A Distributed Operationalization of Conceptual Models[1]

Frank Maurer & Gerd Pews
Universität Kaiserslautern
AG Expertensysteme Prof. Richter
Postfach 3049
6750 Kaiserslautern
e-Mail: {maurer, pews}@informatik.uni-kl.de

In this paper we present an interpreter which allows to support the validation of conceptual models in early stages of the development. We compare hypermedia and expert system approaches to knowledge processing and show how an integrated approach eases the creation of expert systems. Our knowledge engineering tool CoMo-Kit allows a "smooth" transition from initial protocols via a semi-formal specification based on a typed hypertext up to an running expert system. The interpreter uses the intermediate hypertext representation for the interactive solution of problems. Thereby, tasks are distributed to agents via an local area network. This means that the specification of an expert system can directly be used to solve real world problems. If there exist formal (operational) specifications for subtasks then these are delegated to computers. Therefore, our approach allows to specify and validate distributed, cooperative systems where some subtasks are solved by humans and other subtasks are solved automatically by computers.

1.0 Introduction and Overview

To evaluate software Boehm [7] distinguishes two goals:

Verification: Does the program fulfil the specified requirements?

Validation: Does the program solve the problem of the customer?

This paper deals with aspects of the second point. We present an interpreter which uses a semi-formal specification in cooperation with human agents to solve real world problems. Therefore, our approach supports the validation of a knowledge-based system in an early development stage. The semi-formal specification is the basis for further development steps.

Rapid prototyping supports the validation of software systems and was the methodology underlying the development of first generation expert systems. The resulting systems often were unmaintainable. Therefore, model-based approaches, e. g. KADS [9], were developed. The main difference to rapid prototyping is that in KADS much effort is focused on

---

1 The work reported here was partially funded by the Ministerium für Wirtschaft und Verkehr of Rheinland-Pfalz within the project "Integration von Hypermedia und Expertensystemen".

the analysis of the problem domain and the specification of the system. The specification is the base for the implementation. In KADS the specification is called *Conceptual Model*. The Conceptual Model consists of the Model of Cooperation and the Model of Expertise.

A drawback of model-based knowledge acquisition is that often users cannot understand and validate (natural language) specifications. Reasons for this are the ambiguity of the natural language and the complexity of the specification. To overcome the ambiguity problem several authors ([1], [25], [26], [24]) proposed formal and operational specification languages for the Model of Expertise. Our approach follows a different line: We developed an interpreter which uses a semi-formal specification in cooperation with the users. Using the interpreter we are able to validate the Model of Expertise *and* the Model of Cooperation.

Our approach is based on a tight integration of hypermedia and expert system technology which is a topic of undergoing research ([5], [6]). Both technologies allow the management and efficient access on knowledge with the help of computers.

Atomic knowledge entities (nodes) in hypermedia systems (cf. [12] or [11]) are typically represented in a format which is not understandable for computers (e. g. video sequences, pictures, audio signals, natural language text). The stored knowledge may be interpreted by the human user depending on the context. Therefore, humans are able to use the knowledge to solve real world problems. To define a connection between two nodes links are used. A user can follow these links to get further information (associative access).

In hypermedia systems, the search for knowledge needed for the problem solving process is controlled by the user[2]. In table 1 we compare in different dimensions the strong and weak points of hypermedia and expert system technology. The assessments are only the edges of a continuum.

2 "guided tours" are an exception to the description given above because there the search is (partially) controlled by the system.

.

Dimension	Hypermedia	Expert System
Degree of the formalisation of the knowledge	Knowledge is only interpretable by humans	Knowledge is formalized and therefore usable by a computer
Initiative in the problem solving process	The user controls the problem solving process which means that he has to fulfil high requirements	The system controls the problem solving process which means that the user has to fulfil lower requirements
Development effort	Medium development effort: Formalisation of the knowledge not necessary; It is not necessary to formalize the background knowledge and common sense of the user	High development effort: The result of the knowledge acquisition must be a complete decontextualisation of the knowledge because inferences can only be made based on formalized knowledge
User interface	Communication with the user via a multimedia interface; the interface is a major research issue	Multimedia user interface only a “Add-On”

Table 1: Comparison of Hypermedia and Expert System Technology

The questions which come up are: What can the approaches learn from each other? How can the approaches be integrated that the advantages are adopted and the weak points are pushed back? First answers are described within this paper.

The second chapter contains an overview how we structure the knowledge for a new application. The resulting hypertext is the basis for our distributed interpreter which is described in the third chapter. Chapter four discusses how a “smooth” development process can be reached and how the progress can be measured. In chapter five we compare our approach with the KADS methodology which was a basis of our research. Last, we summarise our results, describe the state of our implementation and give an overview on ongoing work.

2.0 Structuring the knowledge

In section 2.1 we introduce the CoMo-Kit[3] system. CoMo-Kit supports the development of expert systems in the sense of a computer-aided knowledge engineering tool. Section 2.2 describes how an intermediate representation is developed with CoMo-Kit starting from initial data.

2.1 CoMo-Kit: Conceptual Model Construction Kit

CoMo-Kit supports teams of experts and knowledge engineers in the development of conceptual models. Further, CoMo-Kit allows to define which user is able to solve a task (task distribution, cf. [13]).

3 CoMo-Kit was developed in cooperation with Susanne Neubert, University of Karlsruhe.

CoMo-Kit is based on the HyperCAKE system [16] und ideas of [18]. HyperCAKE uses an extended hypertext abstract machine [11] for the management of multimedia information. HyperCAKE makes it possible to define views on a global hypertext. Hypermedia networks are stored in an object-oriented database and accessible from all workstations in a local area network. CoMo-Kit uses the HyperCAKE system for the management of *all* data which is used within the knowledge engineering process.

To specify an knowledge-based system we use the following terminology[4]:

- **Protocol:** Protocols are the initial data of the knowledge acquisition process. A protocol contains unstructured information which were elicitated from domain experts or other sources of knowledge. Currently, CoMo-Kit supports texts, bitmaps, audio and video.
- **Concept**: Concepts describe the data which is needed for the problem solving process in textual, natural language form. We distinguish, as it is normal in object-oriented design, between class descriptions and instance descriptions. Concept classes are organised in a IS-A-Hierarchy[5]. Each class description includes a set of attributes. For each attribute its value type can be defined.
- **Task:** A task describes in natural language what has to be done to solve a given problem. For each task the input[6] and output data[7] are specified by defining links to concepts. Each task may consist of several subtasks which are organised in a dataflow graph. This means that tasks built up a hierarchy. For each task a set of agents is defined which are potentially able to “do it”.
- **Agent:** Agents are identified by their name and may belong to several groups. Agents may be humans or computers. Agents handle tasks.

For concepts and tasks it is possible to define operational annotations (program fragments). This feature is discussed in “Formalizing the specification”.

2.2 Using CoMo-Kit

Starting point for the conceptual modelling is the protocol of a discussion with an expert who describes the problem at hand in natural language. Figure 1 shows on the left side a list of all protocols of a domain and a protocol editor on the right.The knowledge engineer[8] selects a part of the text which describes a task and chooses the menu entry “create task” to

---

4 We only give an abstract description of the classes which are relevant for this paper.

5 CoMo-Kit supports further relations between concepts PART-Of, CAUSES etc.) which are not described here.

6 Inputs of task may be concept classes or instances. If instances are used in the specification, they are not changed at runtime and represent static knowledge e. g. laws. The input of a task is everything which is needed to solve the problem.

7 Output of tasks are always concept instances because they are modified as a result of the task.

8 In one example (Baunutzungsverordnung = rules for building houses) of CoMo-Kit the modelling of the domain is partially done by city planners, which means by the experts. We think that this may be transferred to other domains because there is no need for learning a programming language.

create a new task node. Correspondingly, concepts and agents can be created. We implemented graphical interfaces to define the membership of an agent to a group and the concept class hierarchies.

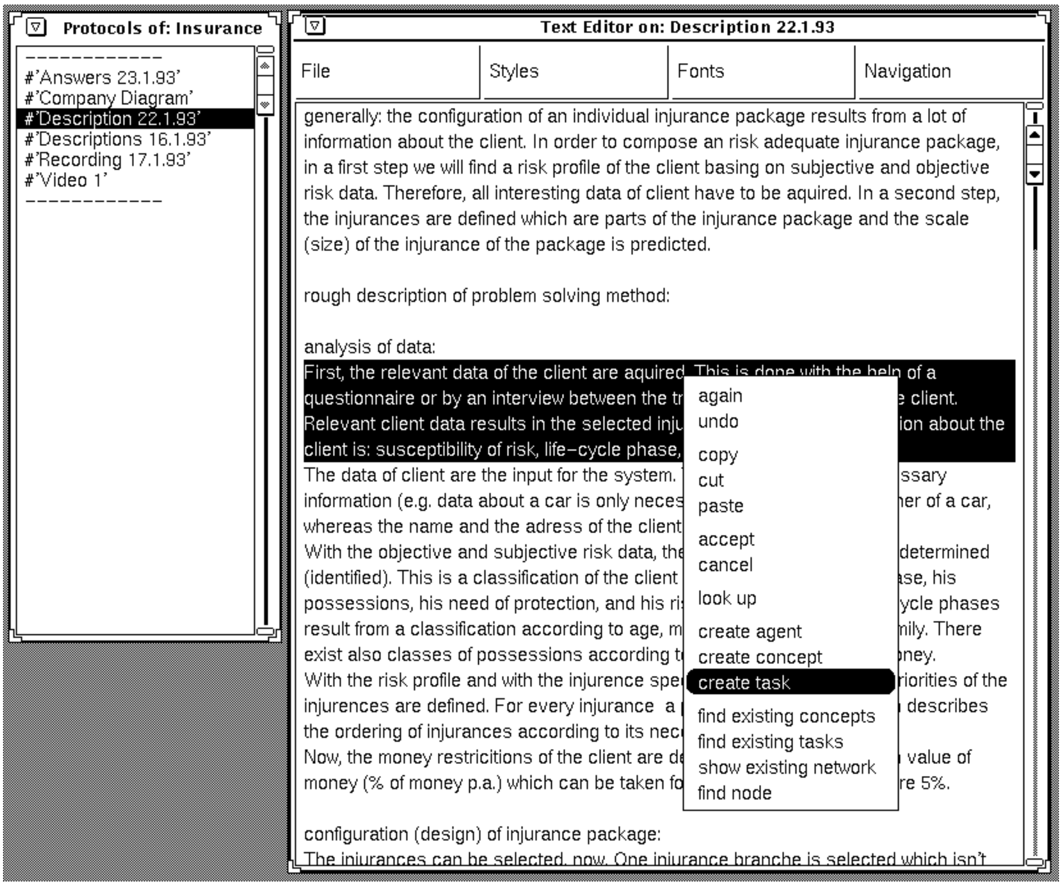

Figure 1: The list of protocols and a protocol editor

Figure 2 shows a task hierarchy. Additionally, one can see the task which was created in Figure 1. The text which was selected in the protocol is copied into the new node. The knowledge engineer can edit the text to specify the task more exactly. The inner structure of a task is specified as a dataflow diagram (cf. Figure 3). Concepts are shown as rectangles whereas ellipses represent tasks. The hierarchy of data flow diagrams is the base of our interpreter which is described in the following chapter.

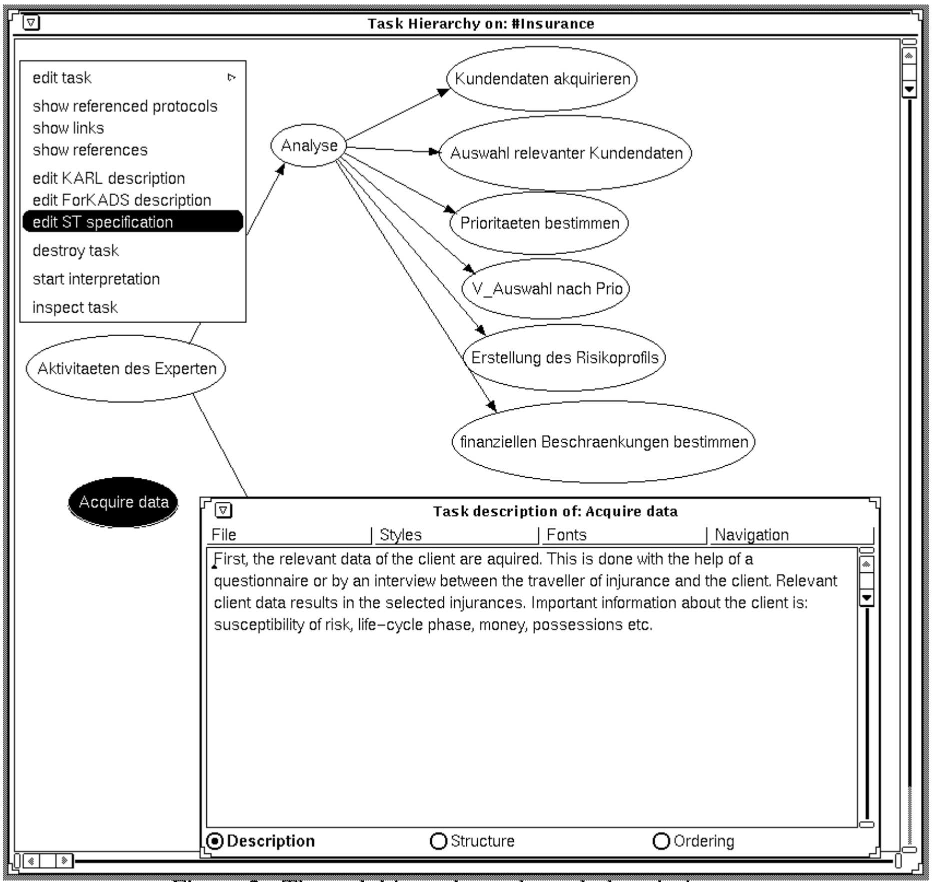

Figure 2: The task hierarchy and a task description

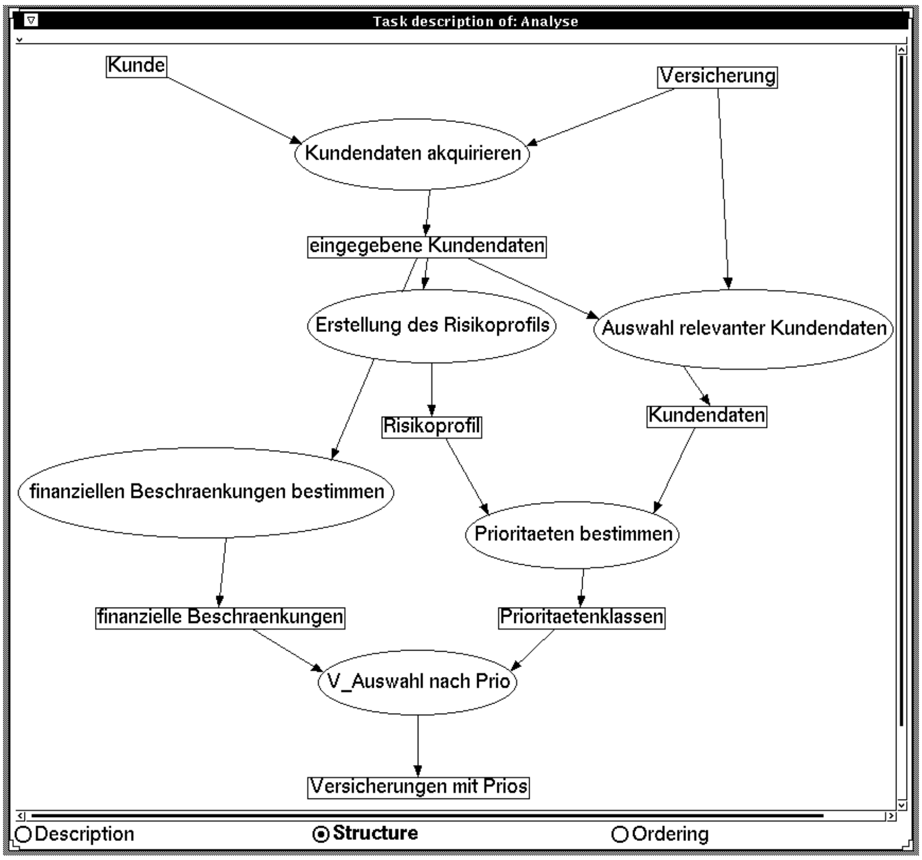

Figure 3: A task structure

3.0 The interpreter for hierarchical task structures

In [13], advantages of a “Wizard of Oz”-experiment in evaluating the cooperation between humans and computers are explained. Guided by the interpreter which is presented in this section, users are enabled to validate directly a task which has just been decomposed. So, a (slightly modified) “Wizard of Oz”-experiment is carried out by this interpretation. Direct questions to the expert are made impossible in order to find out whether the task is sufficiently described to be executed by the user or not. For him, the only data available is the task´s description, it´s inputs and his own knowledge.

Our semi-formal[9] hypertext structuring of the domain leads to usable results: Complex tasks are split to several subtasks which can - under guidance of the interpreter - be executed by (lower qualified) users which can access all task-relevant data. Non-relevant data is hidden.

A survey of distributed task-execution is given in Figure 4. A (privileged) user starts a task by delegating it to one ore more users and by unlocking it. Doing this, he starts a scheduler-process. The scheduler generates instances of all data[10] which will be modified during the interpretation run. Further, the scheduler will only allow to interpret a task if all input data are accessible (generated by the interpretation of previous tasks). So, if a certain user

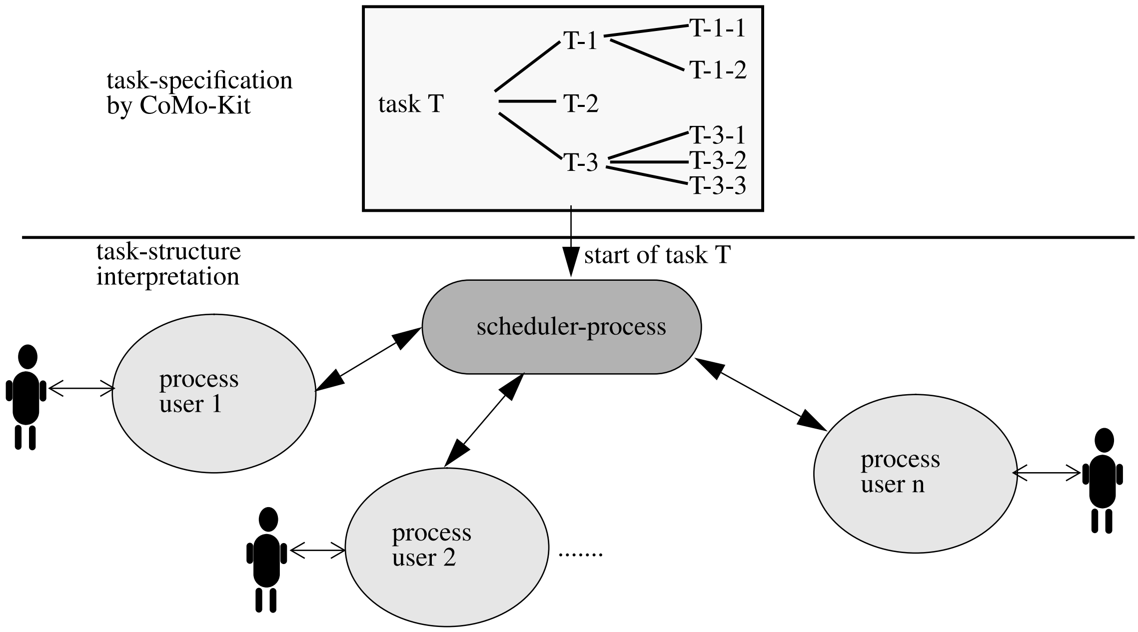

Figure 4: The interpreter´s process-structure

9 Semi-formal, here: The network´s topology is formal (Each kind of node or link has a certain meaning for problem solving determined by the interpreter). Nevertheless, a node´s contents can only be interpreted by a human user.

10 A task can be multiply interpreted several times in parallel (e.g. choosing insurances for different persons).

wants to work on a task, he is free to choose it from a whole pool of tasks delegated to him. These tasks can be divided in two groups: complex tasks and atomic tasks.

Complex tasks consist of several subtasks. They are represented as inner nodes of the tree shown in Figure 2. If a user works on a complex task, he becomes some kind of a manager. In our case, a manager has to

- delegate subtasks to other users and
- supervise their execution

Doing this, he is assisted by the system. Tasks which can only be interpreted by a single person (or by a computer program) are transfered directly to their executor. Additionally, the state of task execution can be displayed, for instance in order to re-delegate it. A control window for usage by the manager is shown in Figure 5.

This window consists or four lists, containing:

- uninterpreted tasks (upper left)
- delegated tasks waiting to be executed (bottom left)
- tasks which are just being interpreted (bottom right)
- tasks whose interpretation has been finished (upper right).

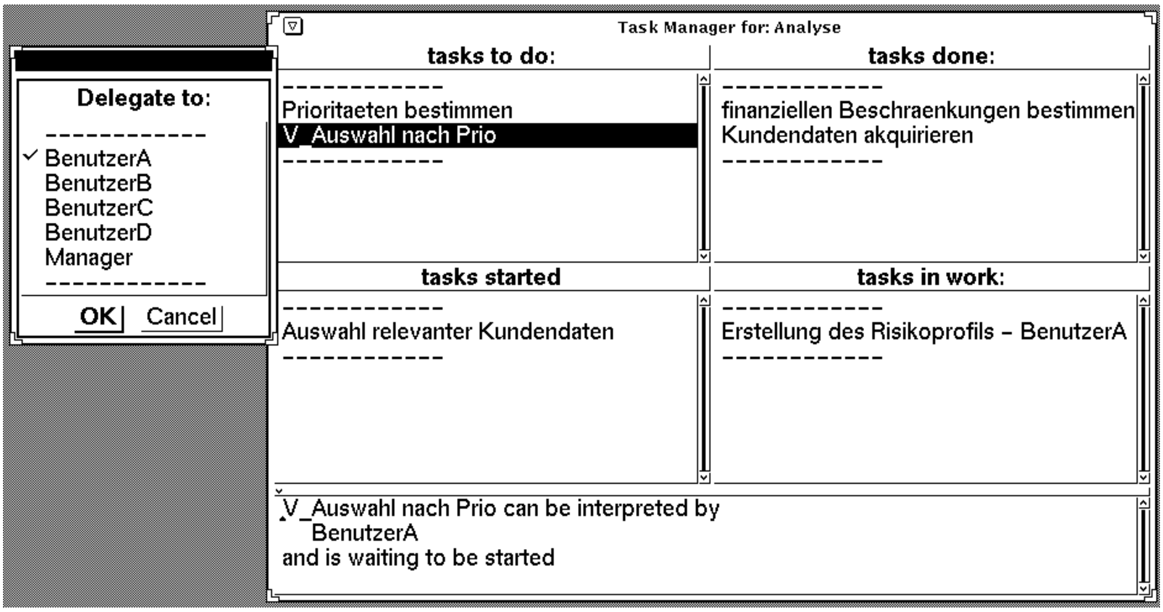

Figure 5: Distribution of tasks among agents

An information space is placed below these lists in order to display textual information about each task´s state of execution.

The window title displays the task the manager currently works on (here: analysis). In the upper left subtasks are shown which are not yet distributed among the agents. They will be delegated to one or more users by selecting them from the dialogue box. After confirming the selection, the subtask appears in the bottom left list. The bottom right list consists of tasks which are just in work by an agent. The agent´s name is added to the task name here. If the execution of this subtask was successful, it moves to the list above, which gives a survey over all executed tasks. So, a subtask moves counterclockwise through the four lists. The manager is enabled to control the execution process and eventually intervene in case of stagnation.

Atomic tasks are solved using the window shown in Figure 6. To the left, dynamically generated buttons are displayed. Each stands for task-relevant[11] data accessible by pressing it. The central part of the window holds a description of the task itself. Basing on this, the user is enabled to edit the results in the editor[12] on the left. Pressing the “Propagate”-button, editings are confirmed and propagated to the scheduler. So, tasks which follow this task in the dataflow diagram and which are relying on it´s outputs can be started next.

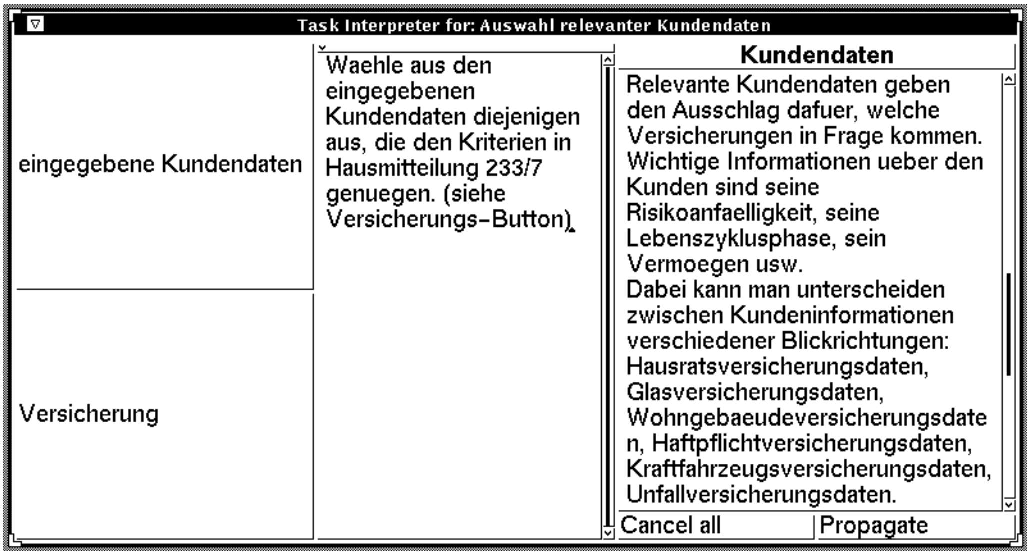

Figure 6: Editor for execution of an atomic task

There are two features of the described interpreter which are of special interest:

1. Knowledge needn’t be formalized completely. We represent the knowledge in a hypermedia network. Basing on the operational semantics of nodes and links, the interpreter uses the network´s structure to support the problem solving process. Interpretation of node-contents is performed by the user. This only is a gradual difference to interactive expert systems, because each question directed to the user will be interpreted by the

11 The relevance of information for executing a certain task has been determined while structuring the domain: Inputs of a task are relevant for it´s solution.

12 If the task produces more than one result, an editor is shown for each output.

user. Following this thought implies that *there is no exactly defined border between hypertexts and expert-systems. Therefore, a "smooth" transition between both kinds of knowledge representation is achievable.*

2. Problem solving is distributed on several agents (humans or computers). So, the interpreter supports validation of the Model of Cooperation.

4.0 Formalizing the specification

The interpreter for task structures guides the user in interactive problem solving. It controls the overall process. Atomic tasks are solved by a human who uses the information stored in the hypertext.

If one transfers Searl's Chinese Room Experiment into our context one can see that the described interpreter is only partially able to "translate chinese". For sentences which are not stored in the lexicon the interpreter has to ask an external human agent.[13] We call an inference engine which delegates tasks which he does not understand to external agents *semi-formal*. Different semi-formal mechanisms can be partially ordered by the number of tasks which can be solved automatically without using an external agent. This ordering also gives us a mean to measure the development progress. A goal of the development of a expert system is to descend the partial ordering as deep as possible. Then the inference mechanism solves many problems by itself and delegates only a few to a human.

Some steps in the development process consist of the formalization of atomic tasks. If there exists a formal specification for a task it will be solved by an agent "computer" which easily may be integrated in the schema described above. To support the maintenance of the system the formalized knowledge should preserve the structure of the specification ("structure preserving design"). For this reason we use a formal specification language which is directed to conceptual modelling. CoMo-Kit supports two of these languages: One is based on Smalltalk-80 and will be described in the next paragraphs. The second is KARL [1] and was integrated because of our cooperation with the group of Prof. Studer, university of Karlsruhe. CoMo-Kit allows to define formal annotations for concepts and tasks in these two languages.

The step from a semi-formal specification to operational programs is divided into two parts. First, the knowledge engineer defines the type of a task. This step is based on the idea of generic tasks which is a key concept of several knowledge engineering

13 This analogy goes back to Prof. Richter.

approaches.[14] CoMo-Kit supports a typology of knowledge sources which helps the development process in different ways:

- The type of a task (e. g. “select”, “select-best”, “match” etc.) gives hints for the design and the implementation of the task.
- The type of a task allows to infer what information is needed to solve the problem.[15]

Second, the task description must be translated by the knowledge engineer into a formal language. For this purpose we use object-oriented technology. For each task type we defined a corresponding formal equivalent (cf. Figure 7):

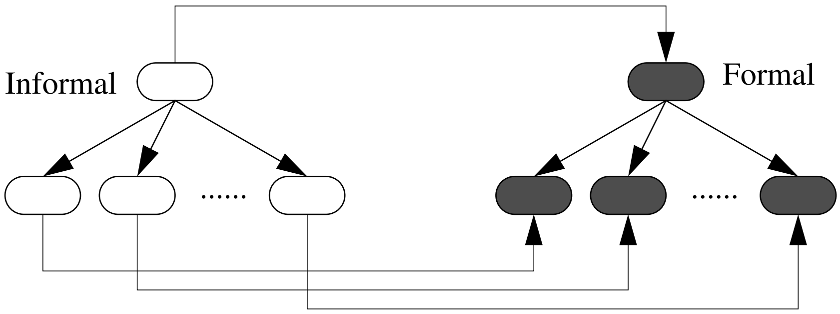

Figure 7: Corresponding class hierarchies for formal and informal task descriptions

The menu entry “edit ST specification” (cf. Figure 2) creates a node which stores the formal annotation of a task. The class of the node depends on the type of the task. The parameters of the formal task description are automatically generated as specified and cannot be edited by the user. The user has to fill the task body with Smalltalk-80 code (cf. Figure 8).

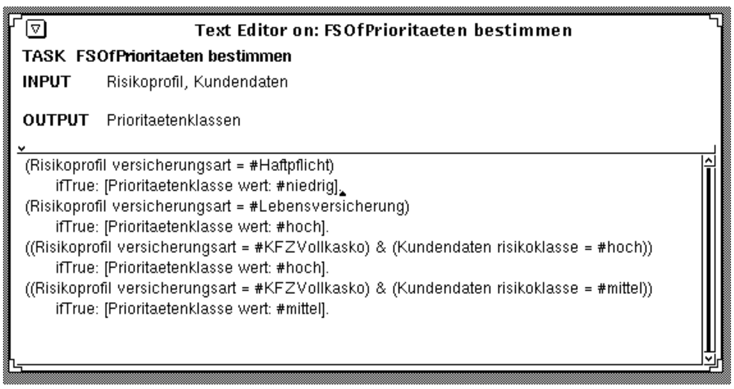

Figure 8: The edtitor for formal task descriptions

CoMo-Kit supports a smooth transition from semi-formal to formal specifications: The user is not forced to formalize the whole task hierarchy but is allowed to choose atomic

14 e. g. “Generic Tasks” by Chandrasekaran and the typology of primitive knowledge sources in KADS.

15 This is a base of the Protege-System (cf. [18]).

tasks which are annotated with Smalltalk-80 code. The replacement of an atomic task by program code is abrupt. Nevertheless, every atomic task is only a very small part of the overall system specification. Therefore, the development of the whole system can be considered to be smooth.

To summarize the ideas of our operational specification language:

- Based on the name of the task we automatically generate the name of the formal annotation.
- The specified inputs and outputs of a task determine the names of the formal parameters of a task (see the upper part in Figure 8).
- The body of the task has to be defined by the knowledge engineer and consists of a list of Smalltalk-80 commands.
- The formal parameters can be used inside the body. The interpreter binds the parameters with the appropriate values at runtime.
- If the type of the task determines the task body CoMo-Kit asks for the needed knowledge. For example, a select-task need only the condition *what* objects must be selected. The knowledge engineer is not forced to determine *how* the selection is done.
- The interpreter checks at runtime if the output of the task is of the specified concept class.

Putting it all together CoMo-Kit supports a smooth development process starting from protocols via a hypertext-based intermediate representation to a operational implementation of distributed knowledge-based systems. The knowledge engineer is not forced to implement the knowledge completely but the implementation of the knowledge can be done *partially.* This means that in the course of the development process it can be decided which subtasks should be implemented and which will be carried out by human agents. The decision is determined by the cost of the implementation[16] and the expected benefits. The financial risk of a wrong decision is reduced because the development follows the Spiral Model (cf. [8]).

5.0 Comparison with the KADS methodology

Our approach is basing on the KADS methodology which - especially in Europe - comes more and more to the foreground. A more detailed description of KADS can be, for instance, found in: [22], [28], [29] or [9].

In KADS, the description of a knowledge-based system is split up into several models: Organizational Model, Application Model, Task Model, Conceptual Model and Design Model.

---

16 Based on the specification the costs can be estimated more precisely compared to the project start.

The result of a domain analysis is named Conceptual Model. This Conceptual Model consists of an user-interface description (Model of Cooperation) and a knowledge model (Model of Expertise). The Model of Expertise deals with four kinds of knowledge, each associated to a different layers. These are: domain-, inference-, task- and strategy-level.

Basing on KADS, our approach differs from KADS itself by several items:

- **Hierarchical Inference Structures**: In our approach, inference structures do not only consist of primitive inferences (knowledge sources) as KADS-structures do. Using hierarchical dataflow charts, they may be described from different levels of abstraction. This approach to modelling includes aspects of the Task Model as well as aspects of the inference layer. Separating aspects of controlling from aspects of inference is unnatural in our eyes, because a priori, it may not be obvious whether a task is primitive or complex[17]. Moreover, hierarchical inference structures are necessary for a modular structuring of the specification and a subsequent implementation.
- **Tool Development**: Developer support by a knowledge acquisition tool is an inherent part of our approach. It depends on tools, what is possible to be produced; and methodologies without tools are not applicable in practice[18]. This becomes obvious when looking at an analogy in mechanical engineering: It makes a fundamental difference what is being used in production: a CNC-machine-tool or a flint axe. So, developers of a methodology will automatically consider which tools are available. If a pencil and a piece of paper are the only resources for working out the specification, an "optimal" method will surely be different from a method which is developed to be executed on a computer. Anyway, tool development in KADS seems to be considered less important (in spite of Shelley´s development [2]).
- **Incremental Development**: Our approach uses hypermedia-networks as an intermediate representation and uses them directly for problem-solving. This supports an incremental development of knowledge-based systems according to Boehm´s Spiral Model.
- **User Interface**: Generating multi-media user interfaces is relatively easy, since our system is basing on an object-oriented hypermedia-system. Rapid development of user interfaces is supported, too.
- **Knowledge Repository**: All information used in the knowledge engineering process is held and handled in a global data base (knowledge repository). The knowledge repository is the analogous to a data dictionary in conventional software engineering tools.
- **Strategy Layer**: The Strategy Layer is meant to hold meta-knowledge about selection and combination of tasks. It hasn't been specified clearly till now, hence it is not supported by our tool.

---

17 Describing Case-Based Reasoning as a KADS-Model (cp. [3]), one discovers that the knowledge source "select best case" will be split up to a complex structure as soon as the specification becomes precise enough for implementation. This effect will be found in many of the models described in [4].

18 Vice versa, a tool has to be based on a method.

- **Interactive Simulation**: In cooperation with the users, the described interpreter processes a semi-formal specification of the knowledge-based system; that means: validation of the knowledge-based system becomes possible without a previous formalisation of the domain. So, problems can be detected in early stages of development and costs can be reduced.
- **Validating the *Conceptual* Model**: In contrast to formal specification languages (KARL, ML^2, ForKADS, MoMo), our approach supports validation of Conceptual Models, not only of Models of Expertise; that means assignment of tasks to agents can be checked with the described interpreter, too.

6.0 Summary and ongoing work

In this paper, we demonstrated how knowledge-acquisition and -structuring can be supported by integrating hypermedia- and expert-system-techniques. We described an interpreter which makes it possible to validate a semi-formal specification of a knowledge-based system. So, in early stages of development it can be recognized whether the system solves the given problem or not. A “smooth” development process is provided by the (incremental) replacement of tasks by operational specification.

HyperCAKE and CoMo-Kit are both completely implemented and linked with the object-oriented data base GemStone from Servio Cooperation. The task-structure interpreter is implemented as single-user system, multi-user implementation will be finished soon. The System is implemented in Smalltalk-80, Rel. 4.1 by ParcPlace Systems. So, single-user versions (without audio- and video-interfaces which depend on the current computer platform) are running on several Unix-workstations (Sun, HP/Apollo, IBM, Dec), 386 PCs and Apple Macintosh.

HyperCAKE/CoMo-Kit are the basis for the development of several expert-system-shells: SAFRaN combines a geographical information system (GIS) and an expert system to provide a knowledge-based interpretation of maps. SAFRaN and an appertaining application are described in [10], [14] and [15]. HyDi ([23]) supports the development of hypermedia-based diagnosis systems and was completed in the beginning of 1993.

Acknowledgements

Prof. Richter gave important impulses for the design of this paper. We´d also like to thank Susanne Neubert for many intensive discussions about using hyper-media in knowledge acquisition. Our colleagues in the research group on expert systems of Prof. Richter supported our work by a pleasant organization climate and their aid as proof readers.

Literature

[1] Angele, J.; Fensel, D.; Landes, D.; and Studer, R: KARL: An Executable Language for the Conceptual Model. In: Proceedings of the Knowledge Acquisition for Knowledge-Based Systems Workshop KAW'91, October 6-11, Banff, 1991

[2] Anjewierden, A., Wielemaker, J., Toussant, C.: Shelley - computer aided knowledge engineering, 1992, in [22]

[3] Bartsch-Spörl, B.: A Simple Interpretation Model for Case-Based Reasoning, 1992, in [4]

[4] Bauer, Ch., Karbach, W.: Proc. 2nd KADS User Meeting, Siemens AG, 17-18. Feb. 1992

[5] Bielawski, L., Lewand, R.: Intelligent Systems Design, Wiley 1991

[6] Biethahn, J.; Bogaschewsky, R.; Hoppe, U. (Hrsg.): Expertensysteme in der Wirtschaft 1992 - Anwendungen und Integration mit Hypermedia, Gabler Verlag, 1992

[7] Boehm, B. W.: Software Engineering Economics, Englewood Cliffs NJ: Prentice-Hall, 1981

[8] Boehm, B. W.: A Spiral Model of Software Development and Enhancemant, Computer, 21, 5 (May 1988), p. 61-72

[9] Breuker, J.; Wielinga, B.; Someren, M.v.; de Hoog, R.; Schreiber, G.; de Greef, P.; Bredeweg, B.; Wielemaker, J.; and Billault, J.-P.: Model-Driven Knowledge Acquisition: Interpretation Models. Esprit Project P1098, University of Amsterdam (The Netherlands), 1987

[10] Burde, M.: Die langfristige Sicherung von Grundwasservorkommen - durch die Ausweisung von Grundwasservorranggebieten - als gemeinsame Aufgabe von Raumplanung und Fachplanung, Unveröffentliches Manuskript; Dissertation im Fachbereich ARUBI, Universität Kaiserslautern, 1992

[11] Campbell, B., Goodman, J. M.: HAM: A General Purpose Hypertext Abstract Machine, Communications of the ACM, July 1988, Vol. 31, No. 7

[12] Conklin, J.: Hypertext: An introduction and survey. Survey and tutorial series. IEEE Computer, September 1987, S. 17-41

[13] de Greef, H. P., Breuker, J. A.: Analysing system-user cooperation in KADS, In [22]

[14] Hemker, H..: Entwurf und Implementierung eines Expertensystems mit GIS-Kopplung, Diplomarbeit Uni Kaiserslautern, 1992

[15] Jäckel, Th.: Entwurf und Implementierung einer Benutzeroberfläche für ein Expertensystem unter Berücksichtigung planerischer Vorgehensweisen, Diplomarbeit Uni Kaiserslautern, 1992

[16] Maurer, F.: HyperCAKE: Ein Wissensakquisitionssystem für hypermediabasierte Expertensysteme, 1992, in [6]

[17] Maurer, F., Pews, G.: Hypermedia als Zwischenrepräsentation bei der Expertensystementwicklung, Proc. Hypermedia 93, Springer Verlag, 1993

[18] Musen, M.: Automated Generation of Model-Based Knowledge Acquisition Tool, Pitman, 1989, London

[19] Neubert, S.: Einsatz von Hypermedia im Bereich der modellbasierten Wissensakquisition, 1992, in [6]

[20] Nielsen, J.: Hypertext & Hypermedia. Academic Press, San Diego, London, 1990

[21] Sommerville, I.: Software Engineering, Addison-Wesley, 1992 (Fourth edition)

[22] Schreiber, G. (Ed.): Special Issue: The KADS approach to knowledge engineering, Knowledge Acquisition, Vol. 4 No. 1, March 1992, Academic Press

[23] Traphöner, R., Maurer, F.: Integrating Hypermedia and Expert System Technology for Technical Diagnosis, Proc. Expersys 92, Paris, 21.-22. Okt. 1992

[24] van Harmelen, F., Balder, J.: $(ML)^2$: A formal language for KADS models of expertise, 1992, in [22]

[25] Walther, J., Voß, A., Linster, M., Hemmann, Th., Voß, H., Karbach, W.: MoMo, Arbeitpapiere der GMD 658, Juni 1992

[26] Wetter, Th.: First-order logic foundation of the KADS conceptual model, 1990, in [27]

[27] Wielenga, B., Boose, J., Gaines, B., Schreiber, G., van Someren, M. (ed.): Current Trends in Knowledge Acquisition, IOS Press, Amsterdam, Mai 1990

[28] Wielinga, B.J.; Schreiber, A.Th.; Breuker, J.A.: KADS: A Modelling Approach to Knowledge Engineering. ESPRIT Project P5248 KADS-II, An Advanced and Comprehensive Methodolgy for Integrated KBS Development, Amsterdam, 1991

[29] Wielinga, B.J.; Schreiber, A.Th.; Breuker, J.A.: KADS a modelling approach to knowledge engineering, 1992, in [22]