Interactive Knowledge Acquisition Tools: A Tutoring Perspective

Yolanda Gil (gil@isi.edu) and Jihie Kim (jihie@isi.edu)
Information Sciences Institute
University of Southern California
4676 Admiralty Way
Marina del Rey, CA 90292, USA

Abstract

This paper argues that interactive knowledge acquisition tools would benefit from a tighter and more thorough incorporation of tutoring and learning principles. Current systems learn from users in a very passive and disengaged manner, and could be designed to incorporate the proactive capabilities that one expects of a good student. This paper points out what tutoring and learning principles have been used to date in the acquisition literature, though unintentionally and implicitly. We discuss how a more thorough and explicit representation of these principles would help improve enormously how computers learn from users.

Introduction

Computers have long been considered an invaluable tool for education. Intelligent tutoring systems and other kinds of educational software show how people can acquire knowledge about diverse topics by interacting with a computer. Given our reliance in computers for the future of education, we need to realize that an important component of the human education revolution is the computer education revolution: anyone should be able to teach computers on topics that are of value, so that anyone can learn about those topics from computers. So an important question is: how will computers acquire knowledge? In most cases knowledge is entered by hand by software or knowledge engineers, as is often done in intelligent tutoring systems (Forbus & Feltovich, 2001). This limits severely the utility of the tools, as it would be more desirable that the people with expertise in the domain at hand would be the knowledge providers. Knowledge can also be extracted from text (Cowie & Lehnert, 1996), although given the error rates of state of the art systems and the kinds of knowledge they acquire (mostly instance-level information) it will take many years for these techniques to be of practical use to build an accurate body of knowledge about a topic domain. Another possibility is to use interactive knowledge acquisition tools that help users enter knowledge. In recent years these systems have shown that end users with no background in computer science or knowledge representation were able to enter sizeable amounts of knowledge (Kim & Gil, 1999; Eriksson et al, 1995; Clark et al, 2001).

Although interactive knowledge acquisition tools enable end users to enter knowledge, users remain largely responsible for the acquisition process, both the teaching side and the learning side. These tools are quite passive in terms of formulating or pursuing learning goals, keeping track of the flow of a lesson, and generally assess how much they are learning and how useful that knowledge is. At the same time, users are not necessarily skilled teachers by nature, so being in a position to teach a computer is already a challenge for them. Interactive acquisition tools need to be more effective and helpful to users, perhaps by incorporating some of the skills that are expected of good students. And, as good students do, they should also be able to cope with an inexperienced teacher (which their users are likely to be) and still learn from the experience by bringing to bear knowledge about how a good teacher typically goes about a lesson. An interactive acquisition tool could then be viewed as a tool to support augmented cognition, since it would supplement the user's limitations as a teacher and knowledge engineer.

The contributions of this work are twofold. First, we point out how existing knowledge acquisition tools use techniques that are related to widely used tutoring and learning principles. Second, we identify areas that the acquisition tools developed to date have neglected, and suggest promising areas of research based on our findings. This would result in a new generation of acquisition tools that are not only better students but also more helpful to the teacher (the user).

The paper begins with a short introduction and background on interactive knowledge acquisition tools. We then discuss several tutoring and learning principles that we have drawn from the educational literature and that seem useful to support the interactive acquisition process. Next, we show how some existing acquisition tools use techniques that are related to these principles in some aspects of their functionality. We finalize with a discussion of promising directions that we see in designing acquisition tools that incorporate tutoring and learning principles more thoroughly.

Acquisition Tool	Highlights
CHIMAERA (McGuinness et al., 2000)	To acquire concepts, relations, and instances. Diagnoses faulty definitions.
EXPECT (Blythe et al., 2001)	To acquire problem solving knowledge. Exploits dialogue scripts, knowledge interdependency models, and background knowledge.
INSTRUCTO-SOAR (Huffman & Laird 1995)	To acquire task models for Soar.
KSSn (Gaines & Shaw, 1993)	To acquire concepts, rules, and data. Based on personal construct psychology.
PROTOS (Porter et al., 1990)	Users specify cases, tool explains their classification.
SALT (Marcus, 1988)	To acquire constraints and fixes for its underlying engine for configuration design.
SEEK2 (Ginsberg, 1985)	To acquire rules. Uses verification and validation techniques.
SHAKEN (Clark et al., 2001)	To acquire process models. Loosely based on concept maps.
TAQL (Yost, 1993)	To acquire SOAR rules. Based on Problem Space Computational Model.
TEIREISIAS (Davis, 1979)	To acquire rules. Exploits context, derived rule models, and heuristics.

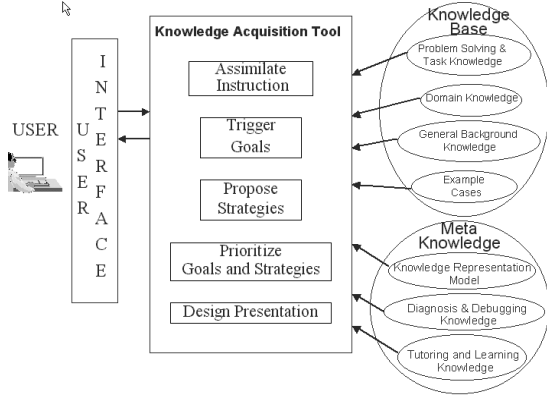Table 1: Some Interactive Knowledge Acquisition Tools.

Figure 1: A Modular View of Interactive Knowledge Acquisition Tools.

Interactive Knowledge Acquisition Tools

The interactive knowledge acquisition tools summarized in Table 1 illustrate different approaches that researchers have undertaken over the years and are representative of the literature. The tools are briefly described in Appendix A. The techniques used range from cognitive theories of expertise and learning, case-based reasoning and analogy, non-monotonic theory revision, induction and machine learning, knowledge engineering approaches, analysis of knowledge interdependencies and buggy knowledge, and agent-based interaction.

Figure 1 shows a diagrammatic view of typical components used in various tools. The functionality of an interactive knowledge acquisition tool can be described along five dimensions, which we will use for reference later in our analysis:

- **Assimilate instruction:** Given a user's instruction, the system makes the necessary additions or changes to the knowledge base and updates any other internal structures. Instruction may be given as an example (PROTOS, SEEK2), a natural language statement (INSTRUCTO-SOAR), a descriptive piece of knowledge (CHIMAERA, EXPECT, SALT, TAQL), or a graphical rendering (KSSn, SHAKEN).

- **Trigger goals:** The system analyzes its knowl-

edge and generates learning goals of what knowledge it still needs to acquire. Many tools focus on detecting inconsistencies or gaps in the knowledge base, which generate the goals to fix them or seek the information missing (CHIMAERA, EXPECT, SEEK2, TAQL, TEIREISIAS).

- **Propose strategies:** The tool can generate possible strategies that the user could follow in achieving the learning goals. It can also generate predictions of what strategy the user is more likely to pursue, or what answers the user is more likely to give to the user's questions (TEIREISIAS). This is often done by analyzing existing knowledge. Planning strategies are often used to make suggestions to the user in terms of what to do to achieve the active learning goals, which will make the acquisition process more efficient (EXPECT).

- **Prioritize goals and strategies:** An acquisition tool can help users further if it is able to organize and prioritize the active learning goals and candidate strategies, so that it can make more focused suggestions to the user. Sometimes these are organized by the type of knowledge sought (SALT), or by the type of goal being pursued or error being fixed (EXPECT).

- **Design presentation:** The tool can make decisions about what to bring to the attention of the user at each point in time to help him or her decide what to do next. There are many possibilities, and the tool can take into account the user's situation (user modeling), the stage of the process (initial stage versus final testing), and the content of the current knowledge base. The tool may present the user with a single question (INSTRUCTO-SOAR) or give the user a choice in the form of an agenda containing multiple items (CHIMAERA, EXPECT, SALT). The tool can suggest a specific strategy, anticipate the user's answer and ask for confirmation, or simply present the user with multiple possible strategies and suggestions (EXPECT). Other tools leave it up to the user to figure out what to do and simply make all possible options available to them (KSSn, SHAKEN) The tool may simply ask the user to

review an explanation (PROTOS), check some aspect of the knowledge (KSSn), or confirm a hypothesis (TEIREISIAS).

Another useful view of interactive acquisition tools is in terms of the kinds of knowledge and meta-knowledge that they bring to bear in order to support the user, also illustrated in Figure1. This includes:

- *General problem solving and task knowledge*: General inference structures are used to determine the role that domain-specific knowledge plays in problem solving, as is done in role-limiting approaches to knowledge acquisition (Marcus & McDermott, 1989) (e.g., SALT, TAQL).

- *Prior domain knowledge*: The initial knowledge base may contain terms that are specific to the domain at hand and that can be used to define new terms and tasks (e.g., EXPECT, INSTRUCTO-SOAR).

- *General background knowledge*: The initial knowledge base may include high level theories and ontologies that capture general knowledge, such as time, physical objects, etc. (e.g., SHAKEN).

- *Example cases*: Sample situations, test cases, and problem solving episodes can help ground abstract knowledge (e.g., INSTRUCTO-SOAR, PROTOS, SEEK2, SHAKEN, TEIREISIAS).

- *Underlying knowledge representation*: Models of the underlying knowledge representation will determine how users need to formulate new knowledge (e.g., CHIMAERA, KSSn, SEEK2, TAQL, TEIREISIAS).

- *Diagnosis and debugging knowledge*: Typical diagnosis skills are useful in order to detect errors and potential problems in the knowledge base. Effective debugging strategies can be incorporated to make suggestions to the user about how to fix the errors and problems found (e.g., CHIMAERA, EXPECT, TEIREISIAS).

One source of meta-knowledge that has not received attention is effective tutoring and learning techniques. By exploiting meta-knowledge about how to learn and how to teach, acquisition tools will become more proactive learners and will be able to help users teach them more effectively. Current tools are often too passive, and place on the user the majority of the burden of the acquisition process. Our goal is to understand whether and how knowledge acquisition tools can exploit knowledge about tutoring and learning.

Tutoring and Learning Principles in Existing Interactive Acquisition Tools

We analyzed the tutoring and educational literature to compile tutoring and learning principles that humans and computers exploit to make teaching and learning more effective. We compiled fifteen principles that could be of immediate use in our work, and that are described in detail in (Kim & Gil, 2002) including detailed references to the tutoring literature that are omitted in this paper because of space limitations.

We noticed that many of these principles could be related to the techniques used in existing acquisition tools. Yet, the tutoring literature is seldom mentioned in knowledge acquisition work. In this section, we describe our views on how acquisition techniques can be expressed in terms of these tutoring and learning principles. Table 2 summarizes our analysis, indicating the particular functionality (as outlined in Figure 1) where the principle was applied in specific acquisition tools.

Introduce lesson topics and goals

Teachers often start off by introducing the topics and goals of the lesson. There is no notion in acquisition tools that there is a lesson being started or ended, since at any point users can choose to enter knowledge about any topic. EXPECT allows users to specify the top-level tasks that the system should be able to solve with the new knowledge, which can be viewed as a statement of the goals for that acquisition session. SEEK2 has a suite of test cases that the system should be able to solve after the lesson, and that could be viewed as a statement of the goals of the lesson.

Use topics of the lesson as a guide

It is useful for students and tutors to ensure that what is being learned has some connection or relevance to the topics of the lesson. EXPECT uses the specified top-level tasks to check that any new knowledge specified solves some of their subtask, and if not it notifies the user and suggests how it could play a role in solving the tasks. SEEK2 uses the suite of test cases to detect errors, which then drive the dialogue with the user towards fixing them. SALT can be viewed as having an implicit (and very high level) topic for all sessions, namely to acquire knowledge for configuration design problems. SALT's interface asked users to specify only three kinds of knowledge (parameters, constraints, and fixes) that are relevant to those types of problems.

Subsumption to existing cognitive structure

Learning about a new topic involves relating the new knowledge to what is already known, for example by

Tutoring/Learning principle	Assimilate Instruction	Trigger Goals	Propose Strategies	Prioritize Goals and Strategies	Design Presentation
Introduce lesson topics and goals		EXPECT, SEEK2			
Use topics of the lesson as a guide	SALT	SEEK2	EXPECT		SALT
Subsumption to existing cognitive structure	PROTOS		TEIREISIAS		PROTOS, SALT
Immediate feedback	PROTOS	INSTRUCTO-SOAR	TEIREISIAS		EXPECT
Generate educated guesses		TEIREISIAS	EXPECT		
Keep on track					
Indicate lack of understanding	INSTRUCTO-SOAR				INSTRUCTO-SOAR
Detect and fix "buggy" knowledge	TAQL	EXPECT CHIMAERA			PROTOS, SEEK2 TEIREISIAS
Learn deep models					
Learn domain language					
Keep track of correct answers		SEEK2			
Prioritize learning tasks				EXPECT	
Summarize what was learned					
Assess learned knowledge		KSSn			

Table 2: Tutoring and learning principles used in acquisition tools.

checking inconsistencies, drawing analogies, or deriving generalizations. PROTOS took a new example case provided by the user, and indexes it into one of several classes (or categories) of examples. It also presented the user with an explanation of the classification of the new example to show how the new knowledge was incorporated into the existing structures. TEIREISIAS created generalized rule models from its rule base, and used them to propose to the user additional conditions to newly defined rules. The interface and presentation of SALT was always based on the kinds of knowledge needed for configuration design.

Immediate feedback

Educational systems often provide immediate feedback, as studies show that it is more effective than feedback received after a delay. PROTOS provided immediate feedback as a new case was assimilated by showing the user an explanation of its classification in the knowledge base. INSTRUCTO-SOAR generated clarification and follow-up questions for the user immediately after an instruction was given. TEIREISIAS proposed amendments to rules as soon as the user defined them. EXPECT analyzes the knowledge base after each user action and shows immediately an agenda of errors to resolve and tasks to do.

Generate educated guesses

Students often show their understanding by finishing a tutor's utterance, and tutors often invite students to guess as a way to assess and correct the student's knowledge. TEIREISIAS maps newly entered rules to rule models and proposes corrections based on how it expects a rule to follow the patterns of other rules in that model. EXPECT generates suggestions to a user about how to fix specific problems by making educated guesses about the context of the problem (related domain knowledge, past problem solving states, etc.)

Keep on track

Tutors need to keep track of the lesson and bring back issues that had to be dropped while engaging in clarifications or other side dialogues. Acquisition tools do not keep track of the history and status of the dialogue. Users have free range on what aspects of the knowledge base to extend, what parts of the tool to invoke, and what They can move freely from topic to topic and back and forth, or discontinue teaching about a topic at any point without notifying termination. Current acquisition tools would never even notice that the user is deviating from a topic in any of these situations.

Indicate lack of understanding

Students often volunteer an indication of their lack of understanding, but tutors also will point out the specific aspects introduced in a lesson that the student needs to understand. INSTRUCTO-SOAR detects missing aspects of a task description specified by a user and generates follow up questions. EXPECT and CHIMAERA detect undefined terms that will be used to guide future dialogue with the user to define them.

Detect and fix "buggy" knowledge

Many tutoring systems are aimed to diagnose and fix student's "buggy" knowledge, often by asking questions and checking the student's answers. TAQL analyzes the knowledge specified by the user and points out errors based on static analysis. CHIMAERA and EXPECT detect errors in the knowledge entered that need to be fixed by the user. PROTOS, SEEK2, and TEIREISIAS show explanations or traces to users so they can detect errors in the system's reasoning.

Learn deep models

Students should learn deep conceptual models instead of superficial ones. Knowledge acquisition tools do not have any basis to evaluate or pursue depth in their knowledge base, though this is a long

recognized shortcoming of knowledge-based systems. To date, these systems are at the mercy of the user's intention and of their implementation of any depth in the models.

Learn domain language

Students are expected to be describe their knowledge in terms that are suitable for the domain at hand. Acquisition tools do not help users specify how to describe knowledge in domain terms and how the terminology used depends on the context of the scenario at hand. Knowledge bases are annotated with some lexical information, but acquiring this kind of knowledge has not been a focus of knowledge base development.

Keep track of correct answers

Instructional tools keep track of the questions that the student is able to answer correctly as well as those answered incorrectly, which drives further interactions with the student. SEEK2 keeps track of whether the test cases are answered correctly, and alerts the user when a change to a rule causes a case to be solved incorrectly.

Prioritize learning tasks

Tutoring systems often handle multiple sub-tasks using priority rules that look at the duration and type of task, for example focusing on fixing errors before turning to omissions. EXPECT organizes errors and other problems in the knowledge base based on their type and the amount of help it can provide (e.g., if it has narrowed down the options that the user can take to resolve them).

Limit the nesting of sub-lessons

Tutoring dialogue is sometimes controlled by limiting the amount of subdialogues, which helps the student keep track of the lesson topics.

Summarize what was learned

Many educational systems will summarize to the student the main highlights at the end of the lesson, especially if the student was given hints during the lesson. Acquisition tools do not summarize what they have learned.

Assess learned knowledge

Some instructional tools isolate weaknesses in the student's knowledge and propose further lessons on those areas, some students also volunteer their assessment of how well they understand certain topics. KSSn uses clustering techniques to suggest aspects of the model that users could detail further. Other acquisition tools do not perform this kind of analysis. Users often have to put the knowledge base through a performance system that exercises it in order to be able to assess if the knowledge was learned appropriately.

Discussion

Acquisition tools have used techniques that can be cast in terms of tutoring and learning principles found in educational software research. These principles are implicit in the design of the tool, and they influence their interaction with the user to the degree that they are implemented in the underlying code. Having these principles represented explicitly and declaratively would enable acquisition tools to reason in terms of the teaching and learning process, and their interaction with the user would be dynamically generated given the situation at hand. A declarative representation of meta-knowledge about their learning state, goals, and possible strategies could turn interactive acquisition tools into more proficient and proactive learners.

The principles have only been used in some aspects of the functionality of acquisition tools, and are exhibited by some but not all the tools. The sparseness of the matrix in Table 2 points to many opportunities for future work in incorporating these principles. By having declarative representations of their learning state, goals, and possible strategies, interactive acquisition tools could more easily incorporate these principles throughout the acquisition process and the five functions shown in the table.

Acquisition interfaces should be able to structure the dialogue with the user in tutoring terms. The should organize the dialogue based on lesson topics and sub-topics, be aware of the start and the end of each and generally keep the user on track and delaying termination until the goals of the lesson are satisfied. Acquisition tools should exploit the topics of the lesson throughout the acquisition process, for example to narrow down the prior knowledge that is relevant to that portion of the dialogue and consequently narrowing down the proposed strategies and customizing the presentation of information back to the user. By keeping track of the interactions with the user, the topic of the dialogue at each point in time, and the termination of sub-topics, acquisition tools would be able to manage their participation in the dialogue better and relieve the users from having to remember and keep track of what is going on. They could exploit this information in generating goals by detecting areas where a topic is still unfinished, plan and prioritize more relevant strategies that exploit the context of the currently open topics, and help users view progress and termination.

Acquisition tools should be able to expose and assess the knowledge acquired so far, allowing the user to understand what the system has assimilated and showing the user as well what areas the system thinks need to be further improved. Currently, knowledge-based systems will answer any question they are asked, regardless of the quality of the knowledge used to answer it. It would be very useful for these systems to convey whether they are confident on the answer. This would also help users identify

further areas of improvement for future acquisition sessions.

We are pursuing these ideas in our current work, implementing a front-end dialogue management system that represents and uses tutoring and learning principles to guide knowledge acquisition.

Acknowledgments

Bareiss, R., & Porter, B., & Holte, R. (1990). Concept learning and heuristic classification in weak-theory domains. *Artificial Intelligence Journal* 45(1-2):229–264.

Blythe, J., & Kim, J., & Ramachandran, S., & and Gil, Y. (2001). An integrated environment for knowledge acquisition. *Proceedings of the Intelligent User Interfaces conference (IUI-2001)*.

Clark, P., & Thompson, J., & Barker, K., & Porter, B., & Chaudhri, V., & Rodriguez, A., & Thomere, J., & Mishra, S., & Gil, Y., & Hayes, P., & Reichherzer, T. (2001). Knowledge entry as the graphical assembly of components. *Proceedings of First International Conference on Knowledge Capture (K-CAP-2001)*.

Cowie, J., & Lehnert, W. (1996). Information extraction. *Communications of the ACM* 39(1):80–91.

Davis, R. (1979). Interactive transfer of expertise: Acquisition of new inference rules. *Artificial Intelligence* 12:121–157.

Eriksson, H., &shahar, Y., & Tu, S. W., & Puerta, A. R., & Musen, M. (1995). Task modeling with reusable problem-solving methods. *Artificial Intelligence* 79:293–326.

Forbus, K., & Feltovich, P., eds. (2001). *Smart Machines in Education*. AAAI press.

Gaines, B. R., & Shaw, M. (1993). Knowledge acquisition tools based on personal construct psychology. *The Knowledge Engineering Review* 8(1):49–85.

Ginsberg, A., & Weiss, S., & Politakis, P. (1985). SEEK2: A generalized approach to automatic knowledge base refinement. *Proceedings of IJCAI-85*.

Huffman, S. B., & Laird, J. E. (1995). Flexibly instructable agents. *Journal of Artificial Intelligence Research* 3:271–324.

Kim, J., and Gil, Y. (1999). Deriving expectations to guide knowledge base creation. *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, 235–241.

Kim, J., and Gil, Y. (2002). Deriving acquisition principles from tutoring principles. *Proceedings of the Intelligent Tutoring Systems Conference (ITS-2002)*, Biarritz, France, June 2002.

Marcus, S., and McDermott, J. (1989). SALT: A knowledge acquisition language for propose-and-revise systems. *Artificial Intelligence* 39(1):1–37.

McGuinness, D. L.; Fikes, R.; Rice, J.; and Wilde, S. 2000. An environment for merging and testing large ontologies. *Proceedings of KR-2000*.

Newell, A., & Yost, G., & Laird, J., & Rosenbloom, P., & Altmann, E. (1991). Formulating the problem-space computational model. Rashid, R. F., ed., *CMU Computer Science: A 25th Anniversary Commemorative*. ACM Press.

Novak, J., ed. 1998. *Learning, Creating, and Using Knowledge: Concept Maps as Facilitative Tools in Schools and Corporations*. Lawrence Erlbaum.

Yost, G. R. 1993. Knowledge acquisition in Soar. *IEEE Expert* 8(3):26–34.

Appendix A: A Brief Overview of Interactive Knowledge Acquisition Tools

Below is a brief summary of ten tools that are representative of the techniques used in interactive knowledge acquisition.

CHIMAERA — CHIMAERA (McGuinness et al., 2000) is an ontology editing environment that helps knowledge engineers develop, browse, and merge ontologies. As users define, extend, or merge classes (concepts), it runs a suite of diagnosis tools that detect different kinds of errors and undesirable features in the knowledge base. An example of an error is a logical inconsistency. An undesirable feature is, for example, a class with a large number of subclasses, where the user should think about defining some intermediate subclasses to provide better structure to the knowledge base. For each diagnosis, it can suggest users possible strategies to follow that will fix that problem.

EXPECT — EXPECT (Blythe et al, 2001) is a tool to acquire problem solving knowledge from end users. It uses a variety of techniques, including dialogue planning through scripts and wizards to help users make complex extensions to the knowledge base, deriving models of knowledge interdependencies to notify users of possible inconsistencies and mismatches, and exploiting background knowledge to create strong expectations of how the new knowledge fits. EXPECT guides users by providing structured editors with context-sensitive choices when users specify a new piece of problem-solving knowledge, by generating wizard-like dialogues based on

the knowledge it expects users to provide, and by showing an agenda of errors in the knowledge base together with suggested fixes.

INSTRUCTO-SOAR — INSTRUCTO-SOAR (Huffman & Laird, 1995) is a tool to help end users specify task knowledge. It uses a combination of machine learning, natural language, agent-based, and instructional technologies. The user provides situated instruction as natural language sentences and the system, a Soar agent, assimilates them into a Problem Space Computational Model (Newell et al, 1991). INSTRUCTO-SOAR processes the natural language sentence using its current task knowledge, formulates operators that model the task, and generalizes operators by deriving their weakest preconditions. It uses explanations and inductive techniques to refine its task model. Users can specify hypothetical situations, contingency options, and test the agent's knowledge at any point by asking it to perform a task.

KSSn — KSSn (Gaines & Shaw, 1993) is a family of knowledge editors based on personal construct psychology that enables end users to specify conceptual structures. Through a graphical editor, users can specify a graph of concepts, roles, constraints, and rules that the system then translates into a formal representation. Users can also be guided to construct a repertory grid that the system can translate into a graphical form. Clustering techniques are used to detect aspects of the model where users should add further structure.

PROTOS — PROTOS (Bareiss et al, 1990) helped users specify knowledge for classification tasks through examples. A user provided training cases, specified the features of a new case, the system classified the case by matching its features with a set of categories, and showed the user an explanation for the classification. If the user disagreed with the explanation then the system asked for salient differences between the case at hand and the matched category, which will be used to correct the matches.

SALT — SALT (Marcus & McDermott, 1989) helped end users specify domain-specific knowledge for configuration design tasks. SALT is one of a family of tools that use what is known as a *role-limiting approach* to knowledge acquisition (Marcus & McDermott, 1989). Essentially, a tool is built for each kind of generic problem solver or inference structure (e.g., classification, configuration design, skeletal planning), and it elicits from users the domain-specific knowledge that plays a specific role during problem solving. SALT was built for configuration design, where an initial configuration is proposed by assigning values to the design parameters, the configuration is checked against required constraints, and for each constraint violated it applies possible fixes that result in a new proposed configuration that results in a new iteration. SALT allows users to specify domain knowledge only as parameters, constraints,

or fixes.

SEEK2 — SEEK2 (Ginsberg et al, 1985) uses validation and verification techniques to check a suite of rules specified by a knowledge engineer. It help the user check that the rule set solves correctly a suite of test cases.

SHAKEN — SHAKEN (Clark et al., 2001) allows end users to specify process models. It uses a graphical editor inspired by concept maps (Novak, 1998), and extended so that process steps and objects are modeled after an existing class in the background knowledge base. Users can test the knowledge entered by instantiating a question template, or by requesting an error report from running a simulation of the process.

TAQL — TAQL (Yost, 1993) is a tool to help knowledge engineers develop knowledge for Soar's Problem Space Computational Model (PSCM) (Newell et al, 1991). Users specify Soar operators using the TAQL programming language using a text editor, which are higher-level descriptions of the task that are then translated by TAQL into the PSCM framework. TAQL performs a static analysis of the operators and detects typical errors that users make when modeling knowledge in the PSCM framework.

TEIREISIAS — TEIREISIAS (Davis, 1979) was developed as an acquisition tool for MYCIN. Users specified new rules in the context of an example problem, and MYCIN used that context and the explanations of the trace to help the user debug the new knowledge. MYCIN derived rule models by generalizing among similar rules, and alerted users when a new rule deviated from the model.