# Incorporating Tutoring Principles into Interactive Knowledge Acquisition

Jihie Kim and Yolanda Gil

Information Sciences Institute
University of Southern California
4676 Admiralty Way, Marina del Rey, CA 90292, USA
jihie@isi.edu, gil@isi.edu

**Abstract**

This paper argues that interactive knowledge acquisition tools would benefit from a tighter and more thorough incorporation of tutoring and learning principles. Current acquisition 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. We first describe our analysis of the literature on tutorial dialogues and present a compilation of tutoring and learning principles that are relevant to interactive knowledge acquisition. We then point out what tutoring and learning principles have been used to date in the acquisition literature, though unintentionally and implicitly and discuss how a more thorough and explicit representation of these principles would help improve enormously how computers learn from users. Finally, we present our design and an initial implementation of a new acquisition dialogue tool called *SLICK* that represents acquisition principles and goals explicitly and declaratively, making the system actively reason about various acquisition tasks and generate its presentations dynamically. The resulting interactions show that the system is aware of its progress towards acquiring the new knowledge, and actively participate in learning by understanding what acquisition goals and strategies to pursue.

Some material in this paper is also reported in the proceedings of the Intelligent Tutoring Systems Conference (ITS 2002) or in the proceedings of the 24th Annual Meeting of the Cognitive Science Society (CogSci-2002).

# 1 Introduction

Transferring knowledge from humans to computers has proven to be an extremely challenging task. Over the last two decades, an array of approaches to interactive knowledge acquisition have been proposed. Some tools accept rules and check them against other existing rules (Davis, 1979; Ginsberg *et al.*, 1985). Some tools acquire knowledge suitable for specific tasks and problem solving strategies (Marcus & McDermott, 1989). Other tools focus on detecting errors in the knowledge specified by the user (Gil & Melz, 1996; Kim & Gil, 1999; McGuinness *et al.*, 2000). Some systems use a variety of elicitation techniques to acquire descriptive knowledge (Gaines & Shaw, 1993; Novak, 1998) often in semi-formal forms. There are some isolated reports of users with no formal background in computer science that are now able to use acquisition tools to build sizeable knowledge bases (Kim & Gil, 2000; Eriksson *et al.*, 1995; Clark *et al.*, 2001). However, the majority of the burden of the acquisition task still remains with the user. Users have to decide what, how, and when to teach the system. Current acquisition tools do not take the kind of initiative and collaborative attitude that one would expect of a good student, mostly reacting to the user's actions instead of being proactive learners.

We set off to investigate how the dynamics of tutor-student interactions could be used to make acquisition tools better students to further support users in their role of tutors of computers. Given the success in deploying educational systems in schools and their reported effectiveness in raising student grades (Koedinger *et al.*, 1997), we expected the tutoring literature to have useful principles that we could exploit. Another strength of tutoring work is that it is typically motivated by extensive analysis of human tutorial dialogues (Fox, 1993), which the knowledge acquisition literature lacks.

The contributions of this work are threefold. First we present our analysis of the literature on tutorial dialogues and provide a compilation of useful principles that students and teachers follow in making tutoring interactions successful and that could be useful in the context of interactive acquisition tools. Second, we point out how existing knowledge acquisition tools use techniques that are related to widely used tutoring and learning principles. Based on these analyses we identify areas that the acquisition tools developed to date have neglected, and suggest promising areas of research based on our findings. Finally, we present our design and implementation of a new acquisition dialogue tool called *SLICK* (Skills for Learning to Interactively Capture Knowledge) that is built based on these observations.

SLICK makes acquisition tools more proactive by maintaining 1) goals that represent what remains to be learned, 2) strategies to achieve these goals and acquire further knowledge, and 3) awareness of the current status of the body of knowledge learned. The tool has been used for acquiring two

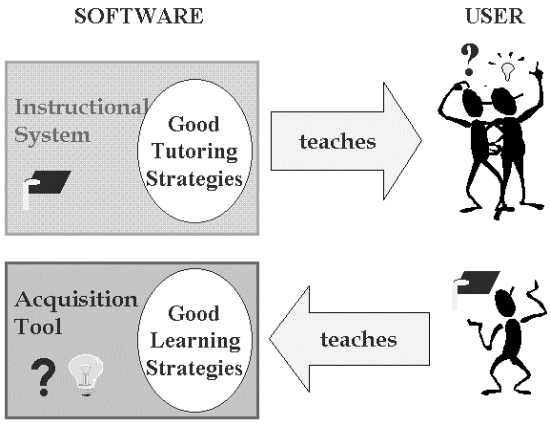



FIGURE 1: Comparing Tutoring Systems and Acquisition Systems

very different types of knowledge: biological process models and military plans. The resulting interactions show that the system is aware of its progress towards acquiring the new knowledge, and moves forward by understanding what acquisition goals and strategies to pursue.

The paper begins with a discussion of the similarities and differences between instructional systems (educational software and human tutoring) and interactive acquisition tools, illustrating how acquisition tools can exploit knowledge about tutoring and learning. The next section presents a summary of our analysis of tutoring principles, describing fifteen learning principles that we believe can be immediately incorporated into our current tools. We then show how some existing acquisition tools use techniques that are related to these principles in some aspects of their functionality. We also discuss promising directions that we see in designing acquisition tools that incorporate tutoring and learning principles more thoroughly. Finally, we present our design and an initial implementation of SLICK, and describe lessons learned from user feedback.

## 2 Comparing Tutoring Systems and Acquisition Systems

Figure 1 illustrates the parallels between an instructional system and an interactive acquisition interface. In instructional systems (both educational software and intelligent tutoring systems), the tutor's role is to help the user (student) achieve some degree of proficiency in a certain topic (the lesson). In interactive acquisition tools, these roles are reversed. Acquisition tools can be seen as students learning new knowledge from the user (teacher) and they should be able to use some of the strategies that good learners pursue during a tutoring dialogue. Ideally, it should also be able to supplement the user's skills as a teacher by helping the user pursue effective tutoring techniques.

This would help the user teach the material better and faster to the system, as well as delegate some of the tutor functions over to the system.

In essence, we are trying to investigate what it takes to create a good student, while most ITS work has focused on creating good teachers. We believe that the work in educational systems and acquisition systems share a lot of issues and they may be able to contribute to each other in many ways. In fact there has been work that bridges these two communities. For example, there have been recent interests in acquiring knowledge for intelligent tutoring systems (Murray, 1999). We think that technology built by the knowledge acquisition community will be useful for building tools to help users develop the knowledge and models used in ITS.

There are some issues that interactive acquisition tools will not face. Human students in need of tutoring often have a lack of motivation that the instructional system has to address (Lepper *et al.*, 1993). Instructional systems need to use special tactics to promote deep learning, such as giving incremental hints instead of showing the student the correct answers. Finally, our student will not be subject to the cognitive limitations of a typical human student, and can exploit memory and computational skills that would be exceptional (if not infrequent) for human students.

# 3 Teaching and Learning Principles Relevant Interactive Acquisition

We have been investigating various tutoring principles[1] used by human tutors and educational software (Forbus & Feltovich, 2001; Wenger, 1987; Fox, 1993). Although human tutors provide more flexible support, the tutoring principles supported by educational software are often inspired by human tutors (Merrill *et al.*, 1992) and we derive learning principles from both. Table 1 shows a summary of the principles that we found useful. The rest of this section describes these principles.

Instructional systems contain other components such as student models and domain models, but here we are focusing on tutoring principles and leave user modeling as future work.

1. **Introduce lesson topics and goals**

   In the beginning of the lesson, tutors often outline the topics to be learned during the session and try to assess the student's prior knowledge on these topics. For example, the advance organizer approach (Ausubel, 1968) lets the student see the big picture of what is to be learned and provides what the tutor's argument will be in order to bridge the gap between what the

[1] In the tutoring literature these are often referred to as tutoring strategies. We prefer to refer to them as tutoring principles, since we found that they can be implemented as goals, strategies, or plans during the dialogue, or simply be taken into account in the design of the interaction.

| Tutoring/Learning principle | References |
|---|---|
| P1: Introduce lesson topics and goals | Atlas-Andes, Meno-Tutor, Human tutorial dialog human learning |
| P2: Use topics of the lesson as a guide | BE&E, UMFE |
| P3: Subsumption to existing cognitive structure | human learning, WHY, Atlas-Andes |
| P4: Immediate feedback | SOPHIE, Auto-Tutor, LISP tutor Human tutorial dialog, human learning |
| P5: Generate educated guesses | Human tutorial dialog, QUADRATIC, PACT |
| P6: Keep on track | GUIDON, SCHOLAR, TRAIN-Tutor |
| P7: Indicate lack of understanding | Human tutorial dialog, WHY |
| P8: Detect and fix "buggy" knowledge | SCHOLAR, Meno-Tutor, WHY, Buggy, CIRCSIM human learning |
| P9: Learn deep models | PACT, Atlas-Andes |
| P10: Learn domain language | Atlas-Andes, Meno-Tutor |
| P11: Keep track of correct answers | Atlas-Andes |
| P12: Prioritize learning tasks | WHY |
| P13: Limit the nesting of the lesson to a handful | Atlas |
| P14: Summarize what was learned | EXCHECK, TRAIN-Tutor, Meno-Tutor |
| P15: Assess learned knowledge | WEST, Human tutorial dialog |

References: Atlas (VanLehn *et al.*, 2000), Atlas-Andes(Rose *et al.*, 2001), BE&E (Core *et al.*, 2000), Buggy (Brown & Burton, 1978), CIRCSIM-tutor (Zhou *et al.*, 1999), EXCHECK (McDonald, 1981), GUIDON (Clancey, 1987), Human tutorial dialog (Fox, 1993), human learning (Merrill *et al.*, 1992; Gentner *et al.*, 2001; Kulik & Kulik, 1988; Ausubel, 1968; Collins & Stevens, 1982; Festinger, 1957), LISP Tutor (Anderson *et al.*, 1989), Meno-Tutor (Woolf & McDonald, 1984), PACT (Aleven & Koedinger, 2000), QUADRATIC (O'Shea, 1979), SCHOLAR (Carbonell, 1970), SOPHIE (Brown *et al.*, 1982), TRAIN-Tutor (Woolf & Allen, 2000), UMFE (Sleeman, 1984), WEST (Burton & Brown, 1979), WHY (Stevens & Collins, 1977).

TABLE 1: Some Tutoring and Learning Principles relevant to interactive knowledge acquisition

student may already know and what the student should learn. In educational systems, such as Meno-Tutor (Woolf & McDonald, 1984), as the tutor introduces general topics it asks exploratory questions in order to assess the student's prior knowledge. In fact, there are similar findings in teacher-student dialogs. Teachers often let students express how good or bad they are at given topic (Fox, 1993).

2. **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. In planning tutorial dialogues, instructional systems check what is being learned against the topics of the lesson (Core *et al.*, 2000) and try to avoid unfocused dialogue and digressions. In the process of learning, the terms brought up during the lesson are connected to the concepts learned (Sleeman, 1984).

3. **Subsumption to existing cognitive structure**

   The subsumption theory by Ausubel (Ausubel, 1968) emphasizes that learning new material involves relating it to relevant ideas in the existing cognitive structure. The integration of new material with previous information can be done by analogies, generalizations and checking consistency. Through analogy, novel situations and problems can be understood in terms of

familiar ones (Gentner *et al.*, 2001). Effective human tutors ask for similarities and differences for similar cases (Collins & Stevens, 1982). In educational systems such as Atlas-Andes (Rose *et al.*, 2001), the system points out differences between similar objects (e.g., speed vs. velocity) in terms of what they are and how they are calculated. Human tutors help students generalize when there are several similar cases (Collins & Stevens, 1982). For example, they suggest or point out the need to formulate a rule for similar cases by asking how the values of certain factors are related to the values of the dependent variables. Educational systems, such as Atlas (VanLehn *et al.*, 2000), encourage students to abstract plans from the details to see the basic approach behind problem solving. Finally, cognitive dissonance theory (Festinger, 1957) points out that people tend to seek consistency among their cognitions (i.e., beliefs, opinions). When there is an inconsistency (dissonance), something must change to eliminate the dissonance.

4. **Immediate feedback**

   Many educational systems provide immediate feedback on the quality of student's responses (Brown *et al.*, 1982; Anderson *et al.*, 1989). The studies of feedback in a variety of instructional context find that immediate feedback is much more effective than feedback received after a delay (Kulik & Kulik, 1988). Similarly, in the tutorial dialog study by Fox (Fox, 1993), tutors show immediate recognition of every step the student makes and their silence tends to presage the student's confusion. It is reported that in providing feedback, human tutors are more flexible than educational software, using high bandwidth communication to guide the students (Merrill *et al.*, 1992).

5. **Generate educated guesses**

   Some educational systems invite guesses on questions either in the process of letting the student discover the answers (O'Shea, 1979) or in the process of assessing the student's knowledge(Aleven & Koedinger, 2000). Likewise, in the studies of human tutoring, student often display their understanding by finishing the tutor's utterance and the tutor finds out what students understood by inviting their guesses (utterance completion strategy) (Fox, 1993).

6. **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. If the student gives an incorrect answer, the tutor must immediately get the student back on track (Carbonell, 1970). Some educational systems detect change of directions (Woolf & Allen, 2000) or check if the questions are

irrelevant to the case at hand (Clancey, 1987).

7. **Indicate lack of understanding**

   Studies in human tutoring show cases where students themselves indicate lack of understanding of introduced terms (Fox, 1993), but tutors also point out the specific aspects introduced in a lesson that the student needs to understand.(Collins & Stevens, 1982).

8. **Detect and fix "buggy" knowledge**

   Many educational systems have a tutoring goal of diagnosing the student's "bugs" (Woolf & McDonald, 1984; Stevens & Collins, 1977; Brown & Burton, 1978) and question answering is often used in checking student's knowledge. However, simply telling that an error has occurred is much less useful than reminding the student of the current goal or pointing out a feature of the error (McKendree, 1990). If there are insufficient or unnecessary factors in a student's answers, experienced tutors pick counter examples to highlight the problem (Collins & Stevens, 1982). In the process of checking, when the tutor does not understand the answer, sometimes the student is asked to rephrase the answer (Carbonell, 1970).

9. **Learn deep models**

   The tutor and the student should focus on deep conceptual models and explanations rather than superficial ones (VanLehn *et al.*, 2000). Students should not only be expected to give the right answer but to do so for the right reasons. For example, when the student's answer is right, educational systems ask how the correct answer is generated (Aleven & Koedinger, 2000; VanLehn *et al.*, 2000). In some cases, to be able to ensure that the student understood the explanation educational systems use a set of check questions (Rose *et al.*, 2001). Studies of human tutoring show that students themselves occasionally try to check the reasoning behind the answers provided (Fox, 1993).

10. **Learn domain language**

    Another interesting aspect of a lesson is learning to describe the new knowledge in terms that are appropriate in the domain at hand. Educators want to ensure that the students learn to talk science as a part of understanding of the science (VanLehn *et al.*, 2000). Teaching is more difficult when the student organizes and talks about knowledge in a different way than the tutor does (Woolf & McDonald, 1984).

11. **Keep track of correct answers**

Instructional systems 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. Some systems try more specific or simpler version of questions to keep better track of progress. (Rose *et al.*, 2001).

12. **Prioritize learning tasks**

    Tutoring systems often handle multiple sub-tasks using priority rules that look at the duration and type of task. For example, systems can focus on errors before omissions and shorter fixes before longer fixes, prior steps before later steps, etc (Collins & Stevens, 1982).

13. **Limit the nesting of the lesson** Tutoring dialogue is sometimes controlled by limiting the amount of subdialogues, which helps the student keep track of the lesson topics (VanLehn *et al.*, 2000).

14. **Summarize back to teacher what was learned**

    Many educational systems summarize the highlights at the end of the lesson (Woolf & McDonald, 1984; McDonald, 1981). For example, EXCHECK prints out review of the proof for the student to give a clear picture of what has been done (McDonald, 1981). In some systems, when the tutor has given several hints, a summary may be given to ensure that the student has correct information just in case the student gave right answer by following hints without understanding the procedures (Woolf & Allen, 2000).

15. **Assess learned knowledge**

    In their dialogs with human tutors, students often indicate how well they understand the topic as well as what has been learned (Fox, 1993). Also some educational systems have a way of isolating the weaknesses in the student's knowledge and propose further lessons on those areas (Burton & Brown, 1979).

These fifteen principles can be related to the techniques used in existing knowledge acquisition tools, as described in section 5. Before we discuss the relations, the next section gives a short introduction and background on interactive knowledge acquisition tools.

# 4 An Analysis of Interactive Knowledge Acquisition Tools

There have been various knowledge acquisition approaches that researchers have undertaken over the years. The techniques used range from cognitive theories of expertise and learning, case-based

| *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 (Bareiss *et al.*, 1990) | Users specify cases, tool explains their classification. |
| SALT (Marcus & McDermott, 1989) | To acquire constraints and fixes for its underlying engine for configuration design. |
| SEEK2 (Ginsberg *et al.*, 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 2: Some Interactive Knowledge Acquisition Tools.

reasoning and analogy, non-monotonic theory revision, induction and machine learning, knowledge engineering approaches, analysis of knowledge interdependencies and buggy knowledge, agent-based interaction, etc. This section presents a brief survey of the representative of the literature and then discusses their typical components and the kinds of meta-knowledge that they bring to bear in order to support users.

## 4.1 A Brief Survey of Interactive Knowledge Acquisition Tools

The interactive knowledge acquisition tools summarized in Table 2 illustrate different approaches that researchers have undertaken over the years and are representative of the literature. Below is a brief summary of the ten tools.

**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 helps the user to 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 TEIREISIAS used that context and the explanations of the trace to help the user debug the new knowledge. TEIREISIAS derived rule models by generalizing among similar rules, and alerted users when a new rule deviated from the model.

### 4.2 A Functional Analysis of Interactive Knowledge Acquisition Tools

Figure 2 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 knowledge 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).

These capabilities are supported by knowledge acquisition tools by drawing from a variety of knowledge sources:

- *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).

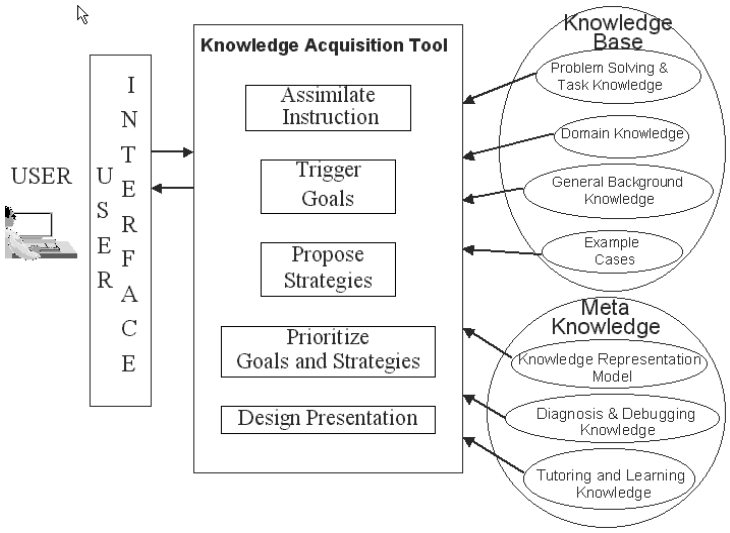



FIGURE 2: A Modular View of Interactive Knowledge Acquisition Tools.

- *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).

These are also illustrated in Figure 2.

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/Learning principle | Acquisition Function | | | | |
|---|---|---|---|---|---|
| | Assimilate Instruction | Trigger Goals | Propose Strategies | Prioritize Goals and Strategies | Design Presentation |
| P1: Introduce lesson topics and goals | | EXPECT SEEK2 | | | |
| P2: Use topics of the lesson as a guide | SALT | SEEK2 | EXPECT | | SALT |
| P3: Subsumption to existing cognitive structure | PROTOS | | TEIREISIAS | | PROTOS, SALT |
| P4: Immediate feedback | PROTOS | INSTRUCTO-SOAR | TEIREISIAS | | EXPECT |
| P5: Generate educated guesses | | TEIREISIAS | EXPECT | | |
| P6: Keep on track | | | | | |
| P7: Indicate lack of understanding | INSTRUCTO-SOAR | | | | INSTRUCTO-SOAR |
| P8: Detect and fix "buggy" knowledge | TAQL | EXPECT CHIMAERA | | | PROTOS, SEEK2 TEIREISIAS |
| P9: Learn deep models | | | | | |
| P10: Learn domain language | | | | | |
| P11: Keep track of correct answers | | SEEK2 | | | |
| P12: Prioritize learning tasks | | | | EXPECT | |
| P13: Limit the nesting of the lesson | | | | | |
| P14: Summarize what was learned | | | | | |
| P15: Assess learned knowledge | | KSSn | | | |

TABLE 3: Tutoring and learning principles used in interactive acquisition tools.

# 5 Tutoring and Learning Principles in Existing Interactive Acquisition Tools

In section 3, we described our analysis of tutoring and educational literature and presented fifteen principles that humans and computers exploit to make teaching and learning more effective. We noticed that many of the principles in learning and tutoring 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 3 summarizes our analysis where the principle was applied in specific acquisition tools. The columns indicate the particular functionality as outlined in Figure 2.

1. **Introduce lesson topics and goals**

   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.

2. **Use topics of the lesson as a guide**

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.

3. **Subsumption to existing cognitive structure**

   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.

4. **Immediate feedback**

   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.

5. **Generate educated guesses**

   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.)

6. **Keep on track**

   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.

7. **Indicate lack of understanding**

   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.

8. **Detect and fix "buggy" knowledge**

   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.

9. **Learn deep models**

   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.

10. **Learn domain language**

    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.

11. **Keep track of correct answers**

    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.

12. **Prioritize learning tasks**

    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).

13. **Limit the nesting of sub-lessons**

    Acquisition tools do not explicitly have a way of controlling the amount of nesting, although it would help our acquisition tools keep track of what is going on as much as it helps a human student.

14. **Summarize what was learned**

    Acquisition tools do not summarize what they have learned.

15. **Assess learned knowledge**

    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.

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 3 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 in all five functions of the acquisition process shown in the table.

Acquisition tools should be able to structure the dialogue with the user in tutoring terms. They 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. The following section presents our initial attempt to address these issues.

# 6 SLICK: Declarative Representation of Tutoring and Learning Principles for Proactive Acquisition

The above analyses of tutoring and learning principles and their relations to knowledge acquisition tools show that the tutoring and learning 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 use of the principles are rather implicit, limiting their influence in the interactions with the user.

Our approach is to having tutoring and learning principles represented explicitly and declaratively and to be aware of the level of competence and confidence of the knowledge they are acquiring. This would enable acquisition tools to reason in terms of the teaching and learning process and to make interaction with the user dynamically generated given the situation at hand.

The following presents the capabilities we provide to acquisition tools based on the principles we have compiled so far.

- Acquisition tools should be able to represent *acquisition goals* explicitly. Many of the tutoring principles suggest a more goal-oriented behavior for acquisition tools. Having acquisition goals explicitly and declaratively is key to making a tool truly proactive because it could then steer the dialogue with the user to work towards those goals. The goals that are achieved at each point during the dialogue represent the progress made towards acquiring the desired body of knowledge.
- Acquisition tools should have *acquisition strategies* in order to understand and actively pursue

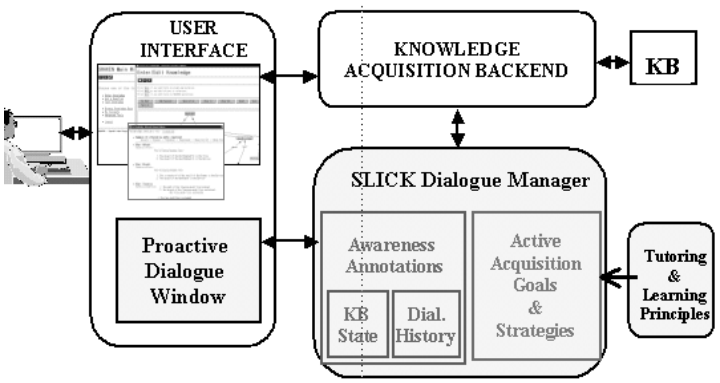



FIGURE 3: SLICK architecture

what is involved in learning about a new topic. Acquisition strategies outline how to achieve acquisition goals. Because so many things are unknown to the system during the lesson, these strategies can only be pursued under the user's guidance and in a mixed-initiative interaction.

- Acquisition tools should have *awareness* of what they have learned already and what they do not know about yet, so that they can better assess their competence and confidence in specific topics, and steer the dialogue with the user in directions that improve their body of knowledge on both counts.

SLICK is developed as a front-end to existing basic acquisition tools by embodying these capabilities. Figure 3 shows the architecture of the system. The boxes in gray represent the SLICK components that extends basic acquisition tools. The arrow between "Tutoring & Learning Principles" and "SLICK Dialogue Manager" means that the general tutoring and learning principles are operationalized based on the target knowledge to be acquired and the features of the given acquisition tool. For example, tools that acquire different forms of knowledge (such as problem-solving knowledge vs. concepts) may need different operational goals because they have different subcomponents and functions to build up knowledge bases. The details are described below. Actions done by the user through the basic acquisition tool are intercepted by our system. While the backend tool will update the backend knowledge base and its own user interface, SLICK will update its own structures and user interface.

## 6.1 Acquisition Goals

Figure 4-(a) shows the general acquisition goals that we currently use. The tutoring and learning principles in Table 1 are mapped into these goals according to the main activities in acquisition tools. The principles used are shown in parentheses. Some of the principles (such as P4, P5, and P12) are used as strategies than goals as described below. The goals and principles not used currently (such

1) SET UP LESSON AND CHECK BACKGROUND: (P1, P2)
- 1.1. Get the overall topic and purpose of the lesson.
- 1.2. Acquire any assumed prior knowledge before pursuing the lesson.

2) ACCEPT AND RELATE NEW DEFINITIONS: (P3, P6)
- 2.1. Accept new definitions.
- 2.2. Ensure that new knowledge is specific as possible.
- 2.3. Ask the user to be complete when enumerating items in terms of the elements and in terms of the significance of the order given.
- 2.4. Get all the information required when existing knowledge indicates it must be provided.
- 2.5. Make all new definitions consistent with existing knowledge.
- 2.6. Connect all new items with the topic of the lesson, keeping on track.

3) TEST AND FIX: (P8, P9, P15, P7)
- 3.1. Test the new body of knowledge and generate tests for the aspects that have not been thoroughly tested.
- 3.3. Detect missing knowledge and fix problems that result from self-checks or from user's indications.
- 3.4. Ensure user checks the reason for the answers, not just the answers.
- 3.5. Confirm new answers that change in light of new knowledge over what the user had seen the answer to be earlier.

4) FIT WITH EXISTING KNOWLEDGE STRUCTURES: (P3)
- 4.1. Establish identity of new objects by checking if existing objects appear to be the same.
- 4.2. Generalize definitions if analogous things exist and there could be plausible generalizations.

5) ACHIEVE PROFICIENCY: (P10)
- 5.1. Acquire domain terms to describe new knowledge.
- 5.2. Learn to reason/generate answers efficiently and with shorter explanations.

6) REACH CLOSURE ON LESSON: (P14, P2, P11, P15)
- 6.1. Ensure that the purpose/topics of the lesson were covered and the test questions appropriately answered.

(a) Acquisition Goals

- Annotations to the new body of knowledge:
  - For each lesson: purpose, assumed background, sub-lessons, overall competence and confidence (based on tests)
  - For each k item: connection to lesson, relation to other items, identity wrt other items, possible analogies and generalizations, domain terminology details, competence, confidence
  - For each axiom of a k item: required information, generality, completeness, confidence
- Annotations to the dialogue history:
  - For each user action: changes to the annotations to the new knowledge, acquisition goals achieved and/or activated, possible future KA strategies

(b) Awareness Annotations

FIGURE 4: Acquisition Goals and Awareness Annotations.

as P13) can be incrementally added in the future. We found it useful to group acquisition goals into six themes, each with a different emphasis on what is being learned. For example, acquisition goal 1.1 (Get the overall topic and purpose of lesson) can be adopted in acquisition interface in order to make the lesson more coherent. 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. Current acquisition tools do not have any basis to evaluate or pursue depth in their knowledge base. One thing acquisition tools can do is to provide a way of enforcing users to check how the answers were generated to check that the system provides the right answer for the right reasons (Goal 3.3).

These high level acquisition goals are mapped to more specific goals to accommodate different acquisition tools and representations. For example, in some cases the purpose of the lesson can be specified as a suite of types of test questions that the system should be able to answer correctly after the lesson. In other cases it could be given as an exhaustive list of new terms to be defined during the lesson.

### 6.2 Learning Awareness

We represent awareness with two kinds of annotations: annotations to the new body of knowledge acquired and annotations to the interaction history.

A new body of knowledge based is associated with the lesson/purpose/topic of the session(s) where it is acquired. We consider a new body of knowledge as a collection of *knowledge items* (e.g., concepts, problem solving methods or rules, instances, examples), each with an associated set of *axioms* (e.g., range constraints, subclass relations) that embody the knowledge about that item. We record this structure (axioms associated with items, items associated with lessons) and extend it as the user goes through the session. This basic structure is annotated with meta-level information about its status, where we aim to capture how much is known about that lesson/item/axiom and how confident the system is about it. Figure 4-(b) shows the annotations that we use.

A novel feature here is the focus on keeping track of what is known, not just on what is not known. Traditionally, the focus of acquisition tools has been on errors and gaps in the knowledge base.

In some sense, a knowledge base is never complete, so these annotations should ideally become part of the knowledge base or at least in an accessible record of how a body of knowledge was acquired by the system in certain sessions with certain users.

Annotations to the interaction history record what action the user took at each point in time (e.g., define a concept as a subclass of another one, define a new role for that concept, test the knowledge with a question), and what progress resulted from that action in terms of the lesson at hand. The system notes the changes to the annotations of the body of knowledge that resulted from the user's action. In addition, the system records what learning goals have been achieved and what learning goals become active, as well as what strategies seem to make sense in order to achieve those goals. These annotations of the interaction history allow the system to share with the user its understanding of what it is learning as the lesson progresses.

### 6.3 Acquisition Strategies

Acquisition tools can follow strategies by understanding the characteristics of acquisition tasks and the kinds of support users may need. For example knowledge base modifications usually require changing several related parts and tools can support users in completing complex acquisition tasks by exploiting prototypical procedures (Tallis & Gil, 1999). Acquisition tools can also use expectations of what the user has to do based on the interdependencies among the different components of the knowledge base (Kim & Gil, 1999). The system can enforce constraints in the knowledge

based system, look for missing pieces of knowledge, and finally propose potential ways of resolving the issues.

Acquisition strategies can be also formulated based on the tutoring and learning principles shown in Table 1. For example, the system can attempt to be a good learner by making educated guesses when possible, and by noting surprise if its guesses are wrong (P5). The feedback from the system can be controlled based on the status of learning and the utility of the interruption (P4). Priority schemes for acquisition goals also help narrow down which strategies the user may pursue next (P12). If the system can use some heuristics to determine that an instantiation of an acquisition strategy is more likely than others (for example, by drawing an analogy with existing knowledge), then that more concrete strategy would be shown to the user. Strategies that achieve more than one acquisition goal are considered more likely. For example, a goal to fill in required information of an item and a goal to connect a new item to the lesson can be both solved if the two items are connected (assuming that the first item is already connected to the lesson). Also, we use the six categories of acquisition goals (shown earlier) to order the active goals and present them to the user in that sequence, where the more likely goals are shown at the top. This is because those six categories reflect stages that users typically follow in an acquisition dialogue, although users often jump from one to the other as they see fit.

Because acquisition strategies drive the interaction with the user, acquisition tools need to strike a balance between exploring and covering all possible strategies that users can follow and not overwhelming them with options that they are unlikely to choose in the first place. This is a very challenging problem and an area of future work.

## 7 Using SLICK: Preliminary User Feedback

SLICK has been used for acquiring two very different types of knowledge: biological process models and military plans. The basic acquisition tools were developed as a part of the DARPA RKF (Rapid Knowledge Formation) that aims to help end users, i.e., people without formal training in computer science, develop knowledge bases. The acquisition tools had various support for the users, including background knowledge that the users can draw on, question answering system, verification and validation mechanism (Kim & Gil, 2001), graphical interfaces to let the users enter knowledge without knowing formal logic (Clark *et al.*, 2001), etc. However, the tools were rather passive in organizing various acquisition tasks and they were not able to actively participate in the learning process. SLICK was built as a front-end to these acquisition tools in order to make them more proactive, able to efficiently reason about learning activities with initiative in its dialogue

with the user.

Figure 5 shows the SLICK interface for acquiring military plans (army courses of actions). With the basic entry tool(Forbus *et al.*, 2003), users describe their plans in terms of the steps (such as attack, seize, destroy, etc.) and the objects involved (military units, terrain features, etc.). Here SLICK is presenting a report on a plan being entered by a military officer, pointing out how the system is understanding the plan.

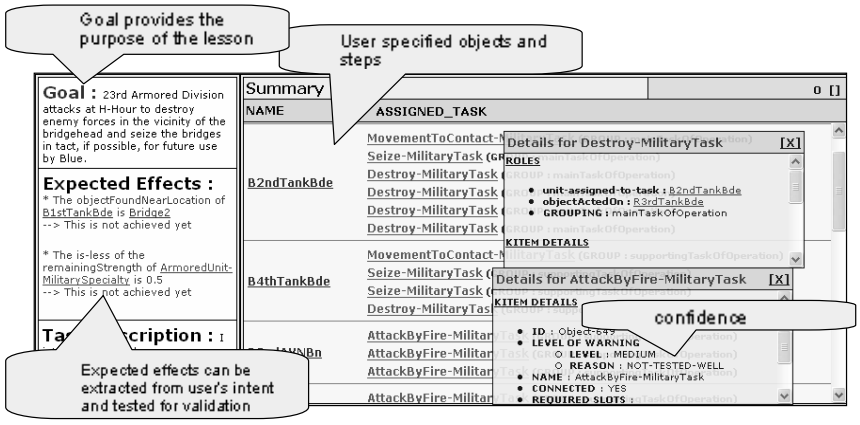
As shown in Figure 5-(a), SLICK keeps track of the lesson goal and the user's intention (e.g., expected effect), which can be used to guide the user as well as to check if the plan is valid (e.g., intended effects are achieved). The summary window shows how the plan is being built, illustrating the essential elements of the plan: the list of involved objects and their tasks. It highlights the objects with potential problems (such as unassigned units) in red and confident subtopics are shown in blue. The user can check details of each item by clicking the interested items, as shown in Figure 5-(a). For example, SLICK presents confidence on knowledge items based on the number of times they were involved in testing.

In past work, in user evaluations with other acquisition tools, we have found that subjects often had difficulty in understanding how they are making progress (Kim & Gil, 2000), and here the officers commented that the SLICK functionalities are very helpful for it. One of the officers said that the status report from SLICK is not only useful for the plan builder (the commander) but also can be sent out to other people (military units) who participate in the plan.

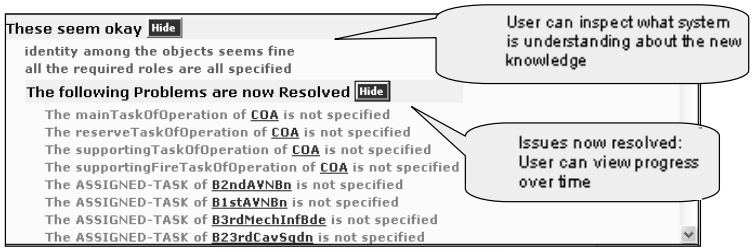
Towards the end of the lesson (i.e., building a plan), SLICK confirms that all the required roles (such as the information that existing knowledge indicates it must be provided) are specified, and identity among the objects are fine (none of the existing objects appear to be the same) (Figure 5-(b)). The user can view the progress by checking the issues resolved over time.

When SLICK notices remaining issues, it also collects the sources of the problems so that it can help users understand the problems better. For example, Figure 5-(c) shows that there is an inconsistency between the plan and existing definitions in the KB because in the existing definitions, the 'objectActedOn' should be a military unit (ModernMilitaryUnitDeployable), but currently the user has assigned a phase line (a terrain feature) for it.

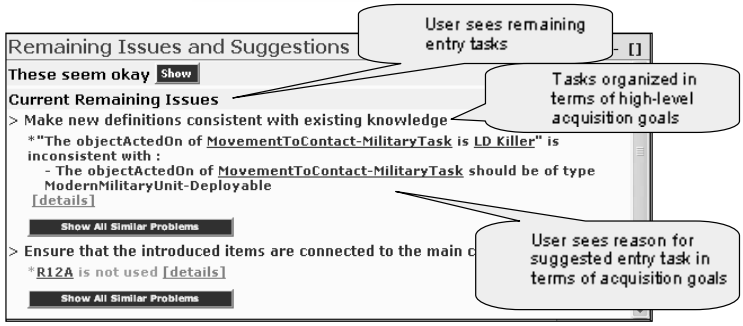
Note how SLICK's learning principles have derived these output. For example, in the figure, SLICK reports its understanding of the lesson and the remaining issues in terms of its goals such as "Make new definitions consistent with existing knowledge", "Ensure that the introduced items are connected to the main concept", "Ensure that the required roles are all specified", "Establish identity among the items", etc.




(a) Lesson overview (goal, expected effects, items being built)




(b) Showing progress over time




(c) Showing remaining issues

FIGURE 5: Acquiring military plans with SLICK.

SLICK has been also applied to acquiring biological process models where learning goals that are active given the state of the lesson are shown to the users.

From these exercises, we have found that SLICK may help users more when the acquisition tasks have many steps involving various sub-tasks (searching, editing, testing, fixing, etc.) and the tasks require keeping track of the context of what needs to be achieved. For example, SLICK may be effectively used in building instructional systems. However, if the given acquisition task is very simple, with a small number of steps, then SLICK may not provide much help.

# 8 Discussion: Relations to Building Intelligent Tutoring Systems

We believe that the work in educational systems and acquisition systems share a lot of issues and they may be able to contribute to each other in many ways. For example, there have been recent interests in acquiring knowledge for intelligent tutoring systems (Murray, 1999). Recent advances in knowledge acquisition research, including our own work (Blythe *et al.*, 2001; Kim & Gil, 1999; Tallis & Gil, 1999; Blythe & Ramachandran, 1999; Tallis *et al.*, 1999), may point to various ways of improving existing tools for tutoring systems. The SLICK framework may further enhance the capabilities since it is designed to be independent of any acquisition tools. SLICK's tutoring and learning principles can be mapped into operational tutoring and learning principles based on the features of the tool. We believe that this enhancement will help users develop the knowledge and models needed in ITS more effectively. Moreover, SLICK's learning goals may be useful for developing learner's models as well as building domain models.

There are some more potential ways of contributing to tutoring systems. We recognize that students taking an active role and assessing their progress in learning is emphasized as an important factor in cognitive development (Chi *et al.*, 1994; White & Frederiksen, 2000; Bandura, 1971), and it may be closely related to our goal: building a proactive learner. We are currently drawing ideas from educational research, but as we gain more experience, we hope our results can contribute to formalizing the student's role in teaching and learning in general.

Another important issue in SLICK is that users of the acquisition tools are not necessarily skilled teachers by nature, so being in a position to teach a computer is a challenge for them. In developing proactive/good learners, we also aim to make our system (students) supplement the user's limitations as a teacher (and knowledge engineer). That is, the system need 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. Some of the issues that arise in building good tutoring techniques here may be transferable to tutoring systems.

Finally, acquisition systems can be excellent tools for teaching and learning where students learn by assuming the role of teacher (knowledge provider) (Davis *et al.*, 2003). This "learning by teaching" paradigm seems to be very effective and powerful, and we believe that developing such environment will greatly benefit from knowledge acquisition techniques.

# 9 Conclusion

This paper summarizes our investigation on exploiting tutoring and learning techniques in interactive knowledge acquisition tools. From the literature on human learning and on educational systems, we extracted fifteen principles that teachers and learners seem to follow in pursuing tutoring dialogues. These principles suggest valid strategies for interactive acquisition tools to mirror in having a dialogue with users that are teaching them new knowledge. The paper also presented a survey of interactive acquisition tools and an analysis of how some of these principles are used in existing tools for different functions and purposes. Our investigation points out that the fifteen tutoring and learning principles that we identified could be used more thoroughly in interactive knowledge acquisition tools by incorporating them into different functions of the tools.

The fifteen tutoring and learning principles described here should be augmented in the future with additional research in the vast literature of education and learning. For example, while our focus so far has been typical student and teacher interactions it should be useful to extract principles that brighter students use as well as principles that students follow when confronted with inexperienced teachers. By incorporating these additional principles in interactive acquisition tools they could be turned not just into good students but into exceptionally bright students. Also, since we do not expect their users to be trained in teaching or tutoring techniques it would be useful for the tools to supplement that with knowledge about teaching that typical students are not expected to have.

We also have presented a new approach for interactive knowledge capture that can be used to extend existing tools with acquisition goals, learning strategies, and awareness annotations over the current state of the knowledge base in terms of its completeness and competence. Our system presents users with useful information regarding the progress made throughout the dialogue, current status of the new body of knowledge, goals that remain to be addressed, and suggested strategies to accomplish those goals. We believe that the information that the system is capturing about its current knowledge and its progress during the acquisition dialogue gives the user a crucial tool for externalization, i.e., an external record of the teacher/student interaction that helps the user visualize where the lesson is at, relieving users of a significant burden during the acquisition process.

## Acknowledgments

This research was funded by the DARPA Rapid Knowledge Formation (RKF) program with award number N66001-00-C-8018. We would like to thank Ken Forbus, Lewis Johnson, Jeff Rickel, Paul Rosenbloom, and David Traum on their insightful comments on earlier drafts.

## References

Aleven, V. & Koedinger, K. (2000). The need for tutorial dialog to support self-explanation. In *Proceedings of the AAAI Fall Symposium on Building Dialogue Systems for Tutorial Applications.*

Anderson, J. R., Conrad, F. G., & Corbett, A. T. (1989). Skill acquisition and the lisp tutor. *Cognitive Science*, 13:467–506.

Ausubel, D. (1968). *Educational psychology: A cognitive approach.* New York, Holt, Rinehart and Winston.

Bandura, A. (1971). *Social Learning Theory.* New York, General Learning Press.

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., & Gil, Y. (2001). An integrated environment for knowledge acquisition. In *Proceedings of the IUI-2001.*

Blythe, J. & Ramachandran, S. (1999). Knowledge aquisition using an english-based method editor. In *Proceedings of the Twelfth Knowledge-Acquisition for Knowledge-Based Systems Workshop.*

Brown, J. S., Burton, R., & de Kleer, J. (1982). Pedagogical natural language and knowledge engineering techniques in SOPHIE I, II, III. In Derek, S. & Brown, J. S., (Eds.), *Intelligent Tutoring Systems.* New York, Academic Press.

Brown, J. S. & Burton, R. R. (1978). Diagnostic models for procedural bugs in basic mathematical skills. *Cognitive Science*, 2:155–191.

Burton, R. & Brown, J. (1979). An investigation of computer coaching for informal learning activities. *International Journal of Man-Machine Studies*, 11:5–24.

Carbonell, J. R. (1970). AI in CAI: An artificial intelligence approach to computer-assisted instruction. *IEEE Transactions on Man-Machine Systems*, 11(4):190–202.

Chi, M., deLeeuw, N., Chiu, M., & Lavancher, C. (1994). Eliciting self-explanations improves understanding. *Cognitive Science*, 18.

Clancey, W., (Ed.) (1987). *Knowledge-Based Tutoring:The GUIDON Program.* MIT press.

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. In *Proceedings of K-CAP-2001.*

Collins, A. & Stevens, A. L. (1982). Goals and strategies of inquiry teachers. *Advances in Instructional Psychology*, 2:65–119.

Core, M. G., Moore, J. D., & Zinn, C. (2000). Supporting constructive learning with a feedback planner. In *Proceedings of the AAAI Fall Symposium on Building Dialogue Systems for Tutorial Applications.*

Davis, J., Leelawong, K., Belynne, K., Bodenheimer, B., Biswas, G., Vye, N., & Bradford, J. (2003). Intelligent user interface design for teachable agent systems. In *Proceedings of the IUI-2003.*

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.

Festinger, L. (1957). *A Theory of Cognitive Dissonance.* Stanford University Press.

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

Forbus, K., Usher, J., & Chapman, V. (2003). In *Proceedings of the Intelligent User Interfaces Conference*, pp. 61–68.

Fox, B. (1993). *The Human Tutorial Dialog Project.* Lawrence Erlbaum.

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

Gentner, D., Holyoak, K. J., & Kokinov, B. N., (Eds.) (2001). *The analogical mind: Perspectives from cognitive science.* MIT press.

Gil, Y. & Melz, E. (1996). Explicit representations of problem-solving strategies to support knowledge acquisition. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence.*

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

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

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

Kim, J. & Gil, Y. (2000). Acquiring problem-solving knowledge from end users: Putting interdependency models to the test. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*.

Kim, J. & Gil, Y. (2001). Knowledge analysis on process models. In *Proceedings of IJCAI-01*.

Koedinger, K., Anderson, J., Hadley, W., & Mark, M. (1997). Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education*, 8:30–43.

Kulik, J. & Kulik, C. (1988). Timing of feedback and verbal learning. *Review of Educational Research*, 58:79–97.

Lepper, M., Woolverton, M., Mumme, D., & Gurtner, J. (1993). Motivational techniques of expert human tutors: Lesson for the design of computer-based tutors. In Lajoie, S. & Derry, S., (Eds.), *Computers as Cognitive Tools*, pp. 75–105. Hillsdale.

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

McDonald, J. (1981). The EXCHECK CAI system. In Suppes, P., (Ed.), *University-level Computer-assisted Instruction at Stanford: 1968-1980*. Stanford.

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

McKendree, J. (1990). Effective feedback content for tutoring complex skills. *Human Coputer Interactions*, 5:381–413.

Merrill, D. C., Reiser, B. J., Ranney, M., & Trafton, J. G. (1992). Effective tutoring techniques: A comparison of human tutors and intelligent tutoring systems. *The Journal of the Learning Sciences*, 2:277–305.

Murray, T. (1999). Authoring intelligent tutoring systems: An analysis of the state of the art. *International Journal of Artificial Intelligence in Education*, 10:98–129.

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

O'Shea, T. (1979). A self-improving Quadratic tutor. *International Journal of Man-Machine Studies*, 11:97–124.

Rose, C. P., Jordan, P., Ringenberg, M., Siler, S., VanLehn, K., & Weinstein, A. (2001). Interactive conceptual tutoring in Atlas-Andes. In *Proceedings of AI in Education.*

Sleeman, D. H. (1984). Inferring student models for intelligent computer-aided instruction. In Michalski, R. S., Carbonell, J. G., & Mitchell, T. M., (Eds.), *Machine Learning: An Artificial Intelligence Approach*, pp. 483–510. Springer.

Stevens, A. & Collins, A. (1977). The goal structure of a Socratic tutor. In *Proceedings of the National ACM Conference.*

Tallis, M. & Gil, Y. (1999). Designing scripts to guide users in modifying knowledge-based systems. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence.*

Tallis, M., Kim, J., & Gil, Y. (1999). User studies of knowledge acquisition tools: Methodology and lessons learned. In *Proceedings of the Twelfth Knowledge-Acquisition for Knowledge-Based Systems Workshop.*

VanLehn, K., Freedman, R., Pamela, J., Murray, C., Osan, R., Ringenberg, M., Rose, C., Schulze, K., Shelby, R., Treacy, D., Weinstein, A., & Wintersgill, M. (2000). Fading and deepening: The next steps for Andes and other model-tracing tutors. In *Proceedings of ITS-2000.*

Wenger, E., (Ed.) (1987). *Artificial Intelligence and Tutoring Systems.* Morgan Kaufmann.

White, B. & Frederiksen, J. (2000). Metacognitive facilitation: An approach to making scientific inquiry accessible to all. In Minstrell, J. & van Zee, E., (Eds.), *Inquiring into Inquiry Learning and Teaching in Science.* ACM Press.

Woolf, B. & Allen, J. (2000). Spoken language tutorial dialogue. In *Proceedings of the AAAI Fall Symposium on Building Dialogue Systems for Tutorial Applications.*

Woolf, B. P. & McDonald, D. D. (1984). Building a computer tutor: Design issues. *IEEE Computer*, 17(9):61–73.

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

Zhou, Y., Freedman, R., Michael, M. G. J., Rovick, A., & Evens, M. (1999). What should the tutor do when the student cannot answer a question? In *Proceedings of FLAIRS-99*.