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, U.S.A.
{jihie, gil}@isi.edu

Abstract. This paper argues that interactive knowledge acquisition tools would benefit from a tighter and more thorough incorporation of tutoring and learning principles. Current systems learn from users in a very passive and disengaged manner, and could be designed to incorporate the proactive capabilities that one expects of a good student. This paper describes our analysis of the literature on tutorial dialogues presents a compilation of useful principles that students and teachers typically follow in making tutoring interactions successful. We then point out what tutoring and learning principles have been used to date in the acquisition literature, though unintentionally and implicitly. We discuss how a more thorough and explicit representation of these principles would help improve enormously how computers learn from users. We plan to use these ideas in our future work to develop more proactive and effective acquisition interfaces.

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

1 Introduction

Transferring knowledge from humans to computers has proven to be an extremely challenging task. Over the last two decades, an array of approaches to interactive knowledge acquisition have been proposed. Some tools accept rules and check them against other existing rules [15, 24]. Some tools acquire knowledge suitable for specific tasks and problem solving strategies [30]. Other tools focus on detecting errors in the knowledge specified by the user [22, 25, 32]. Some systems use a variety of elicitation techniques to acquire descriptive knowledge [20, 36] often in semi-formal forms. There are some isolated reports of users with no formal background in computer science that are now able to use acquisition tools to build sizeable knowledge bases [26, 16, 11]. However, the majority of the burden of the acquisition task still remains with the user. Users have to decide what, how, and when to teach the system. Current acquisition tools do not take the kind of initiative and collaborative attitude that one would expect of a good student, mostly reacting to the user's actions instead of being proactive learners.

We set off to investigate how the dynamics of tutor-student interactions could be used to make acquisition tools better students to further support users in their role of tutors of computers. Given the success in deploying educational systems in schools and their reported effectiveness in raising student grades [27], 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 [19], which the knowledge acquisition literature lacks.

The contributions of this work are twofold. First we present our analysis of the literature on tutorial dialogues and provide a compilation of useful principles that students and teachers follow in making tutoring interactions successful and that could be useful in the context of interactive acquisition tools. Second, we point out how existing knowledge acquisition tools use techniques that are related to widely used tutoring and learning principles. Based on these analyses we identify areas that the acquisition tools developed to date have neglected, and suggest promising areas of research based on our findings.

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

2 Tutorial Dialogues in Instructional Systems and in Interactive Knowledge Acquisition

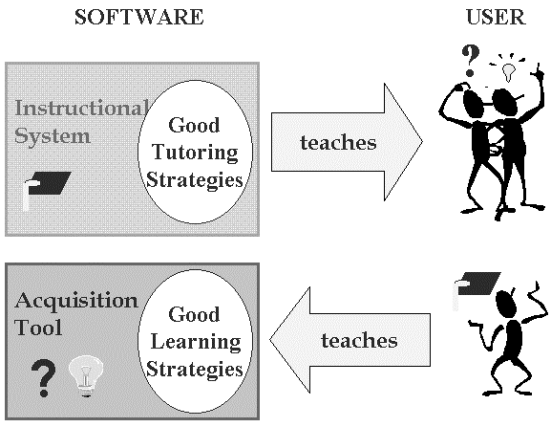

Fig. 1. Comparing Tutoring Systems and Acquisition Systems

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

In essence, we are trying to investigate what it takes to create a good student, while most ITS work has focused on creating good teachers. We believe that the work in educational systems and acquisition systems share a lot of issues and they may be able to contribute to each other in many ways. In fact there has been work that bridges these two communities. For example, there have been recent interests in acquiring knowledge for intelligent tutoring systems [35]. We think that technology built by the knowledge acquisition community will be useful for building tools to help users develop the knowledge and models used in ITS.

There are some issues that interactive acquisition interfaces will not face. Human students in need of tutoring often have a lack of motivation that the

instructional system has to address [29]. Instructional systems need to use special tactics to promote deep learning, such as giving incremental hints instead of showing the student the correct answers. Finally, our student will not be subject to the cognitive limitations of a typical human student, and can exploit memory and computational skills that would be exceptional (if not infrequent) for human students.

3 Teaching and Learning Principles

We have been investigating various tutoring principles[1] used by human tutors and educational software [18, 42, 19]. Although human tutors provide more flexible support, the tutoring principles supported by educational software are often inspired by human tutors [34] and we derive learning principles from both. Table 1 shows a summary of the principles that we found useful. The rest of this section describes these principles.

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

1. **Introduce lesson topics and goals**
   In the beginning of the lesson, tutors often outline the topics to be learned during the session and try to assess the student's prior knowledge on these topics. For example, the advance organizer approach [3] 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 [44], 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 [19].
2. **Use topics of the lesson as a guide**
   It is useful for students and tutors to ensure that what is being learned has some connection or relevance to the topics of the lesson. In planning tutorial dialogues, instructional systems check what is being learned against the topics of the lesson [13] and try to avoid unfocused dialogue and digressions. In the process of learning, the terms brought up during the lesson are connected to the concepts learned [39].
3. **Subsumption to existing cognitive structure**
   The subsumption theory by Ausubel [3] emphasizes that learning new material involves relating it to relevant ideas in the existing cognitive structure.

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

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

References: Atlas [41], Atlas-Andes[38], BE&E [13], Buggy [7], CIRCSIM-tutor [46], EXCHECK [31], GUIDON [10], Human tutorial dialog [19], human learning [34, 21, 28, 3, 12, 17], LISP Tutor [2], Meno-Tutor [44], PACT [1], QUADRATIC [37], SCHOLAR [9], SOPHIE [6], TRAIN-Tutor [43], UMFE [39], WEST [8], WHY [40].

Table 1. Some Tutoring and Learning Principles

The integration of new material with previous information can be done by analogies, generalizations and checking consistency. Through analogy, novel situations and problems can be understood in terms of familiar ones [21]. Effective human tutors ask for similarities and differences for similar cases [12]. In educational systems such as Atlas-Andes [38], 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 [12]. 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 [41], encourage students to abstract plans from the details to see the basic approach behind problem solving. Finally, cognitive dissonance theory [17] points out that people tend to seek consistency among their cognitions (i.e., beliefs, opinions). When there is an inconsistency (dissonance), something must change to eliminate the dissonance.

4. **Immediate feedback**
   Many educational systems provide immediate feedback on the quality of student's responses [6, 2]. The studies of feedback in a variety of instructional context find that immediate feedback is much more effective than feedback received after a delay [28]. Similarly, in the tutorial dialog study by Fox [19], tutors show immediate recognition of every step the student makes and their silence tends to presage the student's confusion. It is reported

that in providing feedback, human tutors are more flexible than educational software, using high bandwidth communication to guide the students [34].

5. **Generate educated guesses**
   Some educational systems invite guesses on questions either in the process of letting the student discover the answers [37] or in the process of assessing the student's knowledge[1]. Likewise, in the studies of human tutoring, student often display their understanding by finishing the tutor's utterance and the tutor finds out what students understood by inviting their guesses (utterance completion strategy) [19].

6. **Keep on track**
   Tutors need to keep track of the lesson and bring back issues that had to be dropped while engaging in clarifications or other side dialogues. If the student gives an incorrect answer, the tutor must immediately get the student back on track [9]. Some educational systems detect change of directions [43] or check if the questions are irrelevant to the case at hand [10].

7. **Indicate lack of understanding**
   Studies in human tutoring show cases where students themselves indicate lack of understanding of introduced terms [19], but tutors also point out the specific aspects introduced in a lesson that the student needs to understand.[12].

8. **Detect and fix "buggy" knowledge**
   Many educational systems have a tutoring goal of diagnosing the student's "bugs" [44, 40, 7] and question answering is often used in checking student's knowledge. However, simply telling that an error has occurred is much less useful than reminding the student of the current goal or pointing out a feature of the error [33]. If there are insufficient or unnecessary factors in a student's answers, experienced tutors pick counter examples to highlight the problem [12]. In the process of checking, when the tutor does not understand the answer, sometimes the student is asked to rephrase the answer [9].

9. **Learn deep models**
   The tutor and the student should focus on deep conceptual models and explanations rather than superficial ones [41]. 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 [1, 41]. In some cases, to be able to ensure that the student understood the explanation educational systems use a set of check questions [38]. Studies of human tutoring show that students themselves occasionally try to check the reasoning behind the answers provided [19].

10. **Learn domain language**
    Another interesting aspect of a lesson is learning to describe the new knowledge in terms that are appropriate in the domain at hand. Educators want to ensure that the students learn to talk science as a part of understanding

of the science [41]. Teaching is more difficult when the student organizes and talks about knowledge in a different way than the tutor does [44].

11. **Keep track of correct answers**
    Instructional systems keep track of the questions that the student is able to answer correctly as well as those answered incorrectly, which drives further interactions with the student. Some systems try more specific or simpler version of questions to keep better track of progress. [38].

12. **Prioritize learning tasks**
    Tutoring systems often handle multiple sub-tasks using priority rules that look at the duration and type of task. For example, systems can focus on errors before omissions and shorter fixes before longer fixes, prior steps before later steps, etc [12].

13. **Limit the nesting of the lesson** Tutoring dialogue is sometimes controlled by limiting the amount of subdialogues, which helps the student keep track of the lesson topics [41].

14. **Summarize back to teacher what was learned**
    Many educational systems summarize the highlights at the end of the lesson [44, 31]. For example, EXCHECK prints out review of the proof for the student to give a clear picture of what has been done [31]. In some systems, when the tutor has given several hints, a summary may be given to ensure that the student has correct information just in case the student gave right answer by following hints without understanding the procedures [43].

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 [19]. Also some educational systems have a way of isolating the weaknesses in the student's knowledge and propose further lessons on those areas [8].

These fifteen principles can be related to the techniques used in existing knowledge acquisition tools, as described in section 5.

4 Interactive Knowledge Acquisition Tools

The interactive knowledge acquisition tools summarized in Table 2 illustrate different approaches that researchers have undertaken over the years and are representative of the literature. The tools are briefly described in section 4.2. The techniques used range from cognitive theories of expertise and learning, case-based reasoning and analogy, non-monotonic theory revision, induction and machine learning, knowledge engineering approaches, analysis of knowledge interdependencies and buggy knowledge, and agent-based interaction.

Acquisition Tool	*Highlights*
CHIMAERA [32]	To acquire concepts, relations, and instances. Diagnoses faulty definitions.
EXPECT [5]	To acquire problem solving knowledge. Exploits dialogue scripts, knowledge interdependency models, and background knowledge.
INSTRUCTO-SOAR [23]	To acquire task models for Soar.
KSSn [20]	To acquire concepts, rules, and data. Based on personal construct psychology.
PROTOS [4]	Users specify cases, tool explains their classification.
SALT [30]	To acquire constraints and fixes for its underlying engine for configuration design.
SEEK2 [24]	To acquire rules. Uses verification and validation techniques.
SHAKEN [11]	To acquire process models. Loosely based on concept maps.
TAQL [45]	To acquire SOAR rules. Based on Problem Space Computational Model.
TEIREISIAS [15]	To acquire rules. Exploits context, derived rule models, and heuristics.

Table 2. Some Interactive Knowledge Acquisition Tools.

4.1 Functionality of Interactive Knowledge Acquisition Tools

Figure 2 shows a diagrammatic view of typical components used in various tools. The functionality of an interactive knowledge acquisition tool can be described along five dimensions, which we will use for reference later in our analysis:

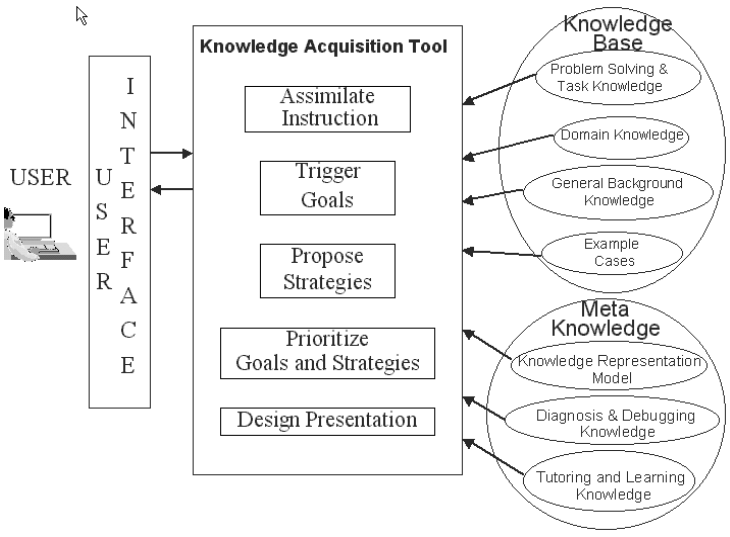

Fig. 2. A Modular View of Interactive Knowledge Acquisition Tools.

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

piece of knowledge (CHIMAERA, EXPECT, SALT, TAQL), or a graphical rendering (KSSn, SHAKEN).

- **Trigger goals:** The system analyzes its knowledge and generates learning goals of what knowledge it still needs to acquire. Many tools focus on detecting inconsistencies or gaps in the knowledge base, which generate the goals to fix them or seek the information missing (CHIMAERA, EXPECT, SEEK2, TAQL, TEIREISIAS).
- **Propose strategies:** The tool can generate possible strategies that the user could follow in achieving the learning goals. It can also generate predictions of what strategy the user is more likely to pursue, or what answers the user is more likely to give to the user's questions (TEIREISIAS). This is often done by analyzing existing knowledge. Planning strategies are often used to make suggestions to the user in terms of what to do to achieve the active learning goals, which will make the acquisition process more efficient (EXPECT).
- **Prioritize goals and strategies:** An acquisition tool can help users further if it is able to organize and prioritize the active learning goals and candidate strategies, so that it can make more focused suggestions to the user. Sometimes these are organized by the type of knowledge sought (SALT), or by the type of goal being pursued or error being fixed (EXPECT).
- **Design presentation:** The tool can make decisions about what to bring to the attention of the user at each point in time to help him or her decide what to do next. There are many possibilities, and the tool can take into account the user's situation (user modeling), the stage of the process (initial stage versus final testing), and the content of the current knowledge base. The tool may present the user with a single question (INSTRUCTO-SOAR) or give the user a choice in the form of an agenda containing multiple items (CHIMAERA, EXPECT, SALT). The tool can suggest a specific strategy, anticipate the user's answer and ask for confirmation, or simply present the user with multiple possible strategies and suggestions (EXPECT). Other tools leave it up to the user to figure out what to do and simply make all possible options available to them (KSSn, SHAKEN) The tool may simply ask the user to review an explanation (PROTOS), check some aspect of the knowledge (KSSn), or confirm a hypothesis (TEIREISIAS).

Another useful view of interactive acquisition tools is in terms of the kinds of knowledge and meta-knowledge that they bring to bear in order to support the user, also illustrated in Figure2. This includes:

- *General problem solving and task knowledge*: General inference structures are used to determine the role that domain-specific knowledge plays in problem solving, as is done in role-limiting approaches to knowledge acquisition (Marcus & McDermott, 1989) (e.g., SALT, TAQL).
- *Prior domain knowledge*: The initial knowledge base may contain terms that are specific to the domain at hand and that can be used to define new terms and tasks (e.g., EXPECT, INSTRUCTO-SOAR).

- *General background knowledge*: The initial knowledge base may include high level theories and ontologies that capture general knowledge, such as time, physical objects, etc. (e.g., SHAKEN).
- *Example cases*: Sample situations, test cases, and problem solving episodes can help ground abstract knowledge (e.g., INSTRUCTO-SOAR, PROTOS, SEEK2, SHAKEN, TEIREISIAS).
- *Underlying knowledge representation*: Models of the underlying knowledge representation will determine how users need to formulate new knowledge (e.g., CHIMAERA, KSSn, SEEK2, TAQL, TEIREISIAS).
- *Diagnosis and debugging knowledge*: Typical diagnosis skills are useful in order to detect errors and potential problems in the knowledge base. Effective debugging strategies can be incorporated to make suggestions to the user about how to fix the errors and problems found (e.g., CHIMAERA, EXPECT, TEIREISIAS).

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

4.2 A Brief Overview of Interactive Knowledge Acquisition Tools

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

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

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

and by showing an agenda of errors in the knowledge base together with suggested fixes.

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

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

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

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

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

SHAKEN — SHAKEN (Clark et al., 2001) allows end users to specify process models. It uses a graphical editor inspired by concept maps (Novak, 1998), and extended so that process steps and objects are modeled after an

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

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

existing class in the background knowledge base. Users can test the knowledge entered by instantiating a question template, or by requesting an error report from running a simulation of the process.

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

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

5 Tutoring and Learning Principles in Existing Interactive Acquisition Tools

In section 3, we described our analysis of tutoring and educational literature and presented fifteen principles that humans and computers exploit to make teaching and learning more effective. We noticed that many of the principles in learning

and tutoring could be related to the techniques used in existing acquisition tools. Yet, the tutoring literature is seldom mentioned in knowledge acquisition work. In this section, we describe our views on how acquisition techniques can be expressed in terms of these tutoring and learning principles. Table 3 summarizes our analysis, indicating the particular functionality (as outlined in Figure 2) where the principle was applied in specific acquisition tools.

5.1 Introduce lesson topics and goals

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

5.2 Use topics of the lesson as a guide

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

5.3 Subsumption to existing cognitive structure

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

5.4 Immediate feedback

PROTOS provided immediate feedback as a new case was assimilated by showing the user an explanation of its classification in the knowledge base. INSTRUCTO-SOAR generated clarification and follow-up questions for the user immediately after an instruction was given. TEIREISIAS proposed amendments to rules as

soon as the user defined them. EXPECT analyzes the knowledge base after each user action and shows immediately an agenda of errors to resolve and tasks to do.

5.5 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.)

5.6 Keep on track

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

5.7 Indicate lack of understanding

INSTRUCTO-SOAR detects missing aspects of a task description specified by a user and generates follow up questions. EXPECT and CHIMAERA detect undefined terms that will be used to guide future dialogue with the user to define them.

5.8 Detect and fix "buggy" knowledge

TAQL analyzes the knowledge specified by the user and points out errors based on static analysis. CHIMAERA and EXPECT detect errors in the knowledge entered that need to be fixed by the user. PROTOS, SEEK2, and TEIREISIAS show explanations or traces to users so they can detect errors in the system's reasoning.

5.9 Learn deep models

Knowledge acquisition tools do not have any basis to evaluate or pursue depth in their knowledge base, though this is a long recognized shortcoming of knowledge-based systems. To date, these systems are at the mercy of the user's intention and of their implementation of any depth in the models.

5.10 Learn domain language

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

5.11 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.

5.12 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).

5.13 Limit the nesting of sub-lessons

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

5.14 Summarize what was learned

Acquisition tools do not summarize what they have learned.

5.15 Assess learned knowledge

KSSn uses clustering techniques to suggest aspects of the model that users could detail further. Other acquisition tools do not perform this kind of analysis. Users often have to put the knowledge base through a performance system that exercises it in order to be able to assess if the knowledge was learned appropriately.

6 Discussion

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

state, goals, and possible strategies could turn interactive acquisition tools into more proficient and proactive learners.

The principles have only been used in some aspects of the functionality of acquisition tools, and are exhibited by some but not all the tools. The sparseness of the matrix in Table 3 points to many opportunities for future work in incorporating these principles. By having declarative representations of their learning state, goals, and possible strategies, interactive acquisition tools could more easily incorporate these principles in all five functions of the acquisition process shown in the table.

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

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

7 Conclusions

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

we identified could be used more thoroughly in future interactive knowledge acquisition tools by incorporating them into different functions of the tools.

We are currently incorporating these principles in our work. We are designing a front-end dialogue system that can be layered over the functionality of existing acquisition interfaces. The learning principles in this paper are stated in general terms, and we are investigating how they can be operationalized by taking into account the features supported by the specific acquisition interface. By structuring the dialogue in a way that exploits these principles, the interactive acquisition tools will interact with users as more proactive learners. In this work, we are designing declarative representations of the state and progress of the lesson that will enable the system to to assess its competence and confidence on the lesson topics as the dialogue with the user progresses. We are also investigating how to combine these principles with relevant dialogue planning and user interaction techniques.

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

Our hope is that this paper will motivate a new generation of interactive acquisition tools that can represent meta-knowledge about the learning process that they are engaging the user in, and can use this meta-knowledge to track the progress made throughout the dialogue and to structure subsequent interactions. This will result in a new generation of interactive acquisition tools that behave more like capable, skilled, and proactive learners.

Acknowledgments

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

References

1. 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.*
2. Anderson, J. R., Conrad, F. G., & Corbett, A. T. (1989). Skill acquisition and the lisp tutor. *Cognitive Science*, 13:467–506.

3. Ausubel, D. (1968). *Educational psychology: A cognitive approach.* New York, Holt, Rinehart and Winston.
4. Bareiss, R. & Porter, B. & Holte, R. (1990). Concept learning and heuristic classification in weak-theory domains. *Artificial Intelligence Journal* 45(1-2):229–264.
5. Blythe, J.; Kim, J.; Ramachandran, S.; and Gil, Y. (2001). An integrated environment for knowledge acquisition. In *Proceedings of the IUI-2001.*
6. Brown, J. S., Burton, R., & de Kleer, J. (1982). Pedagogical natural language and knowledge engineering techniques in SOPHIE I, II, III. In Derek, S. & Brown, J. S., (Eds.), *Intelligent Tutoring Systems.* New York, Academic Press.
7. Brown, J. S. & Burton, R. R. (1978). Diagnostic models for procedural bugs in basic mathematical skills. *Cognitive Science*, 2:155–191.
8. Burton, R. & Brown, J. (1979). An investigation of computer coaching for informal learning activities. *International Journal of Man-Machine Studies*, 11:5–24.
9. Carbonell, J. R. (1970). AI in CAI: An artificial intelligence approach to computer-assisted instruction. *IEEE Transactions on Man-Machine Systems*, 11(4):190–202.
10. Clancey, W., (Ed.) (1987). *Knowledge-Based Tutoring:The GUIDON Program.* MIT press.
11. Clark, P., Thompson, J., Barker, K., Porter, B., Chaudhri, V., Rodriguez, A., Thomere, J., Mishra, S., Gil, Y., Hayes, P., & Reichherzer, T. (2001). Knowledge entry as the graphical assembly of components. In *Proceedings of K-CAP-2001.*
12. Collins, A. & Stevens, A. L. (1982). Goals and strategies of inquiry teachers. *Advances in Instructional Psychology*, 2:65–119.
13. 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.*
14. Cowie, J., & Lehnert, W. (1996). Information extraction. *Communications of the ACM* 39(1):80–91.
15. Davis, R. (1979). Interactive transfer of expertise: Acquisition of new inference rules. *Artificial Intelligence*, 12:121–157.
16. 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.
17. Festinger, L. (1957). *A Theory of Cognitive Dissonance.* Stanford University Press.
18. Forbus, K. & Feltovich, P., (Eds.) (2001). *Smart Machines in Education.* AAAI press.
19. Fox, B. (1993). *The Human Tutorial Dialog Project.* Lawrence Erlbaum.
20. Gaines, B. R. & Shaw, M. (1993). Knowledge acquisition tools based on personal construct psychology. *The Knowledge Engineering Review*, 8(1):49–85.
21. Gentner, D., Holyoak, K. J., & Kokinov, B. N., (Eds.) (2001). *The analogical mind: Perspectives from cognitive science.* MIT press.
22. Gil, Y. & Melz, E. (1996). Explicit representations of problem-solving strategies to support knowledge acquisition. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence.*
23. Huffman, S. B., & Laird, J. E. (1995). Flexibly instructable agents. *Journal of Artificial Intelligence Research* 3:271–324.
24. Ginsberg, A., Weiss, S., & Politakis, P. (1985). SEEK2: A generalized approach to automatic knowledge base refinement. In *Proceedings of IJCAI-85.*
25. Kim, J. & Gil, Y. (1999). Deriving expectations to guide knowledge base creation. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pp. 235–241.

26. Kim, J. & Gil, Y. (2000). Acquiring problem-solving knowledge from end users: Putting interdependency models to the test. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*.
27. Koedinger, K., Anderson, J., Hadley, W., & Mark, M. (1997). Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education*, 8:30–43.
28. Kulik, J. & Kulik, C. (1988). Timing of feedback and verbal learning. *Review of Educational Research*, 58:79–97.
29. Lepper, M., Woolverton, M., Mumme, D., & Gurtner, J. (1993). Motivational techniques of expert human tutors: Lesson for the design of computer-based tutors. In Lajoie, S. & Derry, S., (Eds.), *Computers as Cognitive Tools*, pp. 75–105. Hillsdale.
30. Marcus, S. & McDermott, J. (1989). SALT: A knowledge acquisition language for propose-and-revise systems. *Artificial Intelligence*, 39(1):1–37.
31. McDonald, J. (1981). The EXCHECK CAI system. In Suppes, P., (Ed.), *University-level Computer-assisted Instruction at Stanford: 1968-1980*. Stanford.
32. McGuinness, D. L., Fikes, R., Rice, J., & Wilde, S. (2000). An environment for merging and testing large ontologies. In *Proceedings of KR-2000*.
33. McKendree, J. (1990). Effective feedback content for tutoring complex skills. *Human Computer Interactions*, 5:381–413.
34. 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.
35. 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.
36. Novak, J., (Ed.) (1998). *Learning, Creating, and Using Knowledge: Concept Maps as Facilitative Tools in Schools and Corporations*. Lawrence Erlbaum.
37. O'Shea, T. (1979). A self-improving Quadratic tutor. *International Journal of Man-Machine Studies*, 11:97–124.
38. Rose, C. P., Jordan, P., Ringenberg, M., Siler, S., VanLehn, K., & Weinstein, A. (2001). Interactive conceptual tutoring in Atlas-Andes. In *Proceedings of AI in Education*.
39. Sleeman, D. H. (1984). Inferring student models for intelligent computer-aided instruction. In Michalski, R. S., Carbonell, J. G., & Mitchell, T. M., (Eds.), *Machine Learning: An Artificial Intelligence Approach*, pp. 483–510. Springer.
40. Stevens, A. & Collins, A. (1977). The goal structure of a Socratic tutor. In *Proceedings of the National ACM Conference*.
41. VanLehn, K., Freedman, R., Pamela, J., Murray, C., Osan, R., Ringenberg, M., Rose, C., Schulze, K., Shelby, R., Treacy, D., Weinstein, A., & Wintersgill, M. (2000). Fading and deepening: The next steps for Andes and other model-tracing tutors. In *Proceedings of ITS-2000*.
42. Wenger, E., (Ed.) (1987). *Artificial Intelligence and Tutoring Systems*. Morgan Kaufmann.
43. Woolf, B. & Allen, J. (2000). Spoken language tutorial dialogue. In *Proceedings of the AAAI Fall Symposium on Building Dialogue Systems for Tutorial Applications*.
44. Woolf, B. P. & McDonald, D. D. (1984). Building a computer tutor: Design issues. *IEEE Computer*, 17(9):61–73.
45. Yost, G. R. (1993). Knowledge acquisition in Soar. *IEEE Expert* 8(3):26–34.
46. Zhou, Y., Freedman, R., Michael, M. G. J., Rovick, A., & Evens, M. (1999). What should the tutor do when the student cannot answer a question? In *Proceedings of FLAIRS-99*.