User Studies of Knowledge Acquisition Tools: Methodology and Lessons Learned

Marcelo Tallis, Jihie Kim, and Yolanda Gil

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

Abstract

The area of knowledge acquisition research concerned with the development of knowledge acquisition (KA) tools is in need of a methodological approach to evaluation. Efforts such as the Sisyphus experiments have been useful to illustrate particular approaches, but have not served in practice as testbeds for comparing and evaluating different alternative approaches. This paper describes our experimental methodology to conduct studies and experiments of users modifying knowledge bases with KA tools. We also report the lessons learned from several experiments that we have performed. Our hope is that it will help others design or improve future user evaluations of KA tools. We found that performing these experiments is particularly hard because of difficulties in controlling factors that are unrelated to the particular claims being tested. We discuss our ideas for improving our current methodology and some open issues that remain.

1 Introduction

The field of AI has increasingly recognized throughout the years the need and the value of being an experimental science. Some subfields of AI have developed standard tasks and test sets that are used routinely by researchers to show new results. Researchers in machine learning, for example, use the Irvine data sets to show improvements in inductive learning (Quinlan, 1993; Webb *et al.*, 1999) and routinely use tasks like the n-puzzle or the n-queens for speed-up learning research (Tambe & Rosenbloom, 1990; Kim & Rosenbloom, 1996).

Developing standard tests is harder in other subfields that address more knowledge-intensive problems. For example, planning researchers often show experiments in similar task domains (Gil, 1992; Gil, 1991; Pérez & Carbonell, 1994; Estlin, 1998). The problem is that the implementation of the knowledge base and of the algorithms is so different across systems that the results of the experiments are often hard to analyze. One approach used by some researchers is to use artificially created, very structured knowledge bases to analyze particular behaviors (Barrett & Weld, 1994; Veloso & Blythe, 1994). Another approach has been to define a universal language to describe planning domains, as is done in the Planning Competition of the Artificial Intelligence Planning Systems Conference (McDermott, 1998).

Knowledge acquisition research is still in need of a methodological approach to evaluation. The lack of evaluation in this area is unfortunate, but may be due to a variety of reasons. First, KA 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 KA tool 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. Second, most of the research in the field of KA 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 very few efforts on developing tools for users. Developers may conduct usability studies, but the results are not reported in the literature.

In recognition of the need to evaluate KA research, the community started to design a set of standard task domains that different groups would implement and use to compare their work. This effort is known as the Sisyphus experiments (Linster, 1992; Schreiber & Birmingham, 1996; Shadbolt, 1998), and the domains have included office assignment, elevator configuration, and rock classification. These experiences have been useful to illustrate particular approaches, but have not served in practice as testbeds for comparing and evaluating different approaches (Gil & Linster, 1995). The most recent Sisyphus is an example of the issue discussed above about the intimidating cost of KA evaluations: 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, 1998).

As developers of KA tools (Gil, 1994; Tallis, 1999; Kim & Gil, 1999a), we wanted to evaluate our approaches, and began looking into user studies. With the exception of some isolated evaluations of KA tools (Yost, 1992; Joseph, 1992; Murray, 1995), we found that the field of knowledge acquisition had no methodology that we could draw from to design our evaluations. Even though AI is, as we mentioned earlier, a field where experimental studies have been increasingly emphasized in recent years, user studies are uncommon. User studies to evaluate software tools and interfaces can be found in the literature of tutoring systems (Self, 1993), programming environments (Basili *et al.*, 1986), and human computer interfaces (Olson & Moran, 1998). These communities are still working on developing evaluation methodologies that address their specific concerns. All seem to agree on the difficulty and cost of these studies, as well as on their important benefits. Often times, the evaluations that test specific claims about a tool or approach are perhaps not as thorough or conclusive as we would like to see as scientists, yet we are lucky that these evaluations are taking place at all and are shedding some light on topics of interest (Rombach *et al.*, 1992; Self, 1993; Basili *et al.*, 1986; Olson & Moran, 1998). In developing a methodology for evaluation of KA tools, we can draw from the experiences that are ongoing in these areas.

Over the last few years, we have performed a series of evaluations with our KA tools that have yielded not only specific findings about our tools but that have also allowed us to develop a methodology that we follow in conducting our evaluations. This paper describes our experimental methodology to conduct studies of users modifying knowledge bases with KA tools. It also reports the lessons learned from our experiments so it will help others design or improve future user evaluations of KA tools. This paper describes our experiments in enough detail to illustrate the different point of our methodology and the lessons learned. A more comprehensive description of our experiments and their results can be found in the literature (Tallis & Gil, 1999; Tallis, 1999; Gil & Tallis, 1997; Kim & Gil, 1999a; Kim & Gil, 1999b).

The paper describes our experiences based on tests with two particular KA tools that we developed for the EXPECT framework (Gil & Melz, 1996; Gil, 1994; Swartout & Gil, 1995). A brief overview of both tools can be found in Appendix A. ETM (EXPECT Transaction manager) (Tallis & Gil, 1999; Tallis, 1999; Gil & Tallis, 1997) uses typical KB modification sequences (KA Scripts) to help users modify KBs. EMeD (EXPECT Method Developer) (Kim & Gil, 1999a; Kim & Gil, 1999b) analyzes and exploits interdependencies among KB elements to guide users in making significant KB extensions or changes. Each tool was developed to investigate a different approach to guide

1. State general claims and specific hypotheses – what is to be tested
2. Determine the set of experiments to be carried out – what experiments will test what hypotheses
3. Design the experimental setup
   (a) Choose type of users that will be involved – what kind of background and skills
   (b) Determine the knowledge base used and KA task to be performed – what kinds of modifications or additions to what kinds of knowledge
   (c) Design the experiment procedure – what will the subjects be told to do at each point
4. Determine data collection needs – what will be recorded
5. Perform experiment
6. Analyze results – what results are worth reporting
7. Assess evidence for the hypotheses and claims – what did we learn from the experiment

TABLE 1: Steps for designing experiments to evaluate KA tools. It is very useful to conduct one or more pre-tests, which involves iterating through these steps to refine the overall experiment design design.

users in knowledge acquisition tasks. The approaches are complementary, and we are now in the process of integrating the features of the tools that we found useful in the experiments in order to create a more comprehensive and powerful KA environment for EXPECT.

The paper begins by describing the methodology that we follow to design experiments to evaluate KA tools, illustrated with examples from our evaluations with ETM and EMeD. The next section highlights the lessons that we learned in carrying out these experiments, and describe open issues in KA experiment design. Finally, we discuss related work in other research disciplines that conduct user studies and outline directions for future work.

2 A Methodology to Conduct Experimental User Studies with Knowledge Acquisition Tools

The nature of an experiment is determined by the claims and hypotheses to be tested. Based on the hypotheses to be tested, we need to determine the KA task to be performed by the users, the underlying representations for the knowledge acquired by the tool, the type of users involved, the procedure to be followed to perform the experiment, and the data that needs to be collected. This section discusses each of these issues in detail.

Table 1 summarizes the steps in our methodology. It is by no means a strictly sequential process, rather there is significant iteration and backtracking across these steps due to the interactions among all the constraints and decisions involved. For example, a hypothesis may be revisited if an experiment cannot be designed to test it as it is stated. It is extremely useful to run a pre-test using a smaller-scale or a preliminary version of the experimental setup (e.g., fewer users), so that the design of the overall experiment can be debugged, refined, and validated.

2.1 Stating Claims and Hypotheses

Claims and hypotheses play a pivotal role in the evaluation process, since the experiments revolve around them. Claims and hypotheses are related but not necessarily the same. Claims are stated in broader terms, referring to general capabilities and benefits of our tools. It may or it may not be possible to test a certain claim, but it helps us understand what we think are the advantages of a certain approach. Based on these broader claims, we formulate specific hypotheses that we would like to test. In contrast with claims, hypotheses are stated in specific terms, and we formulate them such that an experiment can be designed to test them and yield evidence that will lead to proving or disproving specific hypotheses. In reality, some experiments are potentially possible but turn out to be infeasible in practice due to lack of time and other resources.

The first step in the design of our evaluations was to state the main claims regarding our KA tools. It turns out that we made similar claims for both tools:

1. Users will be able to complete KA tasks in **less time** using the KA tools.
2. Users will make **less mistakes** during a KA task using the KA tools.
3. The reduction in completion time and number of mistakes will be more noticeable for **less experienced users**.
4. The reduction in time will also be more noticeable for **users lacking a detailed knowledge** of the KBS implementation.
5. The KA tools will be useful for a **broad range of domains and knowledge acquisition scenarios**.

Given these claims, we were able to state specific and measurable hypotheses to be proved or disproved with experiments that were feasible give our resources and constraints.

For example, a specific hypothesis for ETM corresponding to claim 1 is: Completion time for a complex KBS modification will be shorter for subject using ETM (plus EXPECT) than for subjects using EXPECT alone.

A claim can be stated in more general or more specific terms depending on the purpose of the claim. The claims that we showed above are specific to particular KA tools and methodologies, but it would be make them part of more general claims that the whole KA field cares about and that other researchers may want to hear about the state-of-the-art in KA. For example, our experiments and those of others might help us gather evidence towards general claims such as *"It is possible for naive users to make additions and changes to a knowledge base using KA technology"*, with more specific claims stating what technologies help in what kinds of KA tasks to what kinds of users.

2.2 Determining the set of experiments to be carried out

It is useful to test one or more hypotheses in few experiments, but it is not always possible. This is the case when the hypotheses are of a very different nature, or when a given hypothesis needs to be tested over a range of user types, tasks, or knowledge bases. For example, if we have two different hypotheses, such as (1) a KA tool helps to perform a task more efficiently and (2) the KA tool

scales up to large and realistic applications, then it might be necessary to conduct one experiment to support the first hypothesis and a second experiment for the second hypothesis.

A useful way to design an experiment that establishes the benefits of a tool or a technology is to perform a comparison with some baseline tool. In this case, we have to carefully choose the tools to be compared. The only difference between the two tools to be compared has to be the presence or absence of the technology to be evaluated. Otherwise, we may not be able to determine if the differences in the performance of the tools were due to the technology itself or to some other factors (e.g., a different interface design or interaction style). For example, to test the benefits of expectation-based KA, we compared EMeD against a basic KA tool that consisted of the same EMeD interface where a number of features had been disabled. That is, both tools (EMeD and the Basic EMeD) provided a similar user interface environment.

This kind of experiment is a tool *ablation* experiment, where the object of ablation is some capability of the tool. The group of subjects that is given the ablated KA tool serves as the *control group*. We have found these experiments to be the most useful and compelling kind to test for our claims.

2.3 Designing the Experimental Setup

Once we have determined the hypotheses and the kind of experiment to be carried out, we are able to plan the details of the experiment.

2.3.1 Users

An important issue is the choice of subjects who are going to participate. Practical concerns often constrain the experimentation possibilities, for example the accessibility and availability of certain types of subjects.

We design our experiments as *within subject* experiments. This means that each subject uses both the ablated and the non-ablated version of the KA tool (but not to do the same task, as we describe below). Because of the many differences among subjects and the small amount of subjects that we could test, this helps to reduce the effect of individual differences across users. Different subjects use the two versions of the tool in different orders, so as to minimize the effects that result from increasing their familiarity with the environment that we provide. Another possibility is to use a larger number of subjects, and divide them into two separate groups: one that uses only the ablated tool and the other that uses the non-ablated tool. The problem with this design is that it is more costly.

We have identified several types of users according to their background and skills, particularly with respect to their familiarity and expertise with knowledge base development and knowledge acquisition tools and techniques.

2.3.2 Domains, KA Tasks and Scenarios

One issue is the choice of the application *domain*. For our purposes, the knowledge bases needed to be simple enough to be learned in a short time but complex enough to challenge the subjects in the control group. For ETM, we chose a transportation planning evaluation system developed as a part of a DARPA funded project (Gil & Swartout, 1994). An example of a KA task given to the subjects

is to modify a fragment of an existing knowledge-based system capable of evaluating transportation movements carried only by ships in order to enable the system to evaluate movements involving both ships and aircraft. This modification required to add new problem-solving knowledge to handle aircraft and to integrate this new knowledge with the knowledge that already existed in the knowledge-based system.

For EMeD, we chose a Workarounds domain selected by DARPA as one of the challenge problems of the High-Performance Knowledge Bases program (Cohen *et al.*, 1998) that investigates the development of large-scale knowledge based systems. The domain task is to estimate the delay caused to enemy forces when an obstacle is targeted by reasoning about how they could bypass, breach or improve the obstacle. An example of a KA task given to subjects is to add problem-solving knowledge to "estimate the time to move military assets by enemy units", considering the source of the assets, current location, destination, and their moving speeds. For each task, the subjects added new problem-solving knowledge to the system.

There is a range of *difficulty of KA tasks* in terms of the kinds of extensions and/or modifications to be done to a KB. A KA task that requires only adding knowledge is very different in nature and difficulty from a KA task that requires modifying existing knowledge. Also, modifying problem-solving knowledge is a very different task from adding instances, even if they are both KA tasks. It is important to design the scenario so that it covers the kind of KA tasks that the tool is designed for. Both our tools are targeted to the acquisition of problem solving knowledge. We tested ETM with a KB modification task, and EMeD with a KA task that required extending an existing KB by adding new knowledge.

Another issue in within subject experiments is that if one gives a subject the same exact KA task to do with the two versions of the tool there will most probably be a *transfer effect*. This means that it is pretty certain that they will be unlikely to repeat errors the second time they do the task, and that they will remember how they did something and will not need to figure it out the second time around. To avoid this transfer effect, we design two different but comparable *scenarios*, each involving the same kind of KA task in the same domain but involving a different aspect of the knowledge base. One scenario is carried out with the ablated tool and the other one with the non-ablated version of the tool. It is very important that both scenarios are as comparable in size and complexity as possible in order for the results of the experiments to be meaningful. This results in *four subject groups*, each with a specific combination of the two versions of the tools and the two KA scenarios to be carried out.

To facilitate the subject's acquisition of the knowledge required to perform the scenarios and to ensure a uniform understanding of the domain across users, the application domain was explained to all the subjects during a presentation. The subjects and the domain were especially chosen to avoid markedly differences in the subjects previous exposure to the domain. Of course, these points are not relevant when testing subjects that are experts in a specific domain.

A repository of knowledge bases and scenarios to test KA tools that could be shared by different researchers would enable better comparative evaluations among approaches, as well as reduce the amount of work required to evaluate a KA approach. The knowledge bases that we used are available to other researchers by contacting any of the authors.

2.3.3 Experiment procedure

After we had determined the kind of experiment to be carried out, our hypotheses, the type of users, and the nature of the KA task, we were in condition to plan other details of the experiment. These include, for example, what information will be given to the subjects and in what format, what kind of interaction can the subjects have with the experimenters during the tests (e.g., can they ask additional questions about the domain and/or the tool), how many iterations or problems will be given to each subject and in what order, and an indication of the success criteria for the subjects so they know when they have finished the scenario they were given (e.g., the final KB correctly solves some given problems, perhaps according to a gold standard).

The experimental setup has to be carefully designed to control as much as possible the variables that can affect the outcome of the experiment and that are not related to the claims. For example, if the disparity of subjects may affect the results we can have each subject to perform two tasks, one with the tool to be evaluated and the other with the tool used for control. In this case, if the order in which the tasks are executed might also affect the results, we can switch the order in which the tasks are executed for different subjects. If we suspect that not all subjects have the sufficient skills to perform the requested task we can include a practice session previous to the evaluation. We can also ask the subjects to fill a background questionnaire previous to the execution of the experiment to form the groups as balanced as possible.

Our controlled experiments compared the performance of subjects using EXPECT vs. subjects using enhanced versions of EXPECT that use certain KA techniques. Each subject performed two different scenarios, one with EXPECT and the other with EXPECT plus tool, so the results were independent of differences in subject's background and skills. Each scenario was performed with both tools so the results were independent of the complexity of the scenario. Some subjects used plain EXPECT first and others used an enhanced version of EXPECT first so the results were independent of the order in which the tools were used. All these factors were balanced such that they would even out the qualifications of the subjects for each one of the groups, the number of times that each tool was used for each scenario, and the number of times that each tool was used in the first place.

All experiments followed the same general procedure distributed in two stages:

Stage 1: Domain and tools presentation

The subjects attend a presentation that introduced EXPECT, ETM or EMeD, and the application domain.

Stage 2: Practice and execution

The subjects perform the following activities:

1. Execute a practice scenario comparable to the ones to be used during the actual test. This scenario was performed once with each version of the KA tool. The purpose of this practice is to make subjects familiar with the tools, the domain, and the procedure of the experiment.

2. Execute two test scenarios, one using each version of the KA tool, alternating the order of the tool that is used first.

3. Answer a feedback questionnaire regarding their impressions and difficulties in using both tools. Each question is given numerical range (1 to 5), so that the answers are more comparable

across subjects.

For each test scenario, the subjects analyze the domain and the specifications. Then, they perform the given modification scenario until the KBS gave the correct results in the sample problem. During the execution of the scenarios, the experimenter only occasionally assists subjects that had problems interpreting the instructions, using the KA tools, or that get stuck or confused.

We use several approaches to determine when a subject has completed a KA task appropriately. In most cases, we take advantage of the formal validation mechanisms in EXPECT. That is, EXPECT can be given general variabilized goals, such as "estimate time to narrow a gap with any bulldozer", which represent the kinds of actual problem instances that the system will be given for execution. Given the variabilized goals, subjects are asked to make sure they can be achieved correctly. Also, in some cases, the subjects are asked to execute a set of problems, and examine the output to make sure they obtain the expected results. In addition, after each experiment, we check the knowledge added by the subjects by hand.

2.4 Determining what Data to Collect

The kind of data collected during the experiment may be determined and/or limited by what is possible in terms of *instrumenting the KA tool and the KB environment*. Also, intrusive ways of recording data should be avoided. For example, we should not ask the users to fill a long form to describe what they just did if that is going to disrupt their train of thought and make the overall time to complete the task longer.

The following data was collected during the execution of our scenarios:

- time to complete the whole KA task
- automatic log of changes performed to the KBS (e.g., a new parameter was added to a goal)
- automatic log of errors in the KB after each change to the KBS (e.g., a problem solving goal cannot be achieved)
- automatic log of commands executed (e.g., add a new problem solving substep similar to an existing one)
- detailed notes of the actions performed by the subjects (taken manually by the experimenters) including how they approached the problem and what materials they consulted. For this purpose we ask the subjects to voice what they are thinking and doing during the execution of the scenario. We do not use video cameras and tapes, since we find the notes to be sufficient and more cost-effective.
- questionnaires that the subjects fill out at the end of the experiments, with questions regarding the usability of the tools

The data collected should be sufficient to measure our hypothesis and confirm whether they stand or not. Collecting fine grained data is very useful because it not only proves/disproves the hypotheses, but it also helps to explain the outcome of the experiment, and to explore the potential causes of certain experiment outcomes.

We find that conducting pre-tests is very useful not only to help refine the actual evaluation setup but also the data collection strategy.

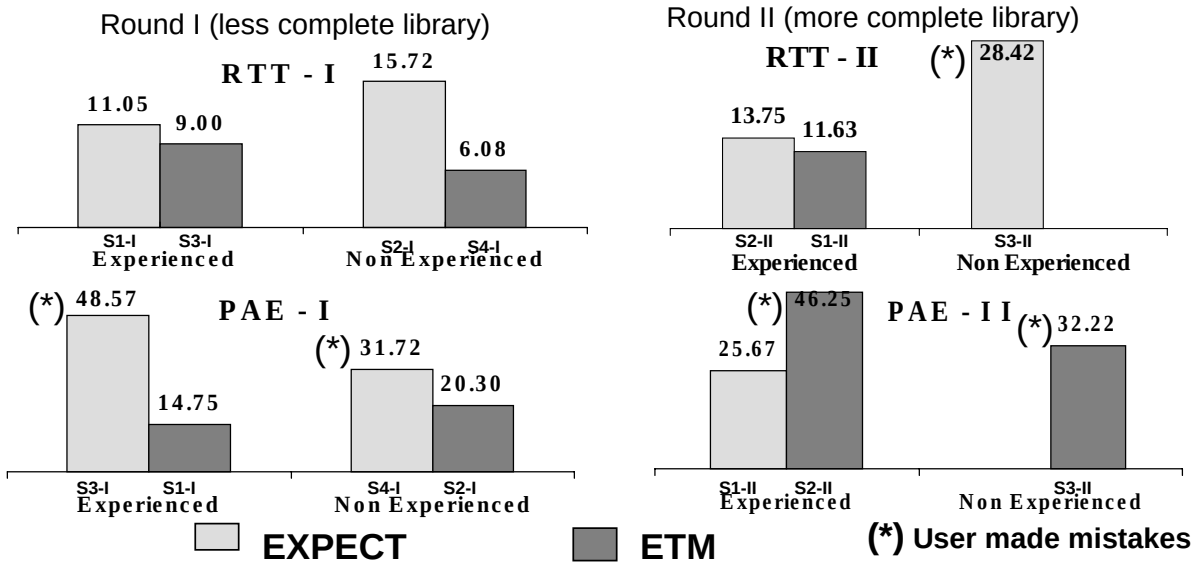

FIGURE 1: ETM evaluation results: Completion times (in minutes)

Results	Time (average in min)	Avg. number of Methods Added	Time/Method (min)
Experienced EXPECT users(4):Without EMeD	54.50	6.25	8.72
Experienced EXPECT users(4):With EMeD	38.25	6.00	6.38
Experienced KBS users(2):Without EMeD	65.50	6.50	10.08
Experienced KBS users(2):With EMeD	39.00	5.00	7.80
Non-Experienced users(4):Without EMeD	88.00	5.50	16.00
Non-Experienced users(4):With EMeD	55.75	5.25	10.62
Without EMeD (all)	70.10	6.00	11.68
With EMeD (all)	45.41	5.50	8.25

TABLE 2: EMeD Results.

2.5 Analyzing the Data

To illustrate this step, we show some data from the experiments with ETM and EMeD, and the conclusions that we extracted from them.

For ETM, the experiments were carried out in two rounds (I and II) with slight changes in the KA tools and the procedures between each round. Each subject executed two KA tasks, RTT and PAE, hence we have four data sets, RTT-I, RTT-II, PAE-I, and PAE-II. Figure 1 shows the completion time for each one of the data sets.

Table 2 shows some of the data collected for EMeD. The total time is computed by adding the times with each subject for each tool. The time for each subject is the time to complete the given task. The results for different user groups are shown separately to contrast the results. Also, the last row in the table summarizes the results. (The numbers in parentheses indicate the number of users in the group).

2.6 Assessing the Evidence for or against Hypotheses and Claims

The following are our conclusions from the experiments with ETM:

- **Subjects using ETM took less time** to complete the modification **except when they made mistakes** (hypothesis 1). Figure 1 indicates with an asterisk (*) the subjects that made mistakes during the execution of the experiment. Subjects in both groups, ETM and the control group, made costly mistakes that severely affected their completion time yet handling those kinds of mistakes was outside of the scope of ETM. As a result, these data points are problematic. Section 3 discusses this issue further. In (Tallis, 1999) we present a detailed analysis of the cases that included mistakes and show that if the time incurred in handling the mistakes is subtracted then subjects in the control group take longer to complete the modification than subjects using ETM.
- The differences in time were **greater for the less experienced subjects**.
- The experiments **did not show clear evidence that ETM would reduce the number of user mistakes**. However, we believe that some of the mistakes made by subjects in the control group would have been avoided if they were using ETM.
- ETM was able to **lead subjects throughout all the required changes of the scenarios** for most of the subjects that used ETM. In the few cases in which subjects had to perform changes without ETM, these cases were out of the scope of ETM. This virtue of ETM was not predicted beforehand.

The following statements summarize our conclusions from the EMeD experiments:

- **Subjects using EMeD took less time** to complete the KA tasks. EMeD was able to reduce the development time to 2/3 of the time that users needed without it.
- The differences in time were not so evident for the less experienced subjects. The ratio for less experienced subjects remain about the same as the ratio for EXPECT users.

In addition to these, the results show that subjects needed to add slightly less KB elements with EMeD. We may use this kind of additional data to explore some other hypotheses in the future, such as the quality of the output KBS. Also, we may be able to understand why the ratios do not improve for less experienced users.

In summary, sometimes the results provide evidence of our hypotheses although we may not be able to confirm them conclusively. This attests to the difficulty of designing this kind of experiments.

We also find very useful to analyze the data in detail, looking for interesting and unexpected phenomena. For example, the ETM experiment showed that users followed KA Scripts as checklists and thus made less omissions of changes, which was an interesting observation about why KA Scripts are useful.

3 Lessons Learned

For many user studies of KA tools, the time required from subjects and experimenters and the specific qualifications that subjects have to meet (e.g., domain experts, knowledge engineers) con-

stitute a practical limit in the number of subjects included in a study. This severely impacts the design of an experiment. The experiment has to be carefully designed to control as much as possible the variables that affect the outcome of the experiment because the number of subjects involved tends to be small. This variability is often very difficult to control. To help others avoid some of the pitfalls that we encountered in the design of our experiments, we list here some of the issues that we had to address:

- **User make mistakes that are outside the scope of the tool.** During the ETM experiments, subjects in both groups made (and then fixed) mistakes not related to the claims. The nature of these mistakes and the time that subjects spent fixing them varied. This was one of the factors that most severely affected the results of the experiments, in which some subjects spend more than half of their time interpreting and repairing their own mistakes. The following are some types of mistakes made by the subjects during the experiment that were not intended to be prevented by our KA tools:
  - Syntax errors: Subjects made syntax errors while entering new knowledge base elements using a text editor. Syntax errors were immediately detected by a parser included in both versions of the tool and were reported back to the users. Although some errors were more difficult to interpret than others, all of these were simple to repair. Neither the text editor or the parser were being tested in the experiment through the hypotheses.
  - Limited familiarity with the details of the domain model: Subjects got confused while entering complex knowledge base elements that made reference to other elements of the KB. For example, a subject referred to the HEIGHT of a RIVER instead of to the HEIGHT of the BANK of a RIVER. Most of these errors were immediately detected by a KB verification facility included in both versions of the tool. These errors were reasonably simple to repair.
  - Misunderstandings about the domain: Subjects performed a sequence of wrong KB modifications because they had misunderstood some specific aspects of the domain. Some of these errors were detected along the execution of the scenario when the subjects encountered clear contradictions that made them revise their interpretation of the domain. Some other errors were not detected until the subjects, believing that they had finished the assigned task, checked the KB with the provided sample problems and obtained wrong results. Detecting and repairing these errors was very difficult and sometimes required to undo some modifications made erroneously by the user.
  - Misunderstanding of the assigned KA task. Subjects performed a sequence of wrong modifications and/or omitted others because they had misunderstood the assigned task. Locating and repairing these errors was very difficult.
- **Differences in subject performance.** To minimize the impact in the results of the experiments of the difference in subjects performance we use within subject experiments (i.e., we asked each subject to perform two tasks, one with the extended version of a KA tool and the other with the basic tool). However, in one of the experiments the individual differences in performance among subjects was so pronounced that it was very hard to compare results across subjects. To make things worse, there were no apparent indicators that could let us predict in advance the performance of subjects in order to assign them better to the different groups. This situation forced us to compare only the difference in performance in using both tools for each subject. That is, to compare the performance of each subject in each one of the scenarios. For this reason, we try to define both scenarios with comparable

complexity, however this is not the only factor that influences the comparability of the data. The following are some of the characteristics that made subjects to perform differently: being extra cautious (e.g., they checked carefully each modification that they performed), exploring the tool's features during the experiment, critiquing the tool during the experiment, making decisions faster, typing faster or managing the text editor more skillfully.

- **Differences in understanding the domain, the assigned tasks, or the use of the KA tool.** Some subjects did not understand correctly the domain or the assigned task. This caused them to make mistakes or to stop in the middle of the experiment to clarify the instructions or the domain specifications. Other subjects did not understand some features of the KA tool and could not take advantage of them, which presumably made them take longer time to complete the tasks.

- **Different ways to solve the assigned tasks.** In some experiments, subjects solved the assigned tasks in different ways, hence the differences in performance are affected by the differences in the amount of work required to implement the different solutions. For example, some subjects defined few general KB elements that applied to several cases while others defined several specific KB elements that applied to few cases each; some subjects modified existing knowledge to handle new requirements while others added new knowledge to handle them; some subjects relied more in the tool's intrinsic inference capabilities while others preferred to state facts and procedures explicitly.

The design of experiments that control all of the above factors is very hard. Besides, there is always the possibility that other unforeseen factors also affect the outcome of the experiments. A practical alternative to enforcing stricter controls is to collect very fine grained measurements throughout the execution of the experiment and based on these measurements analyze the results carefully. The collection of fine grained measurements has other advantages as well. The execution of a KA task involves the execution of several small activities. Table 3 lists some of the observed activities that subjects performed during the executions of the experiments. Usually, a KA tool supports only some of these activities. If we only take into account the subject overall performance we are also weighing some activities that are not related to our claims. In the future, we plan to isolate better the specific activities that our tools are intended to support.

We have learned from our experience that a careful experiment design can enhance the quality and utility of the collected fine grained measurements. For example, in some cases we did not record different activities while editing KB elements. The edition of a KB elements (with a text editor) was treated as a single KB modification action and we only recorded its initial and ending time. However, while editing a KB element and before closing the text editor, a subject might perform several activities which will not get individually recorded in our logs. For example, the subject might modify several different parts of that element, make a mistake and then fix it, and even spend some time deciding how to proceed with that modification. This problem was noticed when we were not able to isolate the time for modifying some aspects of the problem solving knowledge.

The following list summarizes some of the lessons that we have learned from conducting our experiments.

- Use within subject experiments. Each subject should perform two tasks, one with the tool being evaluated and the other with the ablated tool. This helps to compensate for differences in user performance.

- understanding the given KA task
- deciding how to proceed with the KA task (i.e., what to do next)
- browsing the KB to understand it
- browsing the KB to find something
- editing (to create or to modify) a KB element
- checking that a modification had the expected effects in the KB
- looking around for possible errors
  - browsing KB to check that it looks/behaves as expected (i.e., verification)
  - running problems (i.e., validation)
  - browsing through a problem-solving trace to check that it is ok
- understanding and deciding how to fix an error
- recovering from an error by undoing previous steps (to delete or to restore a KB element)
- "putting 2 and 2 together" (i.e., stepping back and thinking about what is going on in the system)
- using the tool
  - deciding which features in the tool to use (multiple features can support similar functions)
  - performing actions using selected features (edit/debug/browse..)
  - understanding what the tool is showing/suggesting

TABLE 3: Activities performed by subjects during a KA session. The KA tool being tested may only address a subset of these activities, and the experiment should be designed to collect measurements for those specific activities.

- Minimize the variables unrelated to the claims to be proven. For example, in one of our experiments the KA tool allowed users to perform the same modification through different mechanisms: using a text editor or a menu based interface. The multiplicity of mechanisms to perform a modification did not add any value to the experiment, however it introduced unnecessary variability that complicated the analysis of the results.
- Minimize the chances that subjects make mistakes unrelated to the claims. Do not introduce unnecessary complications to the KA tasks. One of our experiments required that the subjects used a command that is hard to understand. Since the tool to be evaluated did not provide any support to handle this command it would have been better to avoid the need for that that command in the experiment.
- Isolate as much as possible the KA activities and the data that are relevant to the hypotheses. It was a good decision for the evaluation of ETM to split the KA task in two parts: before and after the first KB modification because ETM is concerned with the latter. Discriminating these two parts helped to perform a more focused evaluation of ETM.
- Ensure that subjects understood the domain and the assigned KA task. In the EMeD experiments the subjects discussed with the experimenters their understanding of the assigned

task. These subjects had less problems in executing the assigned tasks than the subjects in the ETM experiments.

- Avoid the use of text editors. The use of a text editor in our experiments caused subjects to make syntax errors. The differences in the subject's skills with the text editor program also affected the results of the experiments. It also did not allow us to discriminate the fine grained activities performed by the subjects.

4 A note on Statistical Analysis

As we mentioned earlier, the cost and resources required by empirical controlled user studies of KA tools result in relatively small scale experiments. Given the small number of subjects and tasks involved, it does not seem appropriate to analyze the statistical significance of our results. Researchers in other areas concerned with evaluation do not seem to consider this a crucial issue in current evaluation work (Self, 1993; Olson & Moran, 1998; Rombach *et al.*, 1992). In any case, it is interesting to note that our results stand up to standard tests of statistical significance. For example, a t-test on the results reported in (Kim & Gil, 1999a) shows that they are significant at the 0.05 level with $t(2) = 7.03$, $p < .02$. Gathering data from more subjects within each group may be more reassuring than using these tests for validation.

5 Related Work on User Studies in Knowledge Acquisition and Other Fields

A few relevant evaluations of KA tools that have been conducted to date. We describe them in terms of the methodology that we have presented in this paper. The studies are summarized highlighting the hypotheses/claims that were tested, the kinds of tasks and subjects used, the experimental setup, the results reported, and any findings that were surprising.

The TAQL study (Yost, 1993) was done by Greg Yost as part of his PhD work at CMU. TAQL is a KA tool for SOAR. Yost evaluated the tool by itself and also evaluated its performance compared to some basic data that had been reported for two other KA tools (SALT and KNACK).

```
1) Evaluation of Taql
- Hypothesis: Taql has more breadth than other KA tools and still effective
- KA task: implement a new KB given a domain description
- KB domains: 10 puzzles + 9 Expert systems
- Underlying KR: production rules
- Users: Soar programmers, three subjects (including. Yost)
- Experimental setup:
    - each subject given a domain description (domain-oriented, not
      implementation specs) + (at most) 3 test cases
    - three rounds of evaluations, starting with simple domains
- Data collected:
    - times for task understanding, design, coding, debugging
    - bug information: how found, what error, when and how fixed
```

```
- Results reported:
    - encoding rate (minutes per Soar production) for each subject
        in each domain
    - average fix time for catchable and uncatchable errors pre and post tool
- Conclusions:
    - subjects reduced their encoding rates over time
        (i.e., programmed faster)
    - encoding rate did not slow down as task size increased

2) Comparing Taql, Knack, and Salt
 - Users:
    - one subject for each case
    - reimplementation of original system (Knack and Salt cases)
 - Results reported:
    - development time (hours) for Taql and for two tools (Knack and Salt)
        at their respective domains (time reported for reimplementation)
 - Conclusions:
    - Taql outperformed role-limiting KA tools (this was a surprise)
```

The TURVY study (Maulsby *et al.*, 1993) was conducted by David Maulsby and Allen Cypher at the Advanced Technologies Group at Apple. This experiment was more on the area of HCI rather than KA, but it is relevant here because it tests an approach to programming by demonstration that learns as a user performs simple tasks.

```
- Hypotheses:
   - H1: all users would employ same set of commands even if told nothing
           in advance about the instructions that Turvy understands,
           a table of predicted set of commands was compiled in advance
   - H2: users would end up communicating using Turvy's terms
   - H4: users would tech Turvy simple tasks easily and complex tasks with
           reasonable effort
- KA tasks and KB domains:
    - modify bibliography format (main tests)
    - file selection
    - graphical editing
- Underlying KR: none
- Users: non-programmers
- Experimental setup:
   - "Wizard of Oz" experiment (no real software, user interacts
       with facilitator)
   - several rounds, different types of subjects
       - on main task:
           - pre-pilot experiment
           - pilot experiment (4 users)
           - main experiment (8 subjects)
       - on other domains:
           - 3 subjects
           - 2 subjects
- Data collected:
```

```
   - videotapes, notes, interviews
- Results reported:
   - qualitative results mostly (their intention)
   - some quantitative results were obtained by post-analysis
- Conclusions:
   - Evidence for H1, H2, H4
   - Interesting findings: quiet vs talkative users
```

There are other experiments in the field of KA that are not directly relevant but are worth mentioning. The Sisyphus experiments (Linster, 1992; Schreiber & Birmingham, 1996; Shadbolt, 1998) show how different groups would compare their approaches for the same given task, but most approaches lacked a KA tool and no user evaluations were conducted. A very controversial experiment tested whether knowledge engineering models (such as KADS models) were useful to users (Corbridge *et al.*, 1995), but it tested knowledge elicitation through models and did not test any tools or systems. Other evaluations have tested the use and reuse of problem-solving methods, but they measure code reuse rather than how users benefit from KA tools (Runkel & Birmingham, 1995; Eriksson *et al.*, 1995).

Outside of KA, there are relevant studies in other disciplines. As we mentioned earlier, user evaluations are very uncommon in AI research. Most evaluations involve run-time behavior of AI software with no human in the loop. User studies are more common in software engineering, HCI, and intelligent tutoring systems.

In software engineering, empirical evaluations have been used for years to evaluate tools to support programmers (Basili *et al.*, 1986). In this field, many aspects and issues in the software development process have been under study including languages, development environments, reuse, quality, and software management (Rombach *et al.*, 1992). User studies are only of concern for a few of these topics. Interestingly, the kind of controlled methods that we report in this paper generally seem to be in the minority when it comes to evaluate software (Zelkowitz & Wallace, 1998). Many studies do not involve users, others analyze some historical data that may be available, and many collect observations and data as a software project unfolds without any particular control settings.

User studies in the field of HCI share many of the issues that arise in the evaluation of KA tools. An additional complication in evaluating interfaces is that they do not work in isolation, i.e., often times an interface can only be as good as the target system that the user ultimately operates on through the interface. On the other hand, many of the studies in this area can involve more users and settings, since the tasks tend to be simpler and the target users seem to be more numerous (e.g., users are not required to have domain expertise).

In intelligent tutoring systems, there are recognized tradeoffs regarding the merits and needs of different approaches to evaluation (Self, 1993). Although formal evaluations are generally preferred, their cost makes them often impractical. Informal studies tend to be more common and seem to be sufficiently informative in practice to many researchers to guide their work.

Our studies to date do not address thoroughly the evaluation of the product itself, i.e., the knowledge base that results from the knowledge acquisition process. Currently, we test that the final knowledge base has sufficient knowledge to solve the right problems and generating the right answers. Other metrics, such as measures of the quality of the knowledge base, are also important. There is relevant work along these lines in the expert systems area (Hayes-Roth *et al.*, 1983; Chi *et al.*, 1988) as well as in software engineering (Fenton & Pfleeger, 1997). Our studies to date do not assess either how our particular tools would improve the end-to-end process of developing a knowledge base, which

includes interviewing experts, building prototypes, maintaining the knowledge base, and improving system performance. There is relevant work in software engineering on evaluating the improvement to the overall software development process, including studies specific to expert systems as software (Rombach *et al.*, 1992).

6 Conclusions

We have presented a methodology for designing user evaluations of KA tools. We have been using it successfully in our own work to evaluate various approaches within the EXPECT framework. We have also discussed the lessons learned from our studies of two KA tools, and outlined some open issues. Our hope is that sharing our methodology and our experiences with the KA community we will contribute to making our field a more experimental and perhaps a more scientific one.

Acknowledgments

We would like to thank the researchers of the University of Southern California/Information Sciences Institute and the past and present members of the EXPECT research group that patiently participated in our experiments, making possible the evaluations of ETM and EMeD reported here. Many thanks to Paul Cohen, Keith Holyoak, and Kurt Van Lehn for their comments and advise on experimental design and analysis. We gratefully acknowledge the support of DARPA with contract DABT63-95-C-0059 as part of the DARPA/Rome Laboratory Planning Initiative and with grant F30602-97-1-0195 as part of the DARPA High Performance Knowledge Bases Program.

References

Barrett, A. & Weld, D. (1994). Partial-order planning: Evaluating possible efficiency gains. *Artificial Intelligence*, 67(1).

Basili, V., Selby, R. W., & Hutchens, D. H. (1986). Experimentation in software engineering. *IEEE Transactions in Software Engineering*, SE-12(7).

Birmingham, W. & Klinker, G. (1993). Knowledge-acquisition tools with explicit problem-solving methods. *The Knowledge Engineering Review*, 8(1):5–25.

Chi, M., Glaser, R., & Farr, M. (1988). *The Nature of Expertise*. Lawrence Erlbaum.

Cohen, P., Schrag, R., Jones, E., Pease, A., Lin, A., Starr, B., Gunning, D., & Burke, M. (1998). The darpa high-performance knowledge bases project. *AI Magazine*, 19(4).

Corbridge, C., Major, N. P., & Shadbolt, N. R. (1995). Models exposed: An empirical study. In *Proceedings of the Ninth Knowledge-Acquisition for Knowledge-Based Systems Workshop.*

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

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

Estlin, T. A. (1998). *Using Multi-Strategy Learning to Improve Planning Efficiency and Quality.* PhD thesis, University of Texas at Austin, Computer Science Department, Austin, TX.

Fenton, N. E. & Pfleeger, S. L. (1997). *Sofware Metrics: A Rigorous and Practical Approach.* Boston, MA, International Thomson Computer Press.

Gil, Y. (1991). A specification of process planning for PRODIGY. Technical Report CMU-CS-91-179, Computer Science Department, Carnegie-Mellon University.

Gil, Y. (1992). *Acquiring Domain Knowledge for Planning by Experimentation.* PhD thesis, Carnegie Mellon University, School of Computer Science, Pittsburgh, PA.

Gil, Y. (1994). Knowledge refinement in a reflective architecture. In *Proceedings of the Twelfth National Conference on Artificial Intelligence.*

Gil, Y. & Linster, M. (1995). Dimensions to analyze applications. In *Proceedings of the Ninth Knowledge-Acquisition for Knowledge-Based Systems Workshop*, Banff, Alberta, Canada.

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.*

Gil, Y. & Swartout, B. (1994). EXPECT: A Reflective Architecture for Knowledge Acquisition. In *Proceedings of the 1994 Workshop of the ARPA-Rome Laboratory Knowledge-Based Planning and Scheduling Initiative (ARPI-94)*, Tucson, AZ.

Gil, Y. & Tallis, M. (1997). A script-based approach to modifying knowledge-based systems. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, Providence, RI.

Hayes-Roth, F., Waterman, D. A., & Lenat, D. B. (1983). *Building Expert Systems.* Reading, MA, Addison-Wesley Publishing Company.

Joseph, R. L. (1992). *Knowledge Acquisition for Visually Oriented Planning.* PhD thesis, Carnegie Mellon University, School of Computer Science, Pittsburgh, PA.

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

Kim, J. & Gil, Y. (1999b). User studies of an interdependency-based interface for acquiring problem-solving knowledge. Submited for publication.

Kim, J. & Rosenbloom, P. S. (1996). Learning efficient rules by maintaining the explanation structure. In *Proceedings of the Thirteenth National conference on Artificial Intelligence*, pp. 763–770.

Linster, M. (1992). Special issue on sisyphus: Models of problem solving. *Int. Journal of Human-Computer Studies*, 40(2).

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

Maulsby, D., Greenberg, S., & Mander, R. (1993). Prototyping an intelligent agent through wizard of oz. In *INTERCHI-93.*

McDermott, J. (1998). 1998 aips planning competition. ftp.cs.yale.edu/pub/mcdermott/aipscomp-results.html.

Murray, K. S. (1995). *Learning as Knowledge Integration.* PhD thesis, University of Texas at Austin, Computer Science Department, Austin, TX.

Olson, G. M. & Moran, T. P. (1998). Special issue on experimental comparisions of usability evaluation methods. *Human-Computer Interaction*, 13.

Pérez, A. M. & Carbonell, J. G. (1994). Control knowledge to improve plan quality. In *Proceedings of the Second International Conference on AI Planning Systems.*

Quinlan, J. R. (1993). *C4.5: Programs for machine learning.* San Mateo, CA, Morgan Kaufmann.

Rombach, H. D., Basili, V. R., & Selby, R. W., (Eds.) (1992). *Proceedings of the International Workshop on Experimental Software Engineering Issues: Critical Assessment and Future Directions.*, Lecture Notes in Computer Science series, 1993, Dagstuhl Castle, Germany. Springer Verlag.

Runkel, J. T. & Birmingham, W. P. (1995). Knowledge acquisition in the small: Building knowledge-acquisition tools from pieces. *Knowledge Acquisition*, 5(2):221–243.

Schreiber, A. T. & Birmingham, W. P. (1996). The sisyphus-vt initiative. *International Journal of Human-Computer Studies*, 44(3/4).

Self, J. (1993). Special issue on evaluation. *Journal of Artificial Intelligence in Education*, 4(2/3).

Shadbolt, N. (1998). Sisyphus-iii. http://www.psyc.nott.ac.uk/aigr/research/ka/SisIII.

Swartout, W. & Gil, Y. (1995). EXPECT: Explicit representations for flexible acquisition. In *Proceedings of the Ninth Knowledge-Acquisition for Knowledge-Based Systems Workshop.*

Tallis, M. (1999). *A Script-Based Approach to Modifying Knowledge-Based Systems.* PhD thesis, University of Southern California, Computer Science Department, Los Angeles, CA.

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

Tambe, M. & Rosenbloom, P. S. (1990). A framework for investigating production system formulations with polynomially bounded match. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pp. 693–700.

Veloso, M. & Blythe, J. (1994). Linkability: Examining causal link commitments in partial-order planning. In *Proceedings of the Second International Conference on AI Planning Systems.*

Webb, G. I., Wells, J., & Zheng, Z. (1999). An experimental evaluation of integrating machine learning with knowledge acquisition. *Machine Learning*, 35:5–23.

Yost, G. R. (1992). *TAQL: A Problem Space Tool for Expert System Development.* PhD thesis, Carnegie Mellon University, School of Computer Science, Pittsburgh, PA.

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

Zelkowitz, M. V. & Wallace, D. (1998). Experimental models for validating computer technology. *IEEE Computer.*

Appendix A: Overview of the Two Evaluated KA Tools

This appendix describes the two KA tools that we refer in the paper.

ETM (EXPECT Transaction Manager)

The script-based knowledge acquisition (SBKA) approach (Gil & Tallis, 1997; Tallis, 1999) was conceived to support users in completing a KBS modification. KBS modifications usually require changing several related parts of a system. Identifying all of the related portions of the system that need to be changed and determining how to change them is hard for users to figure out. Furthermore, if the modification is not completed, the KBS will be left inconsistent.

To assist users in performing all of the required changes, a KA tool needs to understand how changes in different parts of the system are related. In script-based knowledge-acquisition this is achieved by incorporating a library of *knowledge-acquisition scripts*, which represent prototypical procedures for modifying knowledge-based systems. KA scripts provide a context for relating individual changes of different parts of a KBS, and hence enabling the analysis of each change from the perspective of the overall modification. ETM is our implementation of a script-based KA tool and is integrated to the EXPECT framework for developing knowledge-based systems.

EMeD (EXPECT Method Developer)

An approach that has been very effective to develop tools that acquire knowledge from users is to use *expectations* of what users have to add or may want to add next (Eriksson *et al.*, 1995; Birmingham & Klinker, 1993; Marcus & McDermott, 1989; Davis, 1979). Most of these expectations are derived from the *inter-dependencies* among the components in a knowledge-base system (KBS). For example, EXPECT (Gil & Melz, 1996; Swartout & Gil, 1995) uses dependencies between factual knowledge and problem-solving methods to find related pieces of knowledge in their KBS and create expectations from them.

These tools can successfully build expectations because there is already a body of knowledge where the new knowledge added by the user must fit in. In the configuration example, there would be problem solving knowledge about how to solve configuration tasks (how to describe a configuration, what is a constraint, what is the relation between a constraint and a fix, how to apply a fix, etc.) However, when a new knowledge base (KB) is created (or when an existing one is significantly extended) there is little or no pre-existing knowledge in the system to draw from.

EMeD(Kim & Gil, 1999a; Kim & Gil, 1999b) is a tool to guide users in adding new problem-solving knowledge KB creation during KB extension where there is little or no pre-existing knowledge in the system to draw inter-dependencies. We have classified the sources of errors in the KB development process based on their characteristics, and were able to detect several sources for such expectations. The expectations result from enforcing constraints in the knowledge representation system, looking for missing pieces of knowledge in the KB, and working out incrementally the inter-dependencies among the different components of the KB. In this paper, we call the KA with these additional expectations as *expectation-based KA*. As the user defines new *KB elements* (i.e., new concepts, new relations, new problem-solving knowledge), the KA tool can form increasingly more frequent and more reliable expectations.