Towards an Epistemology of Intelligent Problem Solving Environments: The Hypothesis Testing Approach

Claus Möbus

Department of Informatics
Learning and Teaching Systems
C.v.O. University, 26111 Oldenburg, Germany
E-Mail: Claus.Moebus@informatik.uni-oldenburg.de

Abstract: The main purpose of intelligent problem solving environments (IPSEs) is to offer students the opportunity to acquire knowledge while working on a sequence of problems chosen from the domain. Up to now we have developed IPSEs for various curricula and applications (computer science, hydraulics, chemistry, economic simulation games and causal modeling). On the surface being very different all IPSEs follow a common design theory: the student should acquire knowledge by testing his own hypotheses.

First we want to show that hypothesis testing plays a fundamental role in a *cognitive science* orientated theory of knowledge acquisition (ISP-DL-theory). This theory is the basis of our IPSEs. In a case study three IPSEs and their relationship to hypothesis testing are discussed. Then we define the concept of hypothesis testing in a *logic* framework. We describe knowledge acqusition events and learning effects. It is argued that knowledge acquisition stimulated by IPSEs is based fundamentally on selfexplaining the responses of the IPSE to the student's hypotheses.

Introduction

It has been well recognized that the development of intelligent help systems raises difficult questions, like: How is help and instructional material to be designed? When should remedial information be supplied? Why is the same information useless to one person and helpful to another? Existing intelligent tutorial and help systems have not always provided satisfactory answers to such questions. For example, the information delivered to the learner may assume too little or too much knowledge, the user interaction is too restrictive, or tutoring and help strategies are unprincipled and ad hoc. These shortcomings are basically still true (Self, 1990). To make some progress a theoretical framework seems to be necessary. It should be sufficiently detailed to enable specific design decisions and predictions. At the same time it should be so general that it is applicable to different domains. This paper is a first step in that direction: we try to describe the epistemology of IPSEs.

From our point of view Intelligent Problem Solving Environments (IPSEs) seem to be the most cost effective intelligent systems for the communication of problem solving knowledge. Though they contain an expert system or an oracle that can check the correctness of students' solution proposals, they lack other expensive components like a teaching or a student model. The curricular component in form of a teaching model is abandoned in favor of a simple sequence of task-relevant problems. The student model which should be responsible for the individualization of system responses is missing, too. Instead of that individualization is achieved by the ability of the system to respond intelligently to student hypotheses. In IPSEs an expert system (or an oracle) and the current student hypothesis are sufficient to generate adaptive help.

To avoid design errors the design of IPSEs should be guided by a *psychological* theory of knowledge acquisition. Our work is based on ISP-DL-Theory, an acronym for "Impasse-Success-Problem-Solving-Driven-Learning". ISP-DL is influenced by the cognitive theories of ANDERSON (Anderson, 1986, 1989), NEWELL (1990) and Van LEHN (1988) as well as by the motivational "Rubikon" theory of HECKHAUSEN (1987, 1989) and GOLLWITZER (1990). The theory is sketched in part 2 and design principles for IPSEs which can be (informally) derived from it are discussed in part 3.

To demonstrate the feasabilty of these ideas three case studies from different domains (derivation of functional programs, modeling time-discrete distributed systems, room configuration tasks) with very different (monotonic and nonmonotonic) problem spaces are presented in part 4. It is shown how close one can stick to a special design philosophy despite differences in knowledge domains and despite the use of very different AI-techniques (informed search, rule-based grammars, model checking and inductive learning).

In part 5 we summarize and abstract the results of the case studies. We define formally the concept of a hypothesis in a knowledge revision framework. We show that *hypothesis testing* can be integrated into *theory revision* and *knowledge acquisition* processes of an abstract problem solver. We discuss the question *when* knowledge acquisition events will happen and *what* kind of knowledge is acquired when working with these IPSEs. We present our main hypothesis that knowledge acquisition stimulated by IPSEs is based fundamentally on selfexplanation: the student should try to selfexplain the responses of the IPSE to his hypotheses.

ISP-DL: A Theory of Knowledge Acquisition

From our own empirical investigations (Möbus & Schröder, 1993) we concluded that it is fruitful to describe learning as an interplay of impasse- and success-driven learning. In particular, we developed a model based on these concepts which closely simulates the continuous stream of actions and verbalizations of a single subject while working on a sequence of problems (Möbus, Schröder & Thole, 1995). Further development led to the ISP-DL Theory (Möbus, Schröder & Thole, 1994) which is intended to describe the stream of actions and cognitive processes occurring in problem solving situations. ISP-DL Theory has three aspects:

- The distinction of *different problem solving phases* (Heckhausen, 1989; Gollwitzer, 1990). In the *deliberate* phase the problem solver considers several goals and finally chooses one. In the *plan* phase a solution plan is developed to obtain the goal. Subgoals are created and sequenced. Then the plan is executed, or *implemented*. Finally the problem solver *evaluates* the result.
- The *impasse driven acquisition of new knowledge* (Laird et al., 1987; Van Lehn, 1988; Newell, 1990). When knowledge is not sufficient to implement the goal an impasse occurs. In response to an impasse, the problem solver applies according to the theory weak heuristics, like asking questions and looking for help. Thus the learner obtains new information. As a result of this, the learner may overcome the impasse and acquire new knowledge. Thus impasses trigger the *acquisition* of knowledge. But the new information may cause a secondary problem.
- The *success driven improvement of existing knowledge*. Successfully used knowledge will be improved: e.g. by rule composition (Anderson, 1986, 1989), which can be based on the resolution method (Möbus, Schröder & Thole, 1994). Thus the number of control decisions and subgoals can be reduced.

We formalized ISP-DL theory with higher order petri nets (Huber et al., 1990). A sketch of the theory is shown in figure 1. Learning has two aspects: the process of knowledge optimzation occurs after a solution has been found. This process is *deductive* in the sense that the new optimized knowledge is a logical consequence of old knowledge:

$$\textbf{background knowledge} \cup \textbf{evidence} \models \textbf{optimized knowledge}$$

The more interesting knowledge acquisition process occurs after solutions have been found with the help of heuristics. This process is *inductive*:

$$\textbf{background knowledge} \cup \textbf{new knowledge} \models \textbf{evidence}$$

so that heuristics can be seen as inductive inference rules.

When do we expect hypothesis testing activities? We assume that the problem solver has a solution proposal for the given task. This is evaluated by mental or real time simulation or asking an oracle (eg. the IPSE). When there is negative feedback the student realizes an impasse. The reaction to that is planning and use of weak heuristics. One of them is testing a hypothesis: that means asking the IPSE-system questions concerning the solution status of parts of the original defective proposal: "Is this part of my solution proposal embeddable in a correct solution?".

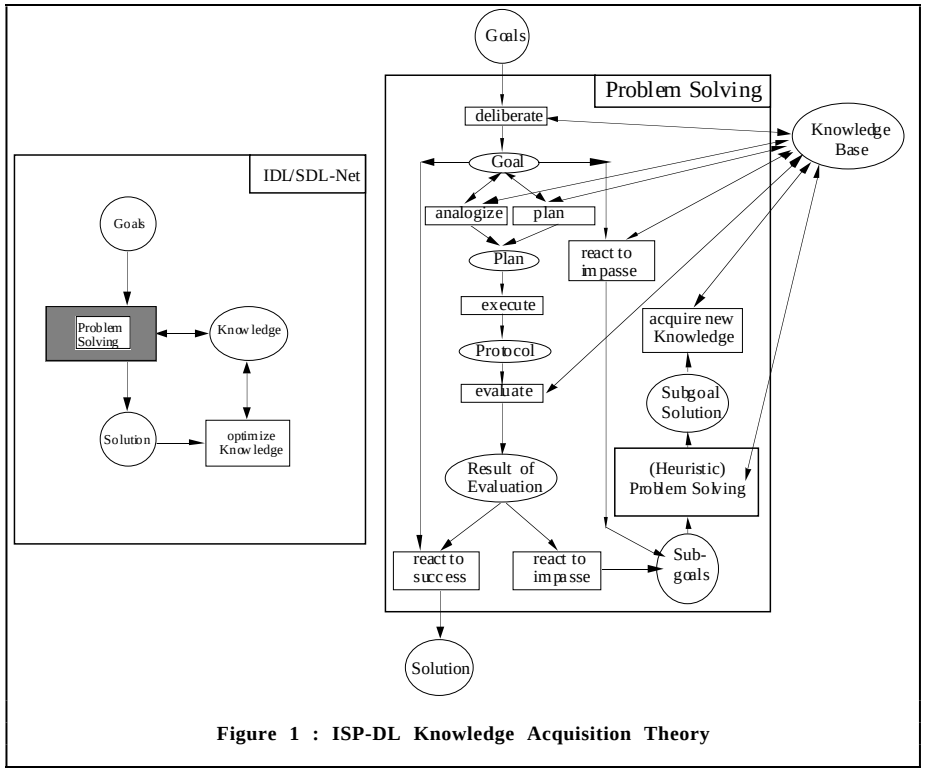

Figure 1 : ISP-DL Knowledge Acquisition Theory

Principles of Design for IPSEs Based on ISP-DL Theory

The ISPDL Theory motivates the following principles:

(1) The IPSE should *not constrain and interrupt* the problem solver but offer information only on demand. According to the theory, information is only helpful at impasse time. This principle is in contrast to active help systems with immediate feedback. We think that it is important first to let the learner develop her/his own solution ideas and then later optimize his solutions. As novices are rather "creative" in generating unusual solutions the systems should be sufficiently powerful.

(2) The student should have the opportunity to obtain detailed feedback and information *any time*. Since impasses are possible at *different phases* of problem solving (figure 1 gives only a simplified sketch of the theory without any recursions), the system must offer support in the problem solving phases *planning*, *implementation*, and *evaluation*.

(3) The learner should be enabled to *make use of her/his pre-knowledge* as much as possible when asking for help. Thus the information provided should be conditional to his hypotheses and preknowledge to avoid follow-up impasses.

(4) The information provided should in *grainsize and amount* be *tailored to the knowledge state* of the problem solver. If the grainsize of the information is too fine or too coarse and the amount not synchronized to the knowledge deficit, then the problem solver has to filter or generate new information which can have undesirable emotional effects preventing any progress. An (expensive) *student model* is needed only iff there is a set of help alternatives to *choose* from.

(5) It is necessary that the learner is free in the choice of his problem solving operators and her/his interaction modality. We should offer an IPSE for *free and unconstrained problem solving.*

Case Studies

Now we want to describe three systems (figures 2-4) which are designed according to ISP-DL theory and which enable hypothesis testing for the student. The first two IPSEs were developed for computer science and the third for health care curricula.

The idea of a hypothesis testing environment was first developed for the ABSYNT system (Möbus, Thole & Schröder, 1993a). Figure 2 shows a typical problem solving state: the reversal of a list. The solution proposal contains operators and planning/goal nodes.

The next system is PETRI-HELP. The system supports the modeling of distributed time discrete systems (eg. traffic lights, production plants, libraries, telephone nets) with simple condition-event nets. The transition of nets can be compared to productions in a production system (Zisman, 1978). Figure 3 shows a solution proposal to model the photosynthesis process.

The third system IKEA was developed as an IPSE within a classical CBT-course for the catholic care organisation CARITAS. The CBT-course should train service personal for elderly handicapped people. One of the tasks in the training course consisted in communicating knowledge how to configure a room for a person who needs help in every day live. Figure 4 shows the room with some regions (door, sun, window, washing, draught) and some furniture already placed

The three knowledge domains differ eg. with respect to the availability of expert knowledge. In functional programming *expert knowledge* is in principle available to derive correct solutions from a formal specification though the programmer may not use it. In Petri-net modeling *no expert knowledge* is available for the correct deduction of nets from temporal logic specifications, though it is possible to check the correctness of a students' solution proposal: model checking. Students solve the problem only with rules of thumb or with heuristics. In configuring rooms a subset of the experts knowledge belongs to *commonsense knowledge* (eg.: a bed should not positioned in the draught region; a tv-set should not face the wall, etc). This knowledge is sometimes intuitively available to the student.

ABSYNT

ABSYNT ("Abstract Syntax Trees") supports programming novices with help and proposals while they acquire functional programming concepts including recursion. ABSYNT was designed to encourage explorative learning. The ABSYNT system consists of four main parts: (1) a *visual editor* for constructing programs. ABSYNT programs consist of trees built from connected primitive and self-defined operator nodes, parameters, and constants. In addition program plans can be constructed using *goal nodes*.. (2) A *visual trace* makes each computational step of the ABSYNT interpreter visible. (3) In a *diagnosis-, hypotheses- and help environment* the learner may state the hypothesis that her/his solution proposal (or part of that proposal) to a programming task is correct. The system then analyses the part of the solution proposal chosen by the student as a hypothesis. As the result, the system gives help and error feedback on the *implementation* and *planning level* by synthesizing complete solutions for the given programming tasks, starting from the learner´s hypothesis. If the hypothesis is embeddable within a complete solution, the learner may ask for completion proposals.

Figure 2 shows the program "list-reversal". Programs are a tree representation of mixed terms. Terms are mixed when they contain runnable operators (round shaped nodes) and not runnable specification terms (cloud shaped nodes). The figure shows a hypothesis stated by the learner (bold parts of the proposal in the upper window) The hypothesis means: "Is the bold marked part of the solution proposal embeddable in a correct solution?" The system generates a complete solution but to offer minimal help information only one node is shown to the learner to stimulate self explanation (Chi et al., 1994). How-, Why-, and Whynot-Explanations are available on demand, too.

PETRI-HELP

In the PETRI-HELP project (Pitschke, 1994, Schöder et al. 1995), a system is developed for supporting problem solvers in the domain of modelling with condition-event Petri nets. Like in ABSYNT, the system will provide help sensitive to the actual knowledge state of the learner. But there are differences to ABSYNT or IKEA due to the special demands of the Petri net domain: (1) specification of the tasks, (2) the analysis of the learner´s solution proposals (3) the generation of "episodic" rules on which help information in the form of completion proposal is based.

• *Specification of tasks.* For Petri net modeling task we use temporal logic specifications (Kröger, 1987). These enable the verification of learners´ Petri net proposals by model checking (Clarke et al., 1986).

• *Analyzing the learners´ solution proposals*. We developed a simple model checker (Clarke et al., 1986) for the

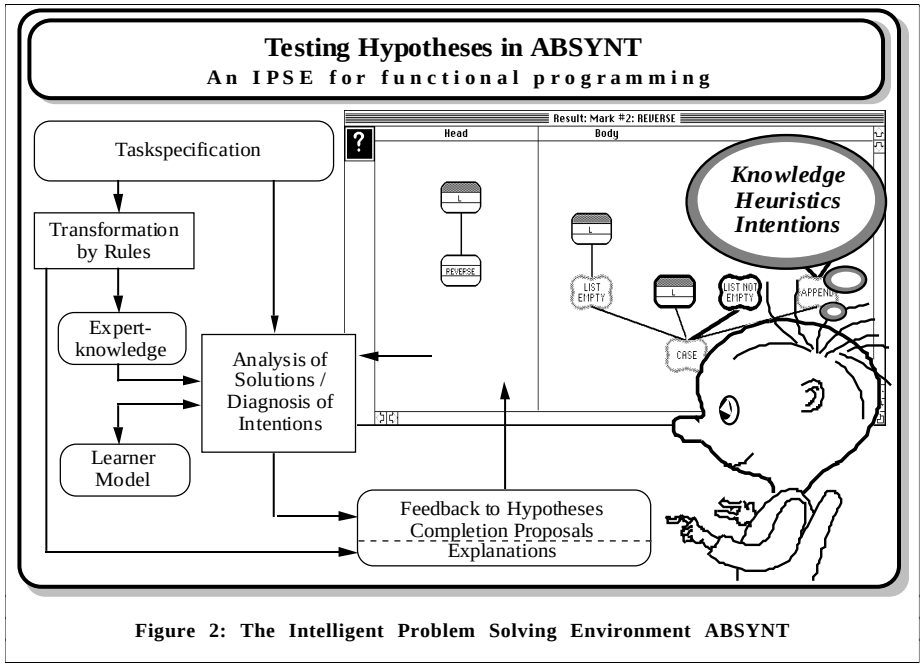

Figure 2: The Intelligent Problem Solving Environment ABSYNT

diagnosis of the user´s solutions in PETRI-HELP. The diagnosis is based on the case graph of the Petri net. In that graph, which describes all possible states of the system, the temporal-logic formulae of the specification are verified. Thus it is possible to detect the set of formulae which is fulfilled by a user-created net.

• *Testing hypotheses.* The student may state hypotheses about which temporal logic formulae s/he considers fulfilled by the current state of the solution proposal. The system analyzes the hypotheses and gives feedback accordingly. The model checker may be used after every editing step. If the formula contains not the temporal-logic operators O ("next time"), ◊ ("eventually"), [] ("always"), then it is a propositional-logic formula and will be evaluated inside the current node of the case graph. If the formula has the pattern O F (F is a formula), then F must hold in every immediate successor of the current state in the case-graph. ◊ F is true iff in every path leaving the current node F will be true at least in one node. Finally, [] F holds iff F holds in every state on every path leaving the current state.

• *Episodic rules and help information*. These rules will be learnt by the system when the model-checker finds that a net-fragement is a model of a specification subset. Completion proposals will be created *by the system* on the basis of the learnt episodic rules.

• Explanations: Why and Whynot-explanations can be given with the case graph. This is similar to the trace in ABSYNT.

IKEA

IKEA was developed, because a classical CBT program presenting configuration rules and multiple choice configuration tasks caused motivational problems. Parts of the configuration knowledge belongs to commonsense knowledge so that their presentation or replication is rather dull for the student. So we were asked to develop an IPSE. The students' task is to configure a room for a handicapped person. The system can test hypotheses (like: "Is my proposal embeddable into a correct solution?"), offer completion proposals and Why-/Whynot-explanations. Explanations show fullfilled and violated rules. The system has even a clearvoyance ability. It warns when the proposed configuration will run into problems.

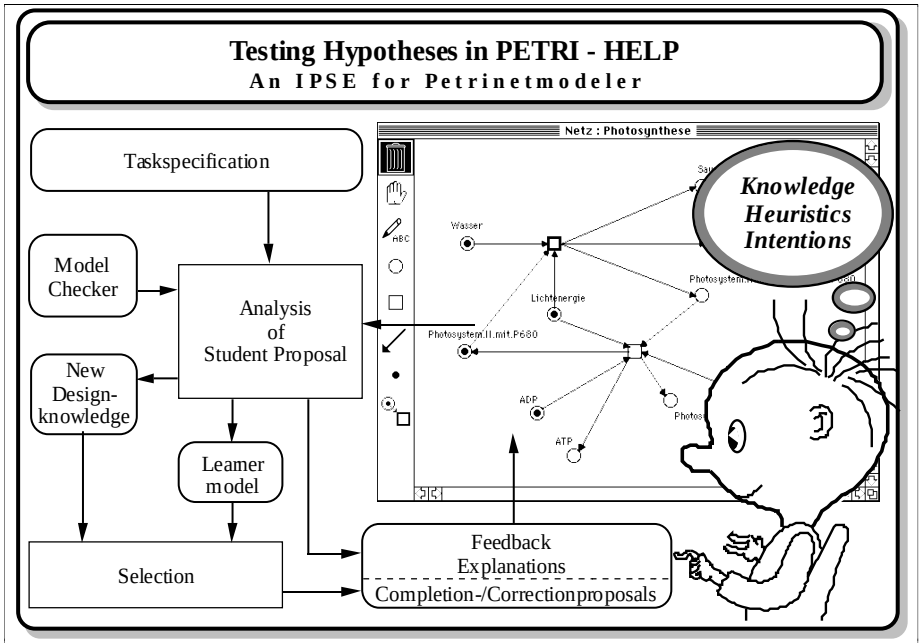

Figure 3: The Intelligent Problem Solving Environment PETRI-HELP

Theory Revision, Hypotheses, Knowledge Acquisition and Selfexplanations

As we stated before the formulation and testing of hypotheses is an important concept in the development of IPSEs. Though we may have an intuitive idea what a hypothesis is we try to give a formal definition. The definition is embedded in the concept of theory revision (De Raedt, 1992). We try to be as abstract as possible so that hypothesis testing in various IPSEs can be subsumed as special cases. The main points are summarized in Figure 5.

According to ISP-DL theory there are several steps when acquiring knowledge with IPSEs. (1) The problem solver generates with his subjective theory S evidence E, which may a solution proposal. From the viewpoint of an ideal expert this proposal may be wrong. (2) This proposal E is submitted to the IPSE. If the proposal is in error it cannot be explained by the domain theory T. So the problem solver can generate a hypothesis and partition his proposal E into two parts Efix and Emod. The student has the hypothesis that Efix can be embedded into a correct solution. According to this partition there is a corresponding partition of the domain theory but this is not under control of the student. (3) Now, the IPSE generates with a revised theory T' a system response to the hypothesis. E' is a system generated solution proposal, which contains Efix. E'mod is help information for the student which in our IPSEs is shown to the student on demand. (4) After these events (hopefully) we have some knowledge acquisition events. The student tries to explain E' with its parts Efix and E'mod to himself. According to (4) this is an inductive inference and when we compare (1) with (4) it is at the same time theory revision from S to S'.

Summary

We tried to show how a cognitive theory of knowledge acquisition (ISP-DL) motivates the IPSE concept and how the hypothesis testing capability can be described on a metalevel and implemented in various domains. Similar system for hydraulics, economic simulation games and causal modelling are under construction.

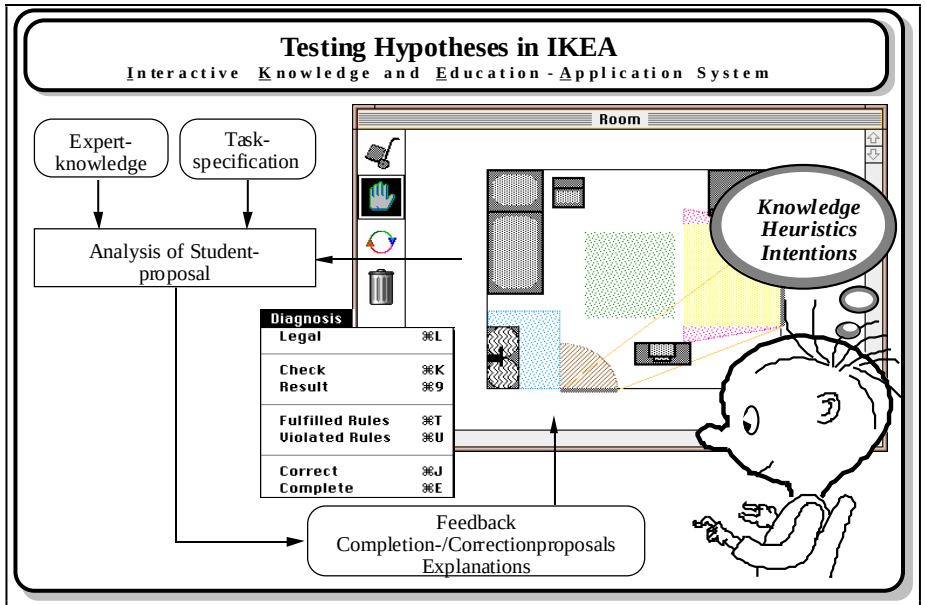

Figure 4: The Intelligent Problem Solving Environment IKEA

(1) Problem Solving: $S \models E$

(2) Testing of Hypothesis: $T = T_{fix} \cup T_{mod} \not\models E$

where: $E = E_{fix} \cup E_{mod}$

and: $T_{fix} \models E_{fix}$

(3) Revision of Theory: $T' = T_{fix} \cup T'_{mod} \models E'$

where: $E' = E_{fix} \cup E'_{mod}$

(4) Selfexplanation: $S' \models E'$

Figure 5 : Problem Solving, Hypothesis Testing and Selfexplanation

References

Anderson, J.R. (1986). Knowledge Compilation: The General Learning Mechanism. In: R.S. Michalski, J.G. Carbonell, T.M. Mitchell, Machine Learning II. Kaufman, 289-310

Anderson, J.R. (1989). A Theory of the Origins of Human Knowledge, Artificial Intelligence, 40, 313-351

Chi, M.T., De Leeuw, N., Chiu, M.-H. & LaVancher, Chr. (1994). Eliciting Self-Explanations Improves Understanding, Cognitive Science, 18, 439-477

Clarke, E.M., Emerson, F.A. & Sistla, A.P. (1986). Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specifications. ACM Transactions on Programming Languages and Systems, Vol. 8, No. 2, 244-263

De Raedt, L. (1992). Interactive Theory Revision, San Diego: Academic Press

Gollwitzer, P.M. (1990). Action Phases and Mind-Sets, in: E.T. Higgins & R.M. Sorrentino (eds), Handbook of Motivation and Cognition: Foundations of Social Behavior, Vol. 2, 53-92

Heckhausen, H. (1987). Perspektiven einer Psychologie des Wollens, 121 - 142, in: H. Heckhausen, P.M. Gollwitzer & F.E.Weinert (eds), Jenseits des Rubikon: Der Wille in den Humanwissenschaften, Berlin: Springer Verlag

Heckhausen, H. (1989). Motivation und Handeln, Heidelberg: Springer, (second ed.)

Huber, P., Jensen, K. & Shapiro, R.M. (1990). Hierarchies in Coloured Petri Nets, in: G. Rozenberg (ed), Advances in Petri Nets, Lecture Notes in Computer Science, Berlin: Springer

Kröger, F. (1987). Temporal Logic of Programs, Berlin: Springer

Laird,J.E, Rosenbloom, P.S. & Newell, A. (1987). SOAR: An Architecture for General Intelligence, Artificial Intelligence, 33, 1-64

Möbus, C., Schröder, O. (1993). The Acquisition of Functional Planning- and Programming Knowledge: Diagnosis, Modelling, and User-Adapted Help, in G. Strube, K.F. Wender (eds), The Cognitive Psychology of Knowledge. The German Wissenspsychologie Project (Advances in Psychology Series), Amsterdam: Elsevier (North-Holland), 233-261

Möbus, C., Schröder, O. & Thole, H.J. (1994). Diagnosing and Evaluating the Acquisition Process of Programming Schemata, in J.E. Greer, G. McCalla (eds), Student Modelling: The Key to Individualized Knowledge-Based Instruction (Proceedings of the NATO Advanced Research Workshop on Student Modelling, in St.Adele, Quebec, Canada), Berlin: Springer (NATO ASI Series F: Computer and Systems Sciences, Vol. 125), 211-264

Möbus, C., Schröder, O. & Thole, H.J. (1995). Online Modeling the Novice-Expert Shift in programming Skills on a Rule-Schema-Case Partial Order, in: K.F. Wender, F. Schmalhofer & H.D. Böcker (eds), Cognition and Computer Programming, Norwood N.J.: Ablex publishing Corporation, 63-105

Möbus, C., Thole, H.-J. & Schröder, O. (1993a). Interactive Support of Planning in a Functional, Visual Programming Language, in P. Brna, S. Ohlsson, H. Pain (eds), Proceedings AI-ED 93, World Conference on Artificial Intelligence and Education, Edinburgh, 362 - 369

Möbus, C., Thole, H.-J. & Schröder, O. (1993b). Diagnosis of Intentions and Interactive Support of Planning in a Functional, Visual Programming Language, in D.M. Towne, T. de Jong, H. Spada (eds), Simulation-Based Experiential Learning, Berlin: Springer (NATO ASI Series F: Computer and Systems Sciences, Vol. 122), 61 - 76

Newell, A. (1990). Unified Theories of Cognition, Cambridge, Mass.: Harvard University Press

Pitschke, K. (1994). User modeling for domains without explicit design theories, in: Proceedings of the Fourth International Conference on User Modeling (UM '94), Hyannis, MA.: The Mitre Corporation, 191-195

Schröder, O., Möbus, C. & Pitschke, K. (1995). A Cognitive model of Design Processes for Modelling Distributed Systems, paper presented on AI-ED '95 (this volume)

Self, J.A. (1990). Bypassing the Intractable Problem of Student Modelling, in C. Frasson, G. Gauthier (eds), Intelligent Tutoring Systems, Norwood: Ablex, 107-123

VanLehn, K. (1988). Toward a Theory of Impasse-Driven Learning, in H. Mandl, A. Lesgold (eds), Learning Issues for Intelligent Tutoring Systems, Berlin: Springer, 19-41

Zisman, M.D.(1978). Use of Production Systems for Modeling Asynchronous Concurrent Processes, in: Waterman D.H. (et al.), Pattern Directed Inference Systems, New York: Academic Press, 53-68

Acknowledgements: I want to thank the following researchers from our unit on "Learning and Teaching Systems" without their contribution this paper would have been only a dream: "Luftschloß" ("castle made of air"): Jörg Folckers, Knut Pitschke, Olaf Schröder, Heinz-Jürgen Thole.