To appear in the International Journal of Human-Computer Studies (IJHCS), 2007.

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
phone:(310) 448-8769
fax: (310) 823-6714

Abstract

This paper argues that interactive knowledge acquisition systems would benefit from a tighter and more thorough incorporation of tutoring and learning principles. Current acquisition systems learn from users in a passive manner, and could instead be designed to incorporate the proactive capabilities that one expects of a good student. We first describe our analysis of the literature on teacher-student interaction and present a compilation of tutoring and learning principles that are relevant to interactive knowledge acquisition systems. 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 how computers learn from users. We present our design and an initial implementation of an acquisition dialogue system called *SLICK* that represents acquisition principles and goals explicitly and declaratively, making the system actively reason about various acquisition tasks and generate its interactions dynamically. Finally, we discuss promising directions in designing acquisition systems by structuring interactions with users according to tutoring and learning principles.

1. Introduction

Transferring complex problem-solving 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 has been proposed. Some approaches accept rules from the user and check them against other existing rules (Davis, 1979; Ginsberg et al., 1985). Other approaches acquire knowledge suitable for specific tasks and problem solving strategies (Marcus and McDermott, 1989; Schreiber et al., 1999; Wielinga et al., 1992). Other systems focus on detecting errors in the knowledge specified by the user (Gil and Melz, 1996; McGuinness et al., 2000; Kim and Gil, 1999). Some systems use a variety of elicitation techniques to acquire descriptive knowledge (Gaines and Shaw, 1993; Clark et al., 2001; Shadbolt and Burton, 1989) 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 systems to build sizeable knowledge bases (Kim and Gil, 2000; Clark et al., 2001). However, the majority of the burden of the acquisition task still remains with the user: users have to decide how and when to enter knowledge, and of what nature.

There are some interesting parallels between an instructional system and a knowledge acquisition system. 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 acquisition systems, these roles are reversed. Acquisition systems 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. Acquisition systems should take the kind of initiative and collaborative attitude that one would expect of a good student, and become proactive learners instead of reacting to the user's actions. Ideally, acquisition systems should also be able to supplement the user's skills (or lack there of) 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 tutoring systems work has focused on creating good teachers.

We set off to investigate how the dynamics of tutor-student interactions could be used to make acquisition systems 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 (Koedinger et al., 1997; Forbus and Feltovich, 2001), 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.

Our work focuses on acquisition systems for capturing complex problem-solving knowledge or procedural knowledge rather than systems for capturing factual knowledge such as concepts, relations and instances (Denny, 2002; Eriksson et al., 1995; Sure et al., 2002). Also, we concentrate on interactive knowledge authoring systems, rather than systems that make use of examples and generalize them into a task representation such as programming by demonstration and learning apprentices (Cypher, 1993; Mitchell et al., 1985). Many educational systems concern how students use their knowledge in solving problems and investigate approaches for improving student performance (Wenger, 1987; Kearsley, 2005; Forbus and Feltovich, 2001). Similarly, systems for acquiring procedural knowledge aim at improving system performance and problem solving capabilities, and we can exploit some of the issues that the acquisition systems share with educational systems.

Our work is inspired by both 1) principles and dialogue strategies that are used by instructional systems in guiding student learning and 2) some of the general educational theories for learning and instruction (Chi, 2000; Clark and Haviland, 1977; Kearsley, 2005; Fox, 1993; de Koning et al., 2000; Merrill et al., 1992). Instruction aims at maximizing the effect of the processes that govern learning (Bransford et al., 1999; Kearsley, 2005; Clark and Haviland, 1977). Since teaching principles are concerned with the design of effective learning, there could be many useful strategies that we can adopt for designing learning skills for computers. For example, good teachers organize and draw from student experiences. Following this principle, the acquisition system could take an active role in organizing and relating its understanding of a lesson. This would address the issue that users of the acquisition system are not necessarily skilled teachers by nature, and interactive acquisition systems need to be effective and helpful to all users and should be able to cope with inexperienced teachers.

Not all of the tutoring and learning principles are relevant to interactive acquisition systems. For example, the theories on human perception that influence learning (Gibson, 1966), such as characteristics of stimuli for effective learning or how students interpret instructions given in natural language, are less applicable. There are other issues that interactive acquisition systems 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 for human students.

The contributions of this paper 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 could be useful in the context of interactive acquisition systems.

Second, we point out how existing knowledge acquisition systems use techniques that are related to widely used tutoring and learning principles. Based on these analyses we identify areas that the acquisition systems 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 system called *SLICK* (Skills for Learning to Interactively Capture Knowledge) that is built based on these observations. SLICK makes acquisition systems 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. SLICK prototypes have been developed for acquiring two very different types of procedural knowledge: biological process models and military plans.

The paper begins with a summary of our analysis of tutoring principles, describing fifteen learning principles that we believe can be immediately incorporated into current interactive acquisition systems. We then show how some existing acquisition systems use techniques that are related to these principles in some aspects of their functionality. We also present our design and an initial implementation of SLICK, and describe how SLICK could be used in assisting users. Finally, we discuss promising directions that we see in designing acquisition systems that incorporate tutoring and learning principles more thoroughly.

2. Teaching and Learning Principles Relevant to Interactive Knowledge Acquisition

We have been investigating various tutoring principles[1] used by human tutors and educational software (Fox, 1993; Wenger, 1987; Forbus and Feltovich, 2001). As described earlier, our work focuses on several aspects of teaching and learning that are more relevant to acquiring and validating problem-solving knowledge that is authored by users. Instructional systems contain other components such as student models and domain models. More comprehensive analyses of how such components are related to interactive knowledge acquisition systems are left for future work.

Table 1 shows a summary of the tutoring and learning principles that we found relevant to interactive knowledge acquisition.

Table 1 goes here.

The following describes the details on each principle.

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 student may already know and what the student should learn. In educational systems, such as Meno-Tutor (Woolf and 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 (Pask, 1975). 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 (Breuker et al., 1987). 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 (Scandura, 2004). 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 and 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 and 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

[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.

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 beliefs or 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 responses (Brown et al., 1982; Anderson et al., 1989; Bredeweg and Forbus, 2004). The studies of feedback in a variety of instructional contexts find that immediate feedback is much more effective than feedback received after a delay (Kulik and 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 and use high bandwidth communication in guiding 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 and Koedinger, 2000). Likewise, in the studies of human tutoring, students often display their understanding by finishing the tutor's utterance, and the tutor finds out what students understand by inviting their guesses (utterance completion strategy) (Fox, 1993).

6. **Indicate lack of understanding**

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

7. **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 immediately gets the student back on track (Carbonell, 1970). Some educational systems detect change of directions (Woolf and Allen, 2000) or check if the questions are irrelevant to the case at hand (Clancey, 1987).

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

Many educational systems have a goal of diagnosing the student's "bugs" (Woolf and McDonald, 1984; Stevens and Collins, 1977; Brown and Burton, 1978), and question answering is often used in checking a student's knowledge. However, simply reporting 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 answer, experienced tutors pick counter examples to highlight the problem (Collins and 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 and Koedinger, 2000; VanLehn et al., 2000). In some cases, in ensuring that the student understands the explanation, educational systems use a set of check questions (Rose et al., 2001). Rich domain models, knowledge-based reasoning, and qualitative reasoning capabilities have been used in some of the instructional systems that guide students according to the domain models (Bredeweg and Forbus, 2004; de Koning et al., 2000). Studies of human tutorial dialogue 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 the science (VanLehn et al., 2000) so that they can communicate what they learned in scientific terms. Teaching is more difficult when the student organizes and talks about knowledge in a different way than the tutor does (Woolf and 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 a more specific or simpler version of questions to keep track of progress more effectively (Rose et al., 2001).

12. **Prioritize learning tasks**

    Tutoring systems often handle multiple sub-tasks using some priority rules that take into account the durations and the types of the tasks (Breuker, 1990). For example, systems can focus on errors before omissions, shorter fixes before longer fixes, prior steps before later steps, etc. (Collins and Stevens, 1982). Teachers can prioritize topics based on a set of general principles and decide what topics are more important (Gibbons, 2001).

13. **Limit the nesting of the topics**

    Tutoring dialogue is sometimes controlled by limiting the amount of sub-dialogues, which helps the student keep track of the lesson topics (VanLehn et al., 2000). The limited capacity of a human learner's memory affects the number of topics that can be discussed or kept track of in the learning dialogue (Lefrancois, 1991; Clark and Haviland, 1977; de Koning et al., 2000; Sweller, 1988).

14. **Summarize what was learned**

    Many educational systems summarize the highlights at the end of the lesson (McDonald, 1981; Woolf and McDonald, 1984). For example, EXCHECK prints out a 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 the right answer by following hints without understanding the procedures (Woolf and 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 in those areas (Burton and Brown, 1979).

Our goal is developing front-end dialogue capabilities for knowledge acquisition systems based on some of the existing tutoring and learning principles (Kim and Gil, 2002). The above list provides a collection of instructional principles that are more relevant to developing dialogue capabilities for interactive knowledge acquisition systems (Kim and Gil, 2003). They address several main aspects of human cognitive processing and learning that are investigated in some of the literature on educational theory (Piaget and Inhelder, 1973; Kearsley, 2005; Ausubel, 1968; Clark and Haviland, 1977). First of all, learning involves relating existing cognitive structure to new material (Piaget and Inhelder, 1973; Ausubel, 1968; Scandura, 2004; Bartlett, 1958). Cognitive development is performed by interpreting new information with respect to existing cognitive structure, and adapting or expanding it in order to make sense of the new information. Based on this principle, teaching can be done by helping the student bridge between new learning materials and existing concepts. P1 (Introduce lesson topics and goals), P2 (Use topics of the lesson as a guide) and P3 (Subsumption to existing cognitive structure) are relevant to this aspect.

The limited capacity of human short-term memory imposes constraints on the number of topics that can be maintained for a learning dialogue (Lefrancois, 1991; Clark and Haviland, 1977; de Koning et al., 2000; Sweller, 1988). The teacher who understands this limitation can help students keep track of what is being learned with respect to the topics. P13 (Limit the nesting of lessons), P1 (Introduce lesson topics and goals), P2 (Use topics of the lesson as a guide), P7 (Keep on track), and P11 (Keep track of correct answers) are relevant to this.

"Feedback" is one of the controversial topics in developing instructional systems (Kulik and Kulik, 1988; Merrill, 1987). When and how to provide feedback have been discussed and evaluated by many researchers. P4 (Immediate feedback) is concerned with the timing of feedback and guidance of learning discourse, as some research shows benefits from immediate feedback (Kulik and Kulik, 1988). P15 (Assess learned knowledge) and P8 (Detect and fix "buggy" knowledge) contribute to feedback that is generated from an assessment of learning and its outcome.

Teachers and instructional systems organize and prioritize learning tasks and topics according to a set of their own principles or instructional models (Breuker, 1990; Collins and Stevens, 1982; Gibbons, 2001). For example, the teacher can decide what topic should be introduced at what time, what instructions should be given, what problems should be given in what order, etc. P12 (Prioritize learning tasks) and P7 (Keep on track) are relevant to this aspect.

Some teachers and instructional systems encourage deep learning approaches in order to achieve enhanced learning outcome (Bransford et al., 1999; Marton et al., 1984; VanLehn et al., 2000). Some instructional systems use rich domain models and knowledge-based reasoning to teach deep conceptual models and promote deep learning (Bredeweg and Forbus, 2004). P9 (Learn deep models), P10 (Learn domain language), P3 (Subsumption to existing cognitive structure), P12 (Prioritize learning tasks) and P15 (Assess learned knowledge) are relevant to some of the deep learning approaches.

Finally, some of the principles are from "desirable" student actions that are observed in student-teacher dialogues (Fox, 1993), such as P5 (generating educated guesses), P6 (indicating lack of understanding), or

P14 (summarizing what was learned back to the teacher). Although these are not a part of instructional strategies, they are very useful in developing learning principles and creating a good learning system.

These fifteen principles can be related to the techniques that have been used in existing knowledge acquisition systems, as described in Section 4. Before we discuss these relations, the next section gives a short introduction and background on some of the interactive systems for acquiring procedural knowledge.

3. An Analysis of Interactive Systems for Acquiring Procedural Knowledge

The goal of our analysis above is to use these principles to enhance existing knowledge acquisition systems by developing more proactive capabilities. The above teaching and learning principles provide a foundation for developing such capabilities. Our analysis of knowledge acquisition systems focuses on typical components and functional aspects of interactive systems for acquiring procedural knowledge. More comprehensive survey of knowledge acquisition systems, knowledge acquisition methodology, or systems for developing ontologies can be found in other work (Boose, 1989; Denny, 2002; Schreiber et al., 1999; Shadbolt et al., 1993; Sure et al., 2002).

There have been various knowledge acquisition approaches that researchers have undertaken over the years. The techniques used include 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, agent-based interaction, etc. This section discusses typical components and kinds of meta-knowledge that each work brings to bear in order to support users.

3.1 A Brief Summary of Some of the Interactive Systems for Acquiring Procedural Knowledge

The interactive knowledge acquisition systems summarized in Table 2 illustrate different approaches that researchers have undertaken over the years in acquiring procedural knowledge and are representative of the literature (Gil and Kim, 2002). The selection of systems covers aspects of typical knowledge acquisition capabilities and knowledge sources rather reporting the most competitive systems or a comprehensive survey.

Table 2 goes here.

EXPECT --- EXPECT (Blythe et al., 2001) is a system 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. EXPECT derives models of knowledge interdependencies to notify users of possible inconsistencies and mismatches, and exploits 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 and Laird, 1995) is a system 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 natural language sentences 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 and 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.

PROTEGE-II --- PROTEGE-II (Puerta et al., 1992) includes a library of problem-solving methods that are highly reusable across different domains. Associated with each method is an ontology that describes the kinds of domain-dependent knowledge used in each method. Each problem-solving method is also associated with a knowledge acquisition system that is designed to be customized to how that method works. The knowledge acquisition system is then used by domain experts to specify the domain-specific knowledge that the problem-solving method needs. Recent versions of PROTEGE draw from this research to provide ontology editors that are accessible to end users.

PROTOS --- PROTOS (Bareiss et al., 1990) helps users specify knowledge for classification tasks through examples. A user provides training cases, specifies the features of a new case, and the system classifies the case by matching its features with a set of categories. PROTOS shows the user an explanation for the classification. If the user disagrees with the explanation then the system asks for salient differences between the case at hand and the matched category, which are used to correct the matches.

SALT --- SALT (Marcus and McDermott, 1989) helps end users specify domain-specific knowledge for configuration design tasks. SALT is one of a family of systems that use a *role-limiting approach* to knowledge acquisition (Marcus and McDermott, 1989). Essentially, a system 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 is 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 correctly solves 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). Process steps and objects entered by users 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 system 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) is developed as an acquisition system for MYCIN. Users specify new rules in the context of an example problem, and TEIREISIAS uses that context and the explanations of the trace to help the user debug the new knowledge. TEIREISIAS derives rule models by generalizing similar rules, and alerts users when a new rule deviated from the model.

3.2. Functional Analysis of Interactive Knowledge Acquisition Systems

This section describes several aspects of interactive knowledge acquisition systems that are useful for understanding how we can develop proactive dialogue capabilities. We make use of the above knowledge acquisition systems in providing examples.

Figure 1 goes here.

Figure 1 shows a diagrammatic view of typical components used in various interactive knowledge acquisition systems. The functionality of an interactive knowledge acquisition system can be described along five dimensions, which we use 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 (EXPECT, SALT, TAQL, PROTEGE-II), or a graphical rendering (KSSn, SHAKEN).
- **Trigger goals**: The system analyzes its knowledge and generates *acquisition goals* of what knowledge it still needs to acquire. Many systems focus on detecting inconsistencies or gaps in the knowledge base, which generate the goals to fix them or seek the information missing (EXPECT, SEEK2, TAQL, TEIREISIAS). Other systems generate forms that elicit from the user all the required information (PROTEGE-II).
- **Propose strategies**: The system can generate possible strategies that the user could follow in achieving the acquisition goals. The system can also generate predictions of what strategy the user is more likely to pursue, or what answers could address the user's question (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 acquisition goals, which will make the acquisition process more efficient (EXPECT). Other approaches use pre-defined problem-solving methods to derive the sequence and nature of the questions to be asked from the user (PROTEGE-II).
- **Prioritize goals and strategies**: An acquisition system can help users further if it is able to organize and prioritize the active acquisition goals and candidate strategies, so that it can make more focused suggestions to the user. An acquisition system can help users further if it is able to organize and prioritize the active acquisition 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 system 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 system 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 system 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 (EXPECT, SALT). The system 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 systems leave it up to the user to figure out what to do and simply make all possible options available to them (KSSn, SHAKEN). The system may simply ask the user to review an explanation (PROTOS), check some aspect of the knowledge (KSSn), or confirm a hypothesis (TEIREISIAS). Other systems have customized forms that are designed to fit a problem-solving context (PROTEGE-II).

These capabilities are supported by knowledge acquisition systems 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 (e.g., SALT, TAQL, PROTEGE-II).
- *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).
- *Models of underlying knowledge representation*: Models of the underlying knowledge representation will determine how users need to formulate new knowledge. Although most acquisition systems are influenced by their underlying representations, in some systems more explicit models of exiting knowledge and knowledge to be captured guide the acquisition process (e.g., TEIREISIAS, KSSn, SEEK2, TAQL, PROTEGE-II).

- *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., EXPECT, TEIREISIAS).

The first four sources of knowledge are part of the system's knowledge base. In contrast, the models of the underlying knowledge representation as well as the diagnosis and debugging knowledge can be seen as *meta-knowledge*, that is, knowledge *about* the system's current knowledge base. In fact, meta-level reasoning and reflection capabilities have been explicitly considered in some knowledge-based systems (Maes and Nardi, 1988; Harmelen et al., 1992). However, they have not been exercised in developing interactive knowledge acquisition systems. Our goal is to understand meta-knowledge of tutoring and learning, and how interactive acquisition systems can exploit such knowledge more effectively.

4. Tutoring and Learning Principles in Existing Interactive Acquisition Systems

Table 3 goes here.

In Section 2, 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 observe that many of the principles in learning and tutoring are related to the techniques used in existing acquisition systems. 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 principles were applied in specific acquisition systems. The columns indicate particular functionality as outlined in Figure 1.

1. **Introduce lesson topics and goals**

   A *lesson* for a knowledge acquisition system can be a new procedure or a chunk of new knowledge that the system should acquire. For example, in the biology domain, a user may want to teach the process of RNA transcription based on existing knowledge of RNA polymerase, copying action, promoter, etc (Clark et al., 2001).

   Unlike in human tutoring, at any point users can choose to enter knowledge about any topic or any lesson. 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, which can be viewed as a statement of the goals of the lesson. PROTEGE-II prompts the user with customized forms aimed to acquire the information needed by a pre-defined problem-solving method.

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

EXPECT uses specific top-level tasks to check that any new knowledge specified solves some of the current subtask. If knowledge is given outside the subtask, EXPECT 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 drives the dialogue with the user toward fixing them. SALT has an implicit (and very high-level) topic for all sessions: to acquire knowledge for configuration design problems. SALT's interface asks users to specify only three kinds of knowledge (parameters, constraints, and fixes) that are relevant to configuration design problems.

3. **Subsumption to existing cognitive structure**

PROTOS takes a new example case provided by the user, and indexes it into one of several classes (or categories) of examples. It also presents the user with an explanation of the classification of the new example, to show how the new knowledge is incorporated into existing structures. PROTEGE-II assumes that the new knowledge acquired will be assimilated in a way that will be usable by the pre-defined problem-solving method. All knowledge acquired is asked from the user through customized forms based on that problem-solving method and organized according to their function and use by the method. TEIREISIAS creates generalized rule models from its rule base, and uses them to propose to the user additional conditions to newly defined rules. The interface and presentation of SALT is always based on the kinds of knowledge needed for configuration design.

4. **Immediate feedback**

PROTOS provides immediate feedback as a new case is assimilated by showing the user an explanation of its classification in the knowledge base. INSTRUCTO-SOAR generates clarification and follow-up questions for the user immediately after an instruction is given. TEIREISIAS proposes amendments to rules as soon as the user defines them. EXPECT analyzes the knowledge base after each user action and shows immediately an agenda of errors to resolve and tasks to do. PROTEGE-II requires the user to provide all necessary domain knowledge required by a form before concluding the session.

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. **Indicate lack of understanding**

INSTRUCTO-SOAR detects missing aspects of a task description specified by a user and generates follow up questions. EXPECT detects undefined terms that guide future dialogue with the user for defining them.

7. **Keep on track**

Acquisition systems 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 system to invoke, and 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 systems would never even notice that the user is deviating from a topic in any of these situations.

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

TAQL analyzes the knowledge specified by the user and points out errors based on static analysis. EXPECT detects 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 systems 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 they are dependent on implementation of any depth in the models.

10. **Learn domain language**

Acquisition systems 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-topics**

Since most acquisition systems are not constrained by limited memory capacity, unlimited amount of sub-dialogues can be supported. However, controlling the amount of nesting would help the user (the human teacher) keep track of what is going on, as it helps a human student.

14. **Summarize what was learned**

Acquisition systems 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 systems 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 assess if the knowledge was learned appropriately.

The above principles have only been used in some aspects of the functionality of acquisition systems, and are exhibited by some but not all the systems. 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 systems could more easily incorporate these principles in all five functions of the acquisition process shown in the table. The next section describes an interactive knowledge acquisition system that does this.

5. SLICK: Declarative Representation of Tutoring and Learning Principles for Proactive Acquisition

As we discussed in the previous section, acquisition systems use 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 system, 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 systems 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 systems into more proficient and proactive learners.

Acquisition systems 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. The termination of the dialogue should be delayed until the goals of the lesson are satisfied. Acquisition systems should exploit the topics of the lesson throughout the acquisition process. For example, topics can be used to narrow down the prior knowledge that is relevant to that portion of the dialogue, 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 systems 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 systems should be able to expose and assess the knowledge acquired so far, allowing the user to understand what the system has assimilated, and show 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 useful for these systems to convey whether they are confident on the answer, helping users identify further areas of improvement for future acquisition sessions.

SLICK is a new capability for interactive knowledge acquisition systems that exploits meta-knowledge about tutoring and learning principles. These principles are captured in three new meta-knowledge structures:

1) Representations of *acquisition goals*. Many of the tutoring principles suggest a more goal-oriented behavior for the system. Having acquisition goals explicitly and declaratively is a key to making a system 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 progress made towards acquiring the desired body of knowledge.

2) Representations of *acquisition strategies* in order to understand and actively pursue what is involved in learning about a new topic. Acquisition strategies outline how to achieve acquisition goals. Because many things are unknown to the system during the lesson (i.e. the given knowledge acquisition session), these strategies can only be pursued under the user's guidance and in a mixed-initiative interaction.
3) Representations of learning *awareness* of what the system has learned already and what it does not know about yet, so that it can better assess its competence and confidence in specific topics, and steer the dialogue with the user in directions that improve its body of knowledge on both counts.

Figure 2 goes here.

Figure 2 shows the architecture of SLICK that is integrated with an interactive acquisition system. The boxes in gray represent the SLICK components that are used in combination with an existing acquisition system (shown on the top of the figure). The arrow between "Tutoring & Learning Principles" and "SLICK" 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 system. For example, systems that acquire different forms of knowledge (such as process models vs. constraints for performing tasks) may need different acquisitionl goals because they have different subcomponents and functions to build up knowledge bases. Actions performed by the user through the basic acquisition system are intercepted by our system. While the backend system updates the backend knowledge base and its own user interface, SLICK will update its own meta-knowledge structures and user interface.

As the principles that SLICK uses do not depend on a particular domain or a given knowledge acquisition system, the main components of the system can be used for different systems with varying applications. In order to operationalize SLICK for a given system, the system developer needs to map the SLICK capabilities to the system features as described below.

5.1 Acquisition Goals

Figure 3 goes here.

Figure 3-(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 systems. The principles used are shown in parentheses. Some of the principles (such as P4, P7, and P12) are used as strategies rather than goals as described below. The goals and principles not used currently (such 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 an acquisition interface in order to make the lesson more coherent. There is no notion in acquisition systems that there is a lesson being started or ended, since at any point users can choose to enter knowledge about any topic. Current acquisition systems do not have any basis to evaluate or pursue depth in their knowledge base. One thing acquisition systems can do is to provide a way of enforcing users to check how the answers were generated to ensure that the system provides the right answer for the right reasons (Goal 3.3).

The above high level acquisition goals are mapped to more specific goals to accommodate different acquisition systems and representations. For example, in some cases the purpose of the lesson can be specified as a suite of test questions that the system should be able to answer correctly after the lesson. The purpose may also be given as an exhaustive list of problems to be solved during the lesson.

5.2 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 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* or *k-items* (e.g., problem solving methods or rules, examples, introduced concepts, etc.), each with an associated set of *axioms* (e.g., range constraints, method sub-method relations, concept roles) that embody the knowledge about that k-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 at capturing how much is known about that lesson/item/axiom and how confident the system is about it. Figure 3-(b) shows the annotations that we use.

A novel feature in SLICK is the focus on keeping track of what is known, not just on what is not known. Traditionally, the focus of acquisition systems 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 procedure for solving a problem, specialize the procedure by adding new constraints associated with the given problem, test the knowledge with a question), and what progress resulted from that action in terms of the lesson at hand. The system notes changes to the annotations of the body of knowledge that result from a user's action. In addition, the system records what acquisition goals have been achieved, what acquisition goals become active, and 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.

5.3 Acquisition Strategies

Acquisition strategies can be formulated based on the tutoring and learning principles shown in Table 1. For example, a system can attempt to be a good learner by making educated guesses when possible, and by pursuing corrections 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). The system can use some heuristics or educated guesses to determine that an instantiation of an acquisition strategy is more likely than others. More concrete strategies or strategies that achieve more than one acquisition goal can be considered more likely. For example, a goal to fill in required information for an item and a goal to connect a new item to the lesson can both be solved if the two items are connected (assuming that the first item is already connected to the lesson).

SLICK uses the six categories of acquisition goals (shown in Figure 3) 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.

5.4 The SLICK procedure

Figure 4 goes here.

Figure 4 shows the steps that SLICK follows in updating acquisition status and providing suggestions to the user. Given the current lesson such as a new complex procedure to acquire and *knowledge acquisition actions* such as modification of the procedure, SLICK assesses the current acquisition status based on the acquisition goals, and updates the awareness annotations that are shown in Figure 3-(b). When a user action during the knowledge acquisition session introduces a k-item in defining a new procedure, the system creates annotations for the introduced item with respect to the procedure, based on its required information (e.g. required role values or substeps needed), connections to other k-items, identity with respect to other k-items that are already introduced, etc. If a user action creates axioms for the k-item or modifies how the k-item is used, the related annotations are modified accordingly. For example, if all of the required role values were specified by the new action, the k-item will be annotated as complete with respect to the goal of getting all the required information (Goal 2.4).

SLICK also analyzes status changes by comparing the current status and the previous status assessment. That is, when required role values are entered by the user, its status changes from incompletion to completion and the change is explicitly annotated and linked to the earlier status annotations.

The suggestions are produced from these assessment results. For example, the system notifies to the user when there are repeated problems, such as the same errors found in multiple checks. Prior problems that are resolved, such as incomplete specification being completed are annotated and are reported to the user as progress. More details on SLICK's presentations and example reports are given in the next section.

The SLICK procedure can be used in two different modes: an "always active" mode and an "on-demand" mode. With the always-active mode, the procedure is called after each individual knowledge acquisition action or KB change, and the system reports the changes in the learning status with its suggestions. The user can see the changes in the reports from the SLICK interface. The on-demand mode allows the user to invoke the SLICK procedure only when he or she wants to check the status changes, or at the end of a knowledge acquisition session.

The SLICK procedure is general and applicable to acquisition systems that are designed to acquire complex problem solving or procedural knowledge. In order to use this procedure, the acquisition system should define the form of k-items and axioms that are used in the system, how individual checks in the procedures can be made (e.g. finding required roles), and how suggestions can be provided.

Table 4 goes here.

Table 4 shows some of the calls to the backend knowledge acquisition system and the knowledge base. In generating presentations to the user, SLICK combines the results from these calls with the annotations that it generates over time. Depending on the reasoning capabilities that are supported by the acquisition system, SLICK can perform the corresponding assessments. That is, SLICK checks rely on corresponding functions of the backend acquisition system.

6. SLICK Prototypes

SLICK has been used for acquiring two very different types of knowledge: biological process models and military plans. In this section, we show examples from the two cases.

6.1 Acquiring biological process models

SLICK has been layered over the functionality of SHAKEN (Clark et al., 2001). The backend knowledge acquisition system (SHAKEN) is built on the KM knowledge representation and reasoning system (Clark and Porter, 2006). KM is a frame based knowledge representation language with first-order logic semantics. The system supports reasoning about events using a situations mechanism. SHAKEN offers various kinds of support for users, including background knowledge that users can draw from, a question answering system, a verification and validation mechanism (Kim and Gil, 2001), graphical interfaces to let users enter knowledge without knowing formal logic, etc. SHAKEN enables users to specify process modes in terms of their substeps and the objects involved. Users model a substep as a type of event from the events available in the knowledge base (e.g. 'Collide', 'Move', etc), and specify the objects that fill the required roles for that event (e.g., 'agent', 'object', 'location', etc). SHAKEN is rather passive in organizing various acquisition tasks. A SLICK prototype was built for SHAKEN in order to make it more proactive and able to reason about learning activities with initiative in its dialogue with the user.

Figure 5 shows example suggestions and annotations generated by SLICK when the user is entering a biological process model. The particular scenario in Figure 5-(a) shows a process model for 'Bacterial Transcription' that combines two different versions built by an expert biologist. Each contains a subset of the problems that are discussed here and went unnoticed by the user. Figure 5-(b) shows the state of the knowledge base and the awareness annotations that the system keeps for the process model shown in Figure 5-(a). The purpose of the lesson is given in this system as a set of test questions, and the overall effects expected after running a simulation of the process. The system also shows the prior knowledge assumed with terms that the user searched in the knowledge base, such as 'Bacterial-DNA', 'Enzyme', etc. The k-items include events as substeps (e.g., 'Collide') and objects that play a role in those events (e.g., 'Promoter'). Inspecting these knowledge items, the user can check their status: 'Promoter' has not been connected to any substep of the process model. The role 'object' of the 'Recognize step' has not been yet assigned.

Figure 5 goes here.

Figure 5-(c) shows the current set of active acquisition goals, and the strategies suggested to the user in order to pursue each goal. The system can show achieved goals (for example, setting up the lesson and the background) and remaining goals as described below. The goals are ordered according to main activities in knowledge acquisition as shown in Figure 3, but the user can examine or pursue goals in any order. The top level suggestion to the user, based on an educated guess, is to assign 'Promoter' as the 'object' of 'Recognize', since that would accomplish two goals (connecting 'Promoter' to the lesson and assigning an 'object' to 'Recognize'). Some acquisition goals point to the missing knowledge that was noted in the state. Other acquisition goals relate individual k-items: determining whether two items defined by the user with the exact same descriptions are the same ('Base-Pair01' and 'Base-Pair02'), asking the user whether the list of substeps is complete, and ensuring that the parent of 'Bacterial-Transcription' is as specific as possible (its current parent, 'Scenario', is the top-level process in the knowledge base and there are several more specific ones). Finally, some acquisition goals shown here are the result of testing and fixing errors through a simulation that checks the conditions and effects of each step (Kim and Gil, 2001). In this example, the 'Make-Contact' step has a condition to check if its 'object' and 'base' roles are already in 'contact' (because if they already are then it is unnecessary to perform the step).

Figure 5-(d) shows the user how each of his or her previous actions accomplish and/or raise acquisition goals. For example, adding a 'next-event' link between 'Recognize' and 'Make-Contact' made the 'Make-Contact' step reachable (in simulating how the steps are executed). The resulting changes are also analyzed with respect to the previous status of the knowledge base that was analyzed before the changes are made. The user can ask to view previous states to further analyze the evolution of the KB.

6.2 Acquiring military plans

The second SLICK prototype was built for tasks for acquiring military plans (Army courses of actions). Using a graphical sketching 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.). The acquisition system is built on a Cyc knowledge base (Witbrock et al., 2003). In this example, SLICK shows a report on a plan being entered by a military officer, pointing out how the system is understanding the plan.

Figure 6 shows an example report from SLICK. In this prototype, the SLICK dialogue interface was extended to present the summary of the current lesson and the lesson progress more effectively. Although we used a different implementation of the dialogue interface, the underlying algorithm is the same. As shown in Figure 6-(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 6-(a). For example, SLICK presents confidence on k-items based on the number of times they were involved in testing, as shown in Figure 6-(c).

Figure 6 goes here.

When SLICK notices remaining issues, it also collects the sources of the problems so it can help users understand problems better. For example, Figure 6-(b) shows that there is an inconsistency between the plan and existing definitions in the KB. This has occurred because in the existing definitions, the 'objectActedOn' should be a military unit (ModernMilitaryUnitDeployable), but the user has assigned a phase line (a terrain feature) for it.

Figure 6-(b) also shows how SLICK's acquisition goals have derived the output shown, similar to Figure 5-(b). For example, SLICK reports its understanding of the lesson and the remaining issues in terms of its goals, such as: "Get the overall topic and purpose of the lesson", "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.

Towards the end of the lesson (i.e., building a plan), SLICK confirms that all the required roles (such as information that must be provided) are specified, and identities among objects are fine (none of the existing objects appear to be the same) (Figure 6-(c)). The user can see progress by checking the issues resolved over time shown in Figure 6-(b).

6.3 Initial User Studies

Unlike other sub-fields of AI, user evaluations of knowledge acquisition systems with end users are rare (Tallis et al., 2001). First, user evaluations are very costly. In areas like machine learning and planning, experiments often amount to running programs repeatedly on already existing test sets. The evaluation of a knowledge acquisition system requires that a number of subjects spend a fair amount of time doing the study, and for the experimenters to spend time and other resources preparing the experiment (often months) and analyzing the results. The Sisyphus evaluations report that the limited number of participants can be tracked back to the significant amount of resources required to tackle the knowledge-intensive task that was selected (Shadbolt et al., 1999). Second, most of the research in the field of knowledge acquisition concentrates on knowledge modeling (e.g., how a knowledge engineer models a task domain) and knowledge elicitation (e.g., techniques for interviewing experts). There are few efforts on developing systems for users and only some acquisition system developers have conducted usability studies (Kim and Blythe, 2003; Tallis et al., 2001; Kim and Gil, 2000; Tecuci et al., 2000). In many cases, the results are not fully reported in the literature. Third, this area of research can still be considered highly exploratory and not a good target for costly and conclusive experimentation (Self, 1993; Basili et al., 1986; Olson and Moran, 199).

This section reports on an informal study of the SLICK system by collecting feedback from end users. It also reports findings from user studies of a basic knowledge acquisition system (SHAKEN before SLICK was developed) that provide useful requirements for how SLICK could be used in assisting end users. In this section, we start with the results from several experiments performed with basic systems and show how SLICK could address users' needs. We then describe some of the user feedback we have received on the SLICK system.

We have analyzed the user experiences with a basic acquisition system SHAKEN. Four biologists have used SHAKEN to create biological process models, and two army generals have developed critiquing rules for military courses of action (COAs). We have performed an analysis on both 1) user entered knowledge and 2) information collected from detailed instrumentation of the system with respect to difficulties the user had. We interviewed end users and received feedback on missing capabilities that they would find useful (Pool et al., 2003; Barker et al., 2003).

Table 5 shows a summary of user desiderata and the SLICK capabilities that address them. Users often have difficulty in starting new knowledge acquisition tasks (F1). In order to help users, the system should proactively elicit the overall topic and goal and use them as a guide, pointing to what remains to be done. This corresponds to a couple of principles that SLICK uses (P1 and P2).

Table 5 goes here.

Users regularly reported that the systems did not tell the user clearly enough what the systems needed to know (F2). Users regularly queried the system to check whether the system was drawing the kinds of inference intended (F3). Users also had difficulty in checking whether they are making progress (F4). By explicitly maintaining what needs to be accomplished as intended outcomes and assessing the current status with respect to the outcomes, the system should help the user manage the acquisition tasks and achieve the intended results more effectively (P1, P6, P7, P11, and P12).

Users had difficulty in entering complex procedural knowledge into methods and sub-methods as the number of objects to keep track of increases (F5). The system should prioritize the tasks and help the user concentrate on a limited number of aspects at a time (P12). P13 is not currently supported.

Connecting or relating new knowledge to existing definitions, and checking consistency across different pieces of knowledge still seem difficult tasks (F6, F7). Although many existing systems perform error checking, in order to help the user more effectively, systems should relate the user entered knowledge to the existing definitions more extensively, and facilitate consistency checking across definitions (P3, P8, and P15).

Users wanted the knowledge captured to be more transparent to them by being able to see how they fit into and contribute to completing tasks (F8). SLICK provides means to access and examine knowledge that is being built (P6, P9, P11, and P14). Currently SLICK uses a simple approach for deciding when to provide the assessment results (P4). The user needs to set when he or she wants to see the feedback. A more flexible approach is left as a future work.

Finally, users wanted the system to be more proactive in providing guidance and suggesting what should be done and how to complete tasks (F9, F10). Although it is very hard for a system with limited knowledge in a domain to provide strong guidance on how and what should be done, a system can generate educated guesses on what could be most promising given what is known as described above (P5).

In summary, SLICK features meet the desiderata that were raised by end users of interactive knowledge acquisition systems.

We also have collected feedback on the SLICK system itself. Two Army officers provided informal comments on the system. These users interactively assessed the system by following a sequence of scripted steps in extending existing military plans and viewing the SLICK dialogue interface. The officers commented that SLICK's features are very helpful for understanding how they are making progress and for identifying remaining tasks. As described above, user evaluations with basic acquisition systems show that subjects often had difficulty in keeping track of how they are making progress. Explicitly presenting to users what they want to achieve or the expected effects, and relating this to the knowledge items being entered, was reported as a very useful feature.

7. Summary and Conclusion

This paper summarizes our investigation on exploiting tutoring and learning techniques in interactive knowledge acquisition systems. From the literature on human learning and on educational systems, we extracted fifteen principles that teachers and learners seem to follow in tutoring dialogues. These principles suggest valid strategies for interactive acquisition systems, as they learn from users that are teaching new knowledge.

The paper also presented an analysis of how some of these principles are used, implicitly, in existing systems for capturing procedural knowledge. The fifteen tutoring and learning principles we identified could be used more thoroughly in interactive knowledge acquisition systems by incorporating them into different functional aspects of the systems.

We also have presented a new approach for interactive knowledge capture that can be used to extend existing systems with acquisition goals, learning strategies, and awareness annotations over the current state of the knowledge base in terms of its completeness and competence. We described SLICK, a system that presents users with useful information regarding the progress made, the current status of the new knowledge, the goals that remain to be addressed, and suggested strategies to accomplish those goals. This external record of the teacher/student interaction can help users visualize where the lesson is at, and relieve them of a significant burden during the acquisition process. It also helps users by pointing out what remains to be done and with suggestions of next steps to pursue.

The fifteen tutoring and learning principles used in this work could be augmented with additional research in the vast literature of education and learning. For example, while our focus so far has been on typical student and teacher interactions, it may be useful to extract principles that brighter students use so that interactive acquisition systems could be not just good students, but exceptionally bright students. Since we do not expect users to be trained in teaching or tutoring techniques, it would be useful for systems to be supplemented with knowledge about teaching that typical students are not expected to have.

We believe that the research 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 has been recent interest 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 intelligent tutoring systems.

8. 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 the members of the SHAKEN project for their feedback on this work. We also 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. P., 1968. Educational psychology: A cognitive approach, Holt, Rinehart and Winston, New York.

Bareiss, R., Porter, B., Holte, R., 1990. Concept Learning and Heuristic Classification in Weak-Theory Domains. Artificial Intelligence, 45, 229-264.

Barker, K., Blythe, J., Borchardt, G., Chaudhri, V., Clark, P., Cohen, P. R., Fitzgerald, J., Forbus, K., Gil, Y., Katz, B., Kim, J., King, G., Mishra, S., Murray, K., Otstott, C., Porter, B., Schrag, R., Uribe, T., Usher, J., Yeh, P., 2003. A Knowledge Acquisition Tool for Course of Action Analysis, in Proceedings of the Innovative Applications of Artificial Intelligence Conference, 43-50.

Bartlett, F. C., 1958. Thinking, Basic Books, New York.

Basili, V., Selby, R. W., Hutchens, D. H., 1986. Experimentation in Software Engineering. IEEE Transactions in Software Engineering, SE-12.

Blythe, J., Kim, J., Ramachandran, S., Gil, Y., 2001. An Integrated Environment for Knowledge Acquisition, in Proceedings of the International Conference on Intelligent User Interfaces, 13-20.

Boose, J. H., 1989. A survey of knowledge acquisition techniques and tools. Knowledge Acquisition, 1 (1), 3-37.

Bransford, J. D., Brown, A. L., Cocking, R. R., 1999. How People Learn: Brain, Mind, Experience, and School.

Bredeweg, B., Forbus, K., 2004. Qualitative modeling in education. AI Magazine, 24, 35-46.

Breuker, J., Winkels, R., Sandberg, J., 1987. A Shell for Intelligent Help Systems., in International Joint Conferences on Artificial Intelligence, 167-173.

Breuker, J. A., 1990. EUROHELP: Developing Intelligent Help Systems, Copenhagen.

Brown, J. S., Burton, R., Kleer, J. D., 1982. Pedagogical Natural Language and Knowledge Engineering Techniques in SOPHIE I, II, III. in: SLEEMAN, D., BROWN, J. S. (Eds.) Intelligent Tutoring Systems, Academic Press, New York.

Brown, J. S., Burton, R. R., 1978. Diagnostic models for procedural bugs in basic mathematical skills. Cognitive Science, 2, 155-191.

Burton, R. R., Brown, J. S., 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, 190-202.

Chi, M., 2000. Self-Explaining Expository Texts: The Dual Processes of Generating Inferences and Repairing Mental Models. in: GLASER, R. (Ed.) Advances in Instructional Psychology, Mahwah, NJ, Lawrence Erlbaum, 161-238.

Clancey, W. J., 1987. Knowledge-Based Tutoring:The GUIDON Program, MIT press.

Clark, H. H., Haviland, S. E., 1977. Comprehension and the given-new contract. in: FREEDLE, R. O. (Ed.) Discourse production and comprehension, Norwood, N.J.: Ablex, 1-40.

Clark, P., Porter, B., 2006. The knowledge machine.

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 the International Conference on Knowledge Capture, 22-29.

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.

Cypher, A., 1993. Watch what I do: Programming by demonstration, MIT press.

Davis, R., 1979. Interactive Transfer of Expertise: Acquisition of New Inference Rules. Artificial Intelligence, 121-157.

De Koning, K., Breuker, J., Wielinga, B., Bredeweg, B., 2000. Model-based reasoning about learner behaviour. Artificial Intelligence, 117 (2), 173-229.

Denny, M., 2002. Ontology Building: A Survey of Editing Tools.

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., 2001. Smart Machines in Education, AAAI press.

Forbus, K., Usher, J., Chapman, V., 2003. Sketching for military courses of action diagrams, in Proceedings of the International Conference on Intelligent User Interfaces, 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, 49-85.

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

Gibbons, A. S., 2001. Model-Centered Instruction. Journal of Structural Learning and Intelligent Systems, 14, 51-540.

Gibson, J. J., 1966. The Senses Considered as Perceptual Systems, Houghton Mifflin, Boston.

Gil, Y., Kim, J., 2002. Interactive Knowledge Acquisition Tools: A Tutoring Perspective, in Proceedings of the 24th Annual Meeting of the Cognitive Science Society, 357-362.

Gil, Y., Melz, E., 1996. Explicit representations of problem-solving strategies to support knowledge acquisition, in Proceedings of the National Conference on Artificial Intelligence, 469-476.

Ginsberg, A., Weiss, S., Politakis, P., 1985. SEEK2: A Generalized Approach to Automatic Knowledge Base Refinement, in Proceedings of the Ninth International Joint Conference on Artificial Intelligence, 367-374.

Harmelen, F. V., Wielinga, B., Bredeweg, B., Karbach, G. S. W., Reinders, M., Voss, A., Akkermans, H., Bartsch-Spoerl, B., Vinkhuyzen, E., 1992. Knowledge-Level Reflection. Enhancing the Knowledge Engineering Process -- Contributions from ESPRIT, Elsevier Science, 175-204.

Huffman, S., Laird, J., 1995. Flexibly Instructable Agents. Journal of Artificial Intelligence Research, 3, 271-324.

Kearsley, G., 2005. Explorations in Learning & Instruction: The Theory Into Practice Database (TIP), http://www.gwu.edu/~tip/.

Kim, J., Blythe, J., 2003. Supporting Plan Authoring and Analysis, in Proceedings of the International Conference on Intelligent User Interfaces, 109-116.

Kim, J., Gil, Y., 1999. Deriving Expectations to Guide Knowledge Base Creation, in Proceedings of the National Conference on Artificial Intelligence, 235-241.

Kim, J., Gil, Y., 2000. Acquiring Problem-Solving Knowledge from End Users: Putting Interdependency Models to the Test, in Proceedings of the National Conference on Artificial Intelligence, 223-229.

Kim, J., Gil, Y., 2001. Knowledge Analysis on Process Models, in Proceedings of the International Joint Conference on Artificial Intelligence, 935-942.

Kim, J., Gil, Y., 2002. Deriving Acquisition Principles from Tutoring Principles, in Proceedings of the Intelligent Tutoring Systems Conference, 661-670.

Kim, J., Gil, Y., 2003. Proactive Acquisition from Tutoring and Learning Principles, in Proceedings of the Artificial Intelligence in Education, 175-182.

Koedinger, K. R., Anderson, J. R., Hadley, W. H., Mark, M. A., 1997. Intelligent tutoring goes to school in the big city. International Journal of Artificial Intelligence in Education, 8, 30-43.

Kulik, J. A., Kulik, C., 1988. Timing of feedback and verbal learning. Review of Educational Research, 58, 79-97.

Lefrancois, G. R., 1991. Psychology For Teaching., Wadsworth, Belmont.

Lepper, M. R., Woolverton, M., Mumme, D. L., Gurtner, J., 1993. Motivational Techniques of Expert Human Tutors: Lesson for the Design of Computer-Based Tutors. Computers as Cognitive Tools, Hillsdale, 75-105.

Maes, P., Nardi, D., 1988. Meta-Level Architectures and Reflection, Elsevier Science, NY.

Marcus, S., Mcdermott, J., 1989. SALT: A knowledge acquisition language for propose-and-revise systems. Artificial Intelligence, 39, 1-37.

Marton, F., Hounsell, D., Entwistle, N., 1984. The Experience of Learning., Scottish Academic Press, Edinburgh.

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 the Seventh International Conference on Principles of Knowledge Representation and Reasoning, 483–493.

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.

Merrill, J., 1987. Levels of questioning and forms of feedback: Instructional factors in courseware design. Journal of Computer-Based Instruction, 14, 18-22.

Mitchell, T., Mahadevan, S., Steinberg, L., 1985. LEAP: A learning apprentice for VLSI design., in Proceedings of the International Joint Conference on Artificial Intelligence, 573–580.

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.

O'shea, T., 1979. A self-improving Quadratic tutor. International Journal of Man-Machine Studies, 11, 97-124.

Olson, G., Moran, T., 1998. Special issue on experimental comparisons of usability evaluation methods. Human-Computer Interaction, 13(3), 203-262.

Pask, G., 1975. Conversation, Cognition, and Learning, Elsevier, New York.

Piaget, J., Inhelder, B., 1973. Memory and intelligence, Basic Books.

Pool, M., Murray, K., Fitzgerald, J., Mehrotra, M., Schrag, R., Blythe, J., Kim, J., Chalupsky, H., Miraglia, P., Russ, T., D.Schneider, 2003. Evaluating SME-Authored COA Critiquing Knowledge, in Proceedings of International Conference on Knowledge Capture, 96-104.

Puerta, A. R., Egar, J. W., Tu, S. W., Musen, M. A., 1992. A multiple-method knowledge-acquisition shell for the automatic generation of knowledge-acquisition tools. Knowledge Acquisition, 4, 171-196.

Rose, C. P., Jordan, P., Ringenberg, M., Siler, S., Vanlehn, K., Weinstein, A., 2001. Interactive Conceptual Tutoring in Atlas-Andes, in Proceedings of Artificial Intelligence in Education, 256-266.

Scandura, J. M., 2004. Structural Learning Theory: Current Status and New Perspectives, http://www.scandura.com/Articles/SLT%20Status-Perspectives.PDF.

Schreiber, G., Akkermans, H., Anjewierden, A., Hoog, R. D., Shadbolt, N., Velde, W. V. D., Wielinga, B., 1999. Knowledge Engineering and Management The CommonKADS Methodology, MIT Press.

Self, J., 1993. Special issue on evaluation. Journal of Articial Intelligence in Education, 4, 129-413.

Shadbolt, N., Burton, A. M., 1989. The empirical study of knowledge elicitation techniques. SIGART Newsletter, 108, 15-18.

Shadbolt, N., Motta, E., Rouge, A., 1993. Constructing Knowledge-Based Systems. IEEE Software, 10, 34 - 38.

Shadbolt, N., O'hara, K., Crow, L., 1999. The experimental evaluation of knowledge acquisition techniques and methods: history, problems and new directions. Int. Journal of Human-Computer Studies, 51(4), 729-755.

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, Springer, 483-510.

Stevens, A. L., Collins, A., 1977. The goal structure of a Socratic tutor, in Proceedings of the National ACM Conference, 256-263.

Sure, Y., Staab, S., Studer, R., 2002. Methodology for development and employment of ontology based knowledge management applications. ACM SIGMOD Record, 31.

Sweller, J., 1988. Cognitive load during problem solving: Effects on learning. Cognitive Science, 12, 257-285.

Tallis, M., Kim, J., Gil, Y., 2001. User Studies Knowledge Acquisition Tools: Methodology and Lessons Learned. Journal of Experimental and Theoretical Artificial Intelligence, 13 (4), 359-378.

Tecuci, G., Boicu, M., Boicu, D., Bowlam, M., Ciucu, F., Levcovici, C., 2000. Rapid Development of a High Performance Knowledge Base for Course of Action Critiquing, in Proceedings of the National Conference on Artificial Intelligence, 1046-1053.

Vanlehn, K., Freedman, R., Pamela, J., Murray, C., And, R. O., 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 the International Conference on Intelligent Tutoring Systems, 474-483.

Wenger, E., 1987. Artificial Intelligence and Tutoring Systems, Morgan Kaufmann.

Wielinga, B. J., Schreiber, A. T., Breuker, A., 1992. KADS: A modelling approach to knowledge acquisition. Knowledge Acquisition, 4, 5–54.

Witbrock, M. J., Baxter, D., Curtis, J., Schneider, D., Kahlert, R., Miraglia, P., Wagner, P., Panton, K., Matthews, G., Vizedom, A., 2003. An Interactive Dialogue System for Knowledge Acquisition in Cyc, in Proceedings of the IJCAI-03 Workshop on Mixed-Initiative Intelligent Systems.

Woolf, B. P., 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, 61-73.

Teaching/Learning principle	Tutoring literature
P1: Start by introducing topics and goals	Atlas-Andes, Meno-Tutor, Human tutorial dialog
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: Indicate lack of understanding	Human tutorial dialog, WHY
P7: Keep on track	GUIDON, SHOLAR, TRAIN-Tutor
P8: Detect and fix “buggy” knowledge	SCHOLAR, Meno-Tutor, WHY, Buggy, CIRCSIM
P9: Learn deep model	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 lessons	Atlas
P14: Summarize what was learned	EXCHECK, TRAIN-Tutor, Meno-Tutor
P15: Provide overall assessment of learning knowledge	WEST, Human tutorial dialog

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

Acquisition Tool	*Highlights*
EXPECT (Blythe et al, 2001)	To acquire problem solving knowledge. Exploits dialogue scripts, knowledge interdependency models, and background knowledge.
INSTRUCTO-SOAR (Huffman and Laird, 1995)	To acquire task models for Soar
KSSn (Gaines and Shaw, 1993)	To acquire concepts, rules, and data. Based on personal construct psychology.
PROTEGE-II (Puerta, 1992)	To acquire domain-specific ontologies. Ontologies are tied to problem-solving strategies.
PROTOS (Bareiss et al., 1990)	Users specify cases, tool explains their classification.
SALT (Marcus and 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 Tools for Acquiring Procedural Knowledge

Tutoring/Learning principle	Acquisition Function				
	Assimilate Instruction	Trigger Goals	Propose Strategies	Prioritize Goals & Strategies	Design Presentation
p1: Introduce topics & goals		EXPECT SEEK2	PROTÉGÉ-II		
P2: Use topics of the lesson as a guide	SALT	SEEK2	EXPECT	SALT	SALT
P3: Subsumption to existing cog. structure	PROTOS PROTÉGÉ-II	PROTÉGÉ-II	TEIREISIAS		PROTOS,SALT PROTÉGÉ-II
P4: Immediate feedback	PROTOS PROTÉGÉ-II	INSTRUCTO-SOAR	TEIREISIAS		EXPECT PROTÉGÉ-II
P5: Generate educated guesses		TEIREISIAS	EXPECT		
P6: Indicate lack of understanding	INSTRUCTO-SOAR				INSTRUCTO-SOAR
P7: Keep on track					
P8: Detect and fix "buggy" knowledge	TAQL	EXPECT			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 lessons					
P14: Summarize what is learned					
P15: Assess learned knowledge		KSSn			

Table 3. Tutoring and learning principles used in interactive acquisition tools.

SLICK function	Supporting calls to the backend acquisition system and the knowledge base
Check purpose of the lesson	Find the purpose/topics of a lesson explain how the answer is generated for a test or a query
Check k-item definition	Check whether an over-general k-item is used in a k-item definition, find required roles for a k-item, check whether an enumeration is complete in a k-item definition, find inconsistent axioms for a k-item, generate tests for a k-item, check whether test results change due to k-item modifications.
Check related k-items	Find inconsistent k-items for a given k-item, check whether two k-items are equivalent, find connections to other k-items for a k-item.
Detect errors and find fixes	Find errors in a definition, find missing k-items or axioms, find system suggestions on a detected error, check whether errors are equivalent.
Basic KB calls	Find role values for a k-item, check whether a k-item is more general than another k-item.

Table 4. Example SLICK's calls to backend knowledge acquisition system and the knowledge base

Difficulties and requirements that users had in entering complex procedural knowledge	Supporting SLICK capability/principles
F1: *Users have difficulty in starting new tasks*	P1, P2 (Get the overall topic in the beginning and use it as a guide, system points where to start based on remaining tasks)
F2: *System did not tell the user clearly enough what the system needs to know.*	P7, P15, P6 (User sees remaining tasks and what should be done)
F3: *Users regularly queried the system to determine whether the system was drawing the kinds of inference intended.*	P11, P6 (Expected effects are extracted from user intent and tested for validation)
F4: *Need help manage work in progress and remove unwanted work, keep on track*	P1, P6, P12 (User can view progress over time and prioritized tasks)
F5: *Users have difficulty in entering complex procedural knowledge with nested methods and sub-methods*	P12 (Help users prioritize tasks)
F6: *Connect authored knowledge to prior knowledge*	P3 (relate new knowledge with existing definitions and make them consistent with each other)
F7: *Facilitate consistency checking within the user work and across user efforts*	P8, P15 (User sees inconsistent definition as the reason of remaining tasks)
F8: *Making knowledge captured more transparent*	P9 (User can see what system is understanding about new knowledge and provide feedback).
F9: *Providing dialogue tools to help ensure completeness*	P11, P6, P14 (User sees remaining tasks and progress and a summary of knowledge built)
F10: *Providing proactive help*	P5 (Generate educated guesses and make suggestions on how to make progress)

Table 5. SLICK capabilities for addressing user needs.

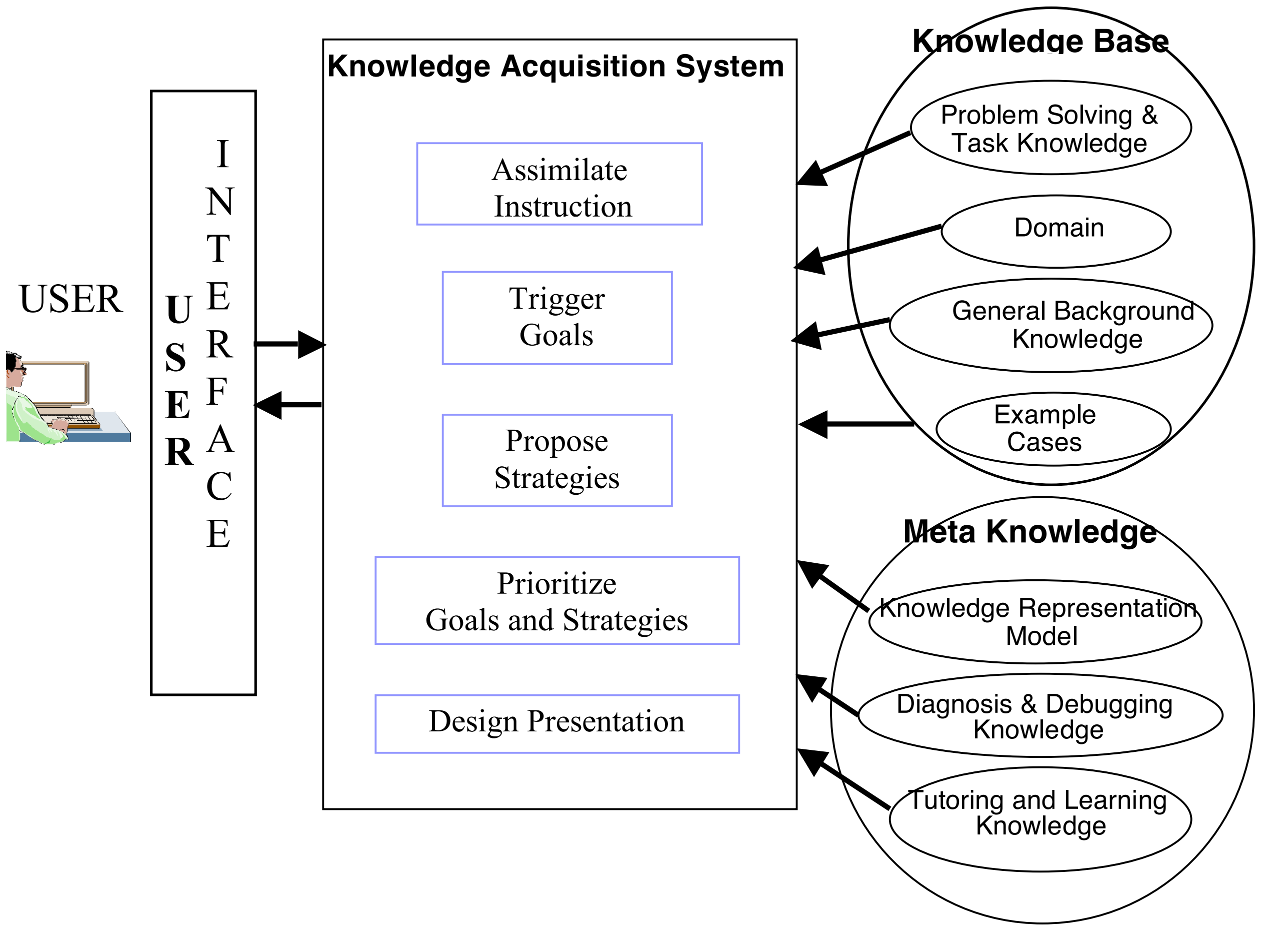

Knowledge Acquisition System
Assimilate Instruction
Trigger Goals
Propose Strategies
Prioritize Goals and Strategies
Design Presentation
USER
USER INTERFACE
Knowledge Base
Problem Solving & Task Knowledge
Domain
General Background Knowledge
Example Cases
Meta Knowledge
Knowledge Representation Model
Diagnosis & Debugging Knowledge
Tutoring and Learning Knowledge

Figure 1: A Modular View of Acquisition Strategies and Knowledge Sources Typical of Interactive Knowledge Acquisition Systems.

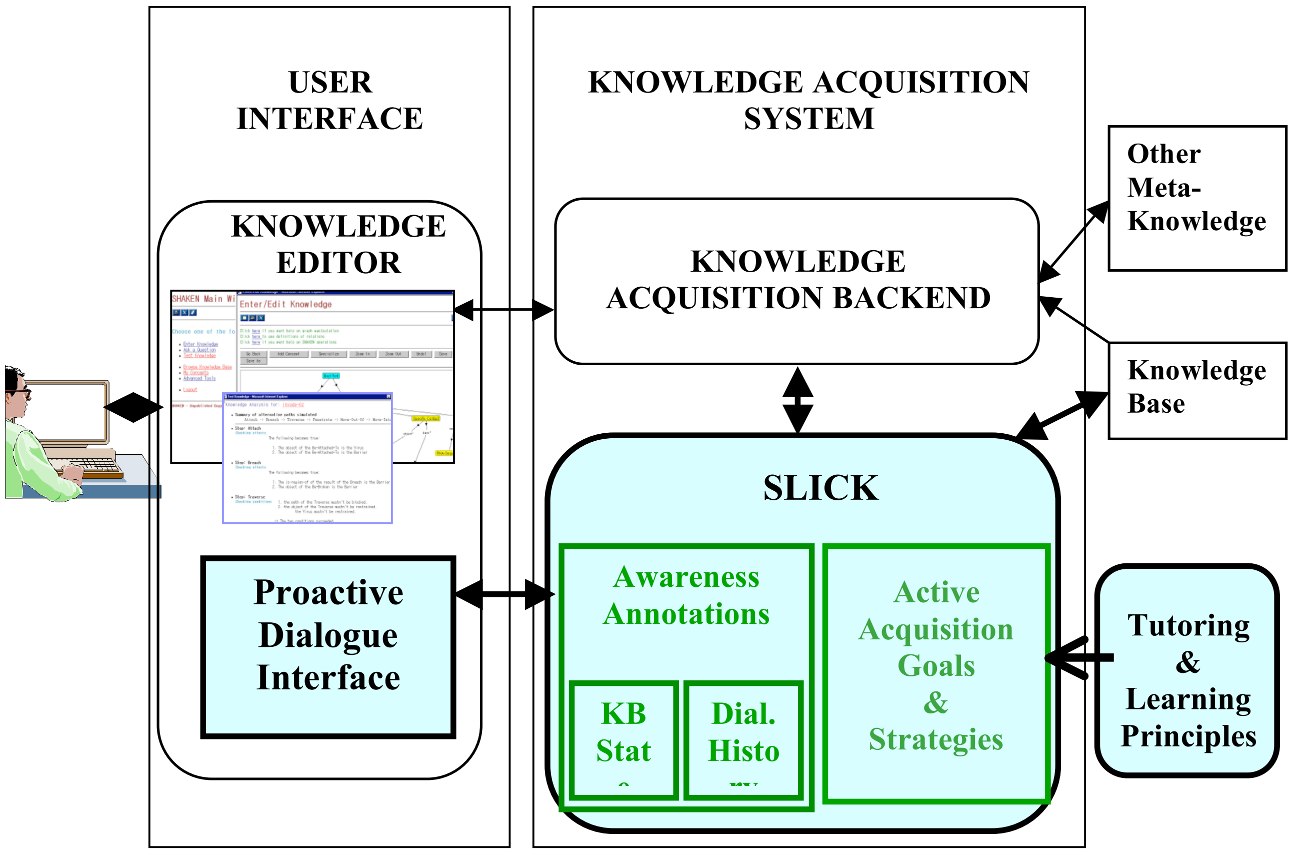

USER INTERFACE
KNOWLEDGE ACQUISITION SYSTEM
KNOWLEDGE EDITOR
KNOWLEDGE ACQUISITION BACKEND
Other Meta-Knowledge
Knowledge Base
SLICK
Awareness Annotations
Active Acquisition Goals & Strategies
KB Stat
Dial. Histo
Proactive Dialogue Interface
Tutoring & Learning Principles

Figure 2. The SLICK architecture

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 as 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, P6, P5)

- **3.1. Test the new body of knowledge and generate tests for the aspects that have not been thoroughly tested.**
- **3.2. Detect missing knowledge and fix problems that result from self-checks or from user's indications.**
- **3.3. Ensure user checks the reason for the answers, not just the answers.**
- **3.4. Confirm new answers that change in light of new knowledge.**
- **3.5. Generate educated guesses for the problems found.**

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) Six Main Category of 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**

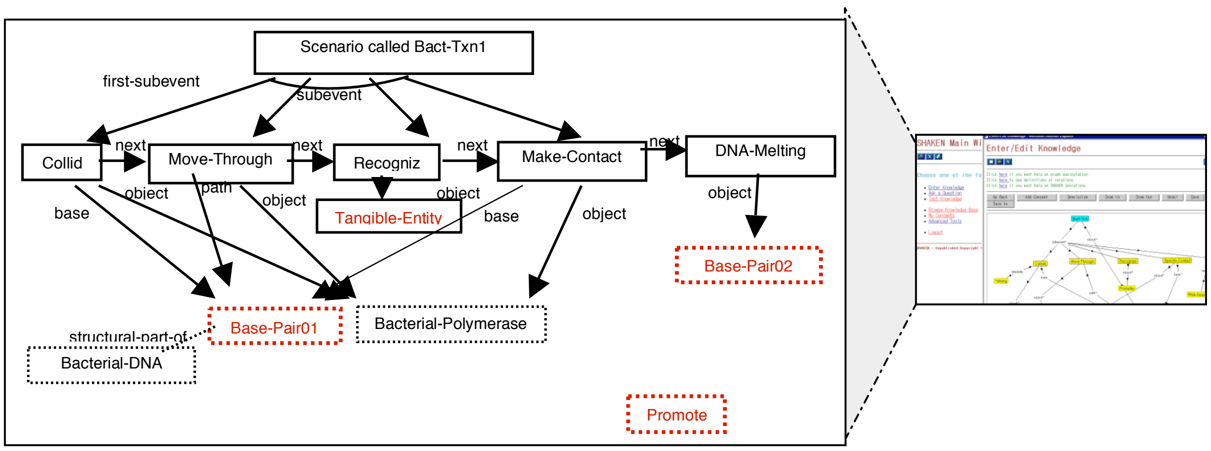
(b) Awareness Annotations

Figure 3. Acquisition Goals and Awareness Annotations in SLICK

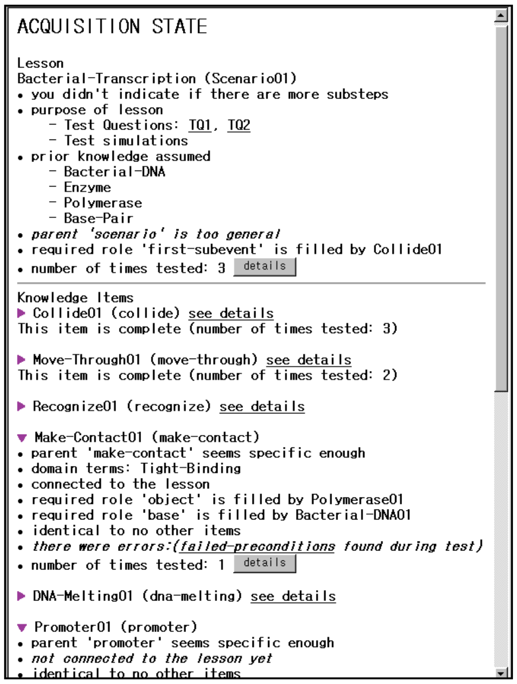
```
Update_Acquisition_Goals_&_Awareness_Annotations_and_Send_Suggestions

Input: Knowledge acquisition action(s), current_lesson, prior annotations of the lesson if exists
Output: changes in awareness annotations and acquisition goals, and suggestions
1. Activate acquisition goals for the knowledge acquisition action(s) and create awareness annotations with
        respect to the goals and strategies (P15, P6)
  1.1. Check the purpose of the lesson and whether the status has been changed. (P1,P2,P11,P9)
    - check whether the purpose or topics are given for the lesson and whether the status has been changed.
    - check whether all the topics are covered and whether the status has been changed.
    - for a test or a query, ensure that the user checks the reason for the answer and check whether the
        status has been changed.
  1.2. For each knowledge item, perform the following. (P3, P7)
    - check whether an overgeneral concept is used in the definition and whether the status has
        been changed.
    - check whether the required slots of the item are all specified and whether the status has
        been changed.
    - check the number of times used in tests and whether the status has been improved.
    - check whether the item is connected to the lesson and whether the status has been changed.
    - check whether there are other items defined with identical k-items and slots and whether the
        status has been changed.
    - for any modification of the k-item, check whether test results change.
    - for each enumeration of items, check whether the enumeration is complete and whether the
        status has been changed.
  1.3. For each axiom of a k-item, perform the following (P8, P15)
    - check whether any values chosen are overgeneral and whether the status has been changed.
    - check the number of times the axiom is used in tests and whether the status has been improved.
    - check whether there are any inconsistent axioms and whether the status has been changed.
2. Report achieved/unachieved learning goals and generate suggestions using acquisition strategies
        (P14,P12)
  2.1. Report the goals that are now achieved.(P15,P6,P11)
  2.2. Prioritize problems (unachieved acquisition goals) and generate educated guesses for them.
        (P5,P7,P12)
       2.2.1. Report unachieved goals and suggestions for them.
         - replace overgeneral concepts with more specific definitions.
         - relate unconnected items to the main topic.
2.2.2. Report repeated problems and generate educated guesses for them.
       2.2.3. Report less confident k-items and axioms, and generate educated guesses for them.
```

Figure 4. SLICK steps for activating acquisition goals, creating awareness annotations with respect to the goals and strategies, and sending suggestions to the user



(a) A process model of Bacterial Transcription entered by a user



ACQUISITION STATE

Lesson
Bacterial-Transcription (Scenario01)
- you didn't indicate if there are more substeps
- purpose of lesson
  - Test Questions: TQ1, TQ2
  - Test simulations
- prior knowledge assumed
  - Bacterial-DNA
  - Enzyme
  - Polymerase
  - Base-Pair
- *parent 'scenario' is too general*
- required role 'first-subevent' is filled by Collide01
- number of times tested: 3 details

Knowledge Items
▶ Collide01 (collide) see details
This item is complete (number of times tested: 3)

▶ Move-Through01 (move-through) see details
This item is complete (number of times tested: 2)

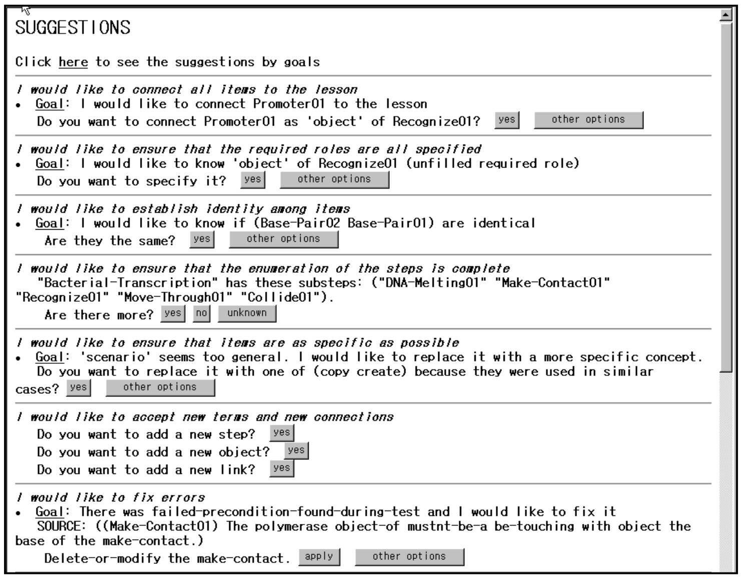
▶ Recognize01 (recognize) see details

▼ Make-Contact01 (make-contact)
- parent 'make-contact' seems specific enough
- domain terms: Tight-Binding
- connected to the lesson
- required role 'object' is filled by Polymerase01
- required role 'base' is filled by Bacterial-DNA01
- identical to no other items
- *there were errors:(failed-preconditions found during test)*
- number of times tested: 1 details

▶ DNA-Melting01 (dna-melting) see details

▼ Promoter01 (promoter)
- parent 'promoter' seems specific enough
- *not connected to the lesson yet*
- identical to no other items

(b) SLICK annotations to KB



SUGGESTIONS

Click here to see the suggestions by goals

I would like to connect all items to the lesson
- Goal: I would like to connect Promoter01 to the lesson
  Do you want to connect Promoter01 as 'object' of Recognize01? yes other options

I would like to ensure that the required roles are all specified
- Goal: I would like to know 'object' of Recognize01 (unfilled required role)
  Do you want to specify it? yes other options

I would like to establish identity among items
- Goal: I would like to know if (Base-Pair02 Base-Pair01) are identical
  Are they the same? yes other options

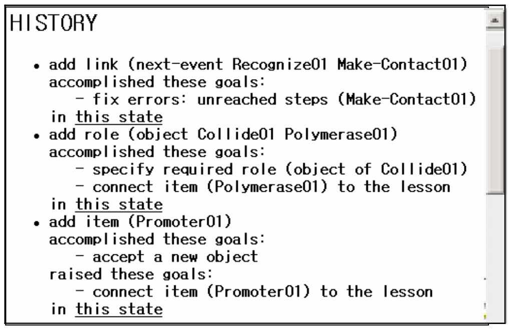
I would like to ensure that the enumeration of the steps is complete
"Bacterial-Transcription" has these substeps: ("DNA-Melting01" "Make-Contact01" "Recognize01" "Move-Through01" "Collide01").
Are there more? yes no unknown

I would like to ensure that items are as specific as possible
- Goal: 'scenario' seems too general. I would like to replace it with a more specific concept.
  Do you want to replace it with one of (copy create) because they were used in similar cases? yes other options

I would like to accept new terms and new connections
Do you want to add a new step? yes
Do you want to add a new object? yes
Do you want to add a new link? yes

I would like to fix errors
- Goal: There was failed-precondition-found-during-test and I would like to fix it
  SOURCE: ((Make-Contact01) The polymerase object-of mustnt-be-a be-touching with object the base of the make-contact.)
  Delete-or-modify the make-contact. apply other options

(c) Suggestions are generated from learning goals



HISTORY

- add link (next-event Recognize01 Make-Contact01)
  accomplished these goals:
  - fix errors: unreached steps (Make-Contact01)

  in this state
- add role (object Collide01 Polymerase01)
  accomplished these goals:
  - specify required role (object of Collide01)
  - connect item (Polymerase01) to the lesson

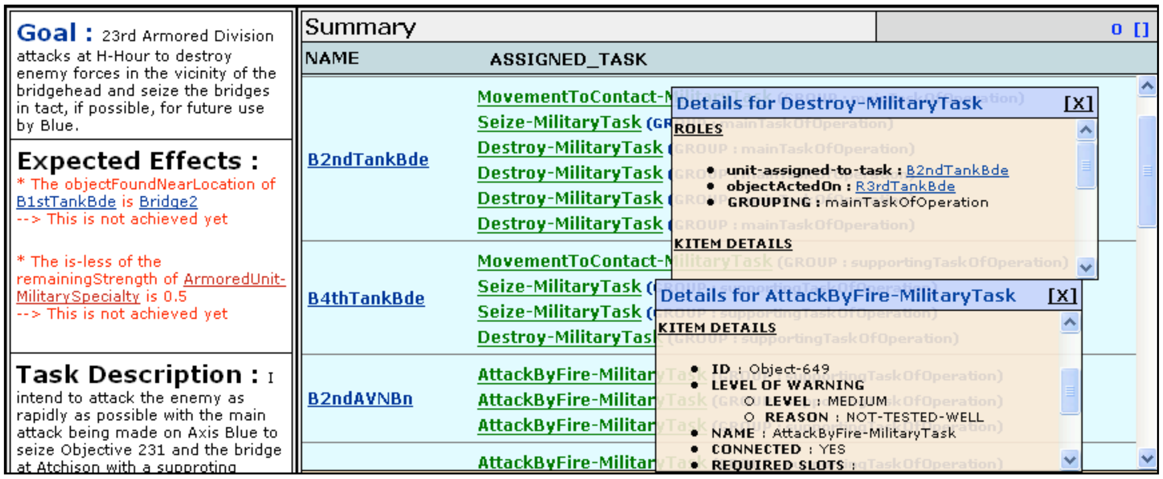
  in this state
- add item (Promoter01)
  accomplished these goals:
  - accept a new object

  raised these goals:
  - connect item (Promoter01) to the lesson

  in this state

(d) SLICK annotations to history

Figure 5: Proactive acquisition of biological process models.



Goal : 23rd Armored Division attacks at H-Hour to destroy enemy forces in the vicinity of the bridgehead and seize the bridges in tact, if possible, for future use by Blue.

Expected Effects :

* The objectFoundNearLocation of B1stTankBde is Bridge2
--> This is not achieved yet

* The is-less of the remainingStrength of ArmoredUnit-MilitarySpecialty is 0.5
--> This is not achieved yet

Task Description : I intend to attack the enemy as rapidly as possible with the main attack being made on Axis Blue to seize Objective 231 and the bridge at Atchison with a supproting

Summary

NAME	ASSIGNED_TASK
B2ndTankBde	MovementToContact-MilitaryTask (GROUP : mainTaskOfOperation)
	Seize-MilitaryTask (GROUP : mainTaskOfOperation)
	Destroy-MilitaryTask (GROUP : mainTaskOfOperation)
	Destroy-MilitaryTask (GROUP
	Destroy-MilitaryTask (GROUP
	Destroy-MilitaryTask (GROUP : mainTaskOfOperation)
B4thTankBde	MovementToContact-MilitaryTask (GROUP : supportingTaskOfOperation)
	Seize-MilitaryTask (GROUP : supportingTaskOfOperation)
	Seize-MilitaryTask (GROUP : supportingTaskOfOperation)
	Destroy-MilitaryTask (GROUP : supportingTaskOfOperation)
B2ndAVNBn	AttackByFire-MilitaryTask (GROUP : supportingTaskOfOperation)
	AttackByFire-MilitaryTask (GROUP : supportingTaskOfOperation)
	AttackByFire-MilitaryTask (GROUP : supportingTaskOfOperation)
	AttackByFire-MilitaryTask (GROUP : supportingTaskOfOperation)

Details for Destroy-MilitaryTask [X]

ROLES

- unit-assigned-to-task : B2ndTankBde
- objectActedOn : R3rdTankBde
- GROUPING : mainTaskOfOperation

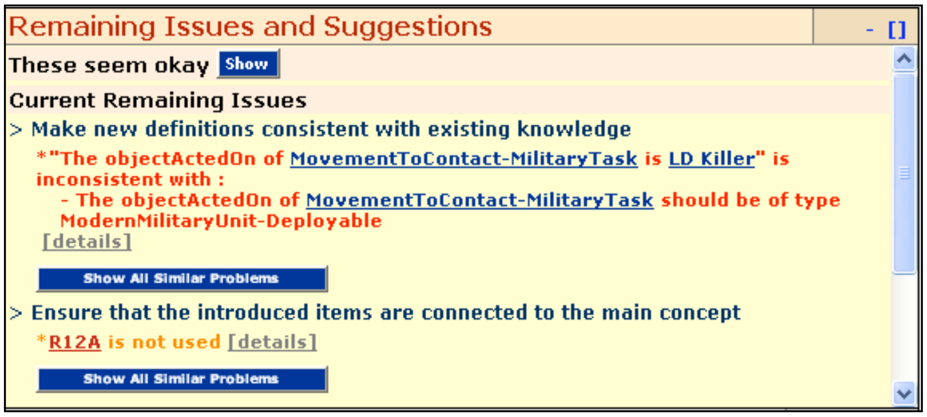
KITEM DETAILS

Details for AttackByFire-MilitaryTask [X]

KITEM DETAILS

- ID : Object-649
- LEVEL OF WARNING
  - LEVEL : MEDIUM
  - REASON : NOT-TESTED-WELL
- NAME : AttackByFire-MilitaryTask
- CONNECTED : YES
- REQUIRED SLOTS :

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



Remaining Issues and Suggestions

These seem okay Show

Current Remaining Issues

> Make new definitions consistent with existing knowledge

*"The objectActedOn of MovementToContact-MilitaryTask is LD Killer" is inconsistent with :
- The objectActedOn of MovementToContact-MilitaryTask should be of type ModernMilitaryUnit-Deployable

[details]

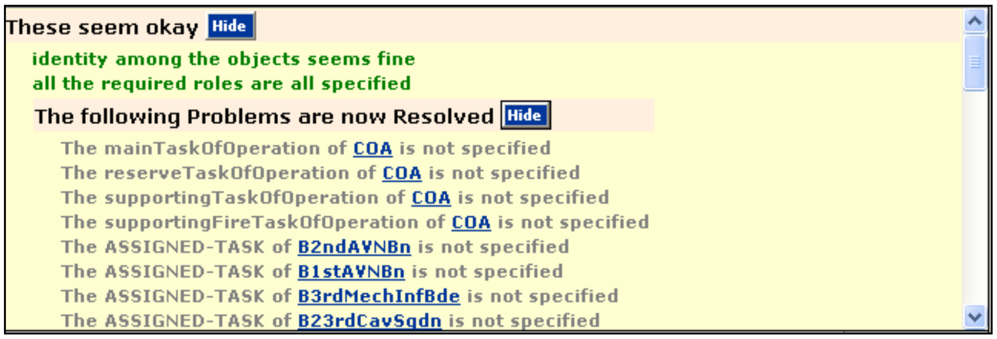
Show All Similar Problems

> Ensure that the introduced items are connected to the main concept

*R12A is not used [details]

Show All Similar Problems

(b) Showing remaining issues



These seem okay Hide

identity among the objects seems fine
all the required roles are all specified

The following Problems are now Resolved Hide

The mainTaskOfOperation of COA is not specified
The reserveTaskOfOperation of COA is not specified
The supportingTaskOfOperation of COA is not specified
The supportingFireTaskOfOperation of COA is not specified
The ASSIGNED-TASK of B2ndAVNBn is not specified
The ASSIGNED-TASK of B1stAVNBn is not specified
The ASSIGNED-TASK of B3rdMechInfBde is not specified
The ASSIGNED-TASK of B23rdCavSqdn is not specified

(c) Showing progress over time

Figure 6. Proactive acquisition of military plans.