinen:
The First-Order Theory of Lexicographic Path Orderings is Undecidable
9 pages

RR-93-43
M. Bauer, G. Paul: Logic-based Plan Recognition for Intelligent Help Systems
15 pages

RR-93-44
Martin Buchheit, Manfred A. Jeusfeld, Werner Nutt, Martin Staudt: Subsumption between Queries to Object-Oriented Databases
36 pages

RR-93-45
Rainer Hoch: On Virtual Partitioning of Large Dictionaries for Contextual Post-Processing to Improve Character Recognition
21 pages

RR-93-46
Philipp Hanschke: A Declarative Integration of Terminological, Constraint-based, Data-driven, and Goal-directed Reasoning
81 pages

RR-93-48
Franz Baader, Martin Buchheit, Bernhard Hollunder:
Cardinality Restrictions on Concepts
20 pages

RR-94-01
Elisabeth André, Thomas Rist:
Multimedia Presentations:
The Support of Passive and Active Viewing
15 pages

RR-94-02
Elisabeth André, Thomas Rist:
Von Textgeneratoren zu Intellimedia-Präsentationssystemen
22 Seiten

RR-94-03
Gert Smolka:
A Calculus for Higher-Order Concurrent Constraint Programming with Deep Guards
34 pages

RR-94-05
Franz Schmalhofer,
J.Stuart Aitken, Lyle E. Bourne jr.:
Beyond the Knowledge Level: Descriptions of Rational Behavior for Sharing and Reuse
81 pages

RR-94-06
Dietmar Dengler:
An Adaptive Deductive Planning System
17 pages

RR-94-07
Harold Boley: Finite Domains and Exclusions as First-Class Citizens
25 pages

RR-94-08
Otto Kühn, Björn Höfling: Conserving Corporate Knowledge for Crankshaft Design
17 pages

RR-94-10
Knut Hinkelmann, Helge Hintze:
Computing Cost Estimates for Proof Strategies
22 pages

RR-94-11
Knut Hinkelmann: A Consequence Finding Approach for Feature Recognition in CAPP
18 pages

RR-94-12
Hubert Comon, Ralf Treinen:
Ordering Constraints on Trees
34 pages

RR-94-13
Jana Koehler: Planning from Second Principles —A Logic-based Approach
49 pages

RR-94-14
Harold Boley, Ulrich Buhrmann, Christof Kremer: Towards a Sharable Knowledge Base on Recyclable Plastics
14 pages

RR-94-15
Winfried H. Graf, Stefan Neurohr: Using Graphical Style and Visibility Constraints for a Meaningful Layout in Visual Programming Interfaces
20 pages

RR-94-16
Gert Smolka: A Foundation for Higher-order Concurrent Constraint Programming
26 pages

RR-94-17
Georg Struth:
Philosophical Logics—A Survey and a Bibliography
58 pages

RR-94-18
Rolf Backofen, Ralf Treinen:
How to Win a Game with Features
18 pages

RR-94-20
Christian Schulte, Gert Smolka, Jörg Würtz:
Encapsulated Search and Constraint Programming in Oz
21 pages

RR-94-31
Otto Kühn, Volker Becker, Georg Lohse, Philipp Neumann:
Integrated Knowledge Utilization and Evolution for the Conservation of Corporate Know-How
17 pages

RR-94-33
Franz Baader, Armin Laux:
Terminological Logics with Modal Operators
29 pages

DFKI Technical Memos

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
27 pages

TM-93-01
Otto Kühn, Andreas Birk: Reconstructive Integrated Explanation of Lathe Production Plans
20 pages

TM-93-02
Pierre Sablayrolles, Achim Schupeta:
Conlfict Resolving Negotiation for COoperative Schedule Management
21 pages

TM-93-03
Harold Boley, Ulrich Buhrmann, Christof Kremer:
Konzeption einer deklarativen Wissensbasis über recyclingrelevante Materialien
11 pages

TM-93-04
Hans-Günther Hein:
Propagation Techniques in WAM-based Architectures — The FIDO-III Approach
105 pages

TM-93-05
Michael Sintek: Indexing PROLOG Procedures into DAGs by Heuristic Classification
64 pages

TM-94-01
Rainer Bleisinger, Klaus-Peter Gores:
Text Skimming as a Part in Paper Document Understanding
14 pages

TM-94-02
Rainer Bleisinger, Berthold Kröll:
Representation of Non-Convex Time Intervals and Propagation of Non-Convex Relations
11 pages

DFKI Documents

D-93-15
Robert Laux:
Untersuchung maschineller Lernverfahren und heuristischer Methoden im Hinblick auf deren Kombination zur Unterstützung eines Chart-Parsers
86 Seiten

D-93-16
Bernd Bachmann, Ansgar Bernardi, Christoph Klauck, Gabriele Schmidt: Design & KI
74 Seiten

D-93-20
Bernhard Herbig:
Eine homogene Implementierungsebene für einen hybriden Wissensrepräsentationsformalismus
97 Seiten

D-93-21
Dennis Drollinger: Intelligentes Backtracking in Inferenzsystemen am Beispiel Terminologischer Logiken
53 Seiten

D-93-22
Andreas Abecker: Implementierung graphischer Benutzungsoberflächen mit Tcl/Tk und Common Lisp
44 Seiten

D-93-24
Brigitte Krenn, Martin Volk:
DiTo-Datenbank: Datendokumentation zu Funktionsverbgefügen und Relativsätzen
66 Seiten

D-93-25
Hans-Jürgen Bürckert, Werner Nutt (Eds.):
Modeling Epistemic Propositions
118 pages
Note: This document is available only for a nominal charge of 25 DM (or 15 US-$).

D-93-26
Frank Peters: Unterstützung des Experten bei der Formalisierung von Textwissen
INFOCOM:
Eine interaktive Formalisierungskomponente
58 Seiten

D-93-27
Rolf Backofen, Hans-Ulrich Krieger, Stephen P. Spackman, Hans Uszkoreit (Eds.):
Report of theEAGLES Workshop on Implemented Formalisms at DFKI, Saarbrücken
110 pages

D-94-01
Josua Boon (Ed.):
DFKI-Publications: The First Four Years
1990 - 1993
93 pages

D-94-02
Markus Steffens: Wissenserhebung und Analyse zum Entwicklungsprozeß eines Druckbehälters aus Faserverbundstoff
90 pages

D-94-03
Franz Schmalhofer: Maschinelles Lernen:
Eine kognitionswissenschaftliche Betrachtung
54 pages

D-94-04
Franz Schmalhofer, Ludger van Elst:
Entwicklung von Expertensystemen:
Prototypen, Tiefenmodellierung und kooperative Wissensevolution
22 pages

D-94-06
Ulrich Buhrmann:
Erstellung einer deklarativen Wissensbasis über recyclingrelevante Materialien
117 pages

D-94-07
Claudia Wenzel, Rainer Hoch:
Eine Übersicht über Information Retrieval (IR) und NLP-Verfahren zur Klassifikation von Texten
25 Seiten

D-94-08
Harald Feibel: IGLOO 1.0 - Eine grafikunterstützte Beweisentwicklungsumgebung
58 Seiten

D-94-09
DFKI Wissenschaftlich-Technischer Jahresbericht 1993
145 Seiten

D-94-10
F. Baader, M. Lenzerini, W. Nutt, P. F. Patel-Schneider (Eds.): Working Notes of the 1994 International Workshop on Description Logics
118 pages
Note: This document is available only for a nominal charge of 25 DM (or 15 US-$).

D-94-11
F. Baader, M. Buchheit, M. A. Jeusfeld, W. Nutt (Eds.):
Working Notes of the KI'94 Workshop:
KRDB'94 - Reasoning about Structured Objects: Knowledge Representation Meets Databases
65 Seiten

D-94-12
Arthur Sehn, Serge Autexier (Hrsg.): Proceedings des Studentenprogramms der 18. Deutschen Jahrestagung für Künstliche Intelligenz KI-94
69 Seiten