User Studies of Knowledge Acquisition Tools: Methodology and Lessons Learned

Marcelo Tallis[a,1], **Jihie Kim**[b,2], **and Yolanda Gil**[b,3]

[a]Teknowledge Corporation
4640 Admiralty Way, Suite 231
Marina del Rey, CA 90292, USA
(310)578-5350

[b]Information Sciences Institute
University of Southern California
4676 Admiralty Way, Marina del Rey, CA
90292, USA
(310) 822-1510

[1]Corresponding author. E-mail: tallis@teknowledge.com.
[2]E-mail: jihie@isi.edu.
[3]E-mail: gil@isi.edu.

Abstract

Knowledge acquisition research concerned with the development of knowledge acquisition tools is in need of a methodological approach to evaluation. This paper describes our experimental methodology to conduct studies and experiments of users modifying knowledge bases with knowledge acquisition tools. We also report the lessons learned from several experiments that we have performed using this methodology. Our hope is that it will help others design user evaluations of knowledge acquisition tools. We discuss our ideas for improving our current methodology and some open issues that remain.

1 Introduction

The field of Artificial Intelligence has increasingly recognized throughout the years the need and the value of being an experimental science. Some subfields 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 (Blake & Merz, 1998) 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 (KB) 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. 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, 2000).

Knowledge Acquisition (KA) research has a traditional focus on even more knowledge-intensive problems. Different systems use a wide variety of representations and are often built to address different aspects of knowledge base reasoning as well as to acquire different kinds of knowledge. 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, 1994; Schreiber & Birmingham, 1996; Shadbolt *et al.*, 1999), 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).

As developers of knowledge acquisition tools we wanted to evaluate our approach, 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 Artificial Intelligence 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 not as thorough or conclusive as we would like to see as scientists, yet these evaluations are very valuable 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.

The lack of evaluation in knowledge acquisition research is unfortunate, but could be due to a variety of reasons. First, user evaluations are very costly. In areas like machine learning and planning, experiments often amount to running programs repeatedly on already existing test sets. The evaluation of a 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. 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 *et al.*, 1999). 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. KA tool developers may have conducted usability studies, but the results are not reported in the literature. Third, unless human experiments are carefully designed and conducted, it is hard to draw conclusive results from the data.

Over the last few years, we have performed a series of user 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 evaluations (Gil & Tallis, 1997; Tallis & Gil, 1999; Tallis, 1999; Kim & Gil, 1999; Kim & Gil, 2000b; Kim & Gil, 2000a). 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. This paper describes our experiments in enough detail to illustrate the main points of our methodology. A more comprehensive description of our experiments and their results can be found in the above references.

The paper describes our experiences based on tests with two particular KA tools that we developed for EXPECT (Gil & Melz, 1996; Gil, 1994; Swartout & Gil, 1995). EXPECT is a framework for developing and modifying knowledge based systems (KBSs) whose main purpose is to enable domain experts lacking computer science or artificial intelligence background to directly manipulate the elements of a KB without the mediation of a knowledge engineering. The two tools that were the subject of our evaluation were intended to enhance some aspect of the EXPECT support to end users. ETM (EXPECT Transaction manager) (Tallis & Gil, 1999; Tallis, 1999; Gil & Tallis, 1997) uses typical KB modification sequences (KA Scripts) to help users make complex changes that require many steps to modify KBs. EMeD (EXPECT Method Developer) (Kim & Gil, 1999; Kim & Gil, 2000b; Kim & Gil, 2000a) analyzes and exploits interdependencies among KB elements to create expectations about how new knowledge fits and detect missing knowledge that needs to be acquired from users. Each tool was developed to investigate a different approach to guide users in knowledge acquisition tasks. The approaches are complementary, and we have recently integrated 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 (Blythe *et al.*, 2001). A brief overview of both tools can be found in Appendix A. Please note that the focus of this paper is not on the details of these tools but on our experimental methodology and the lessons learned from our experiments.

The paper begins by describing the methodology that we follow 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 our initial experiments, and describes open issues in KA experiment design. Finally, we discuss related work in KA and 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 questions that it will help answer or understand. In our case, these are stated as claims and hypotheses about our tools. The hypotheses to be tested determine what are the KA tasks to be performed by users, 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.

[Insert table 1 here]

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.

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 is often not possible to test a broad claim, but formulating 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.

Note that many experiments are designed to explore how something works, without any specific hypotheses or claims. For example, several alternative interface designs can be evaluated with users to find out which designs are more suitable, perhaps without any prior hypothesis about which features are best.

The first step in the design of our evaluations is to state the main claims regarding our KA tools. We ended up formulating similar claims for both ETM and EMeD:

1. Users will be able to complete KA tasks in **less time** using the KA tools.

   Rationale: Our KA tools would support some time consuming activities involved in KB modification tasks. For example, they support the analysis of the interactions among the elements of KB and the choice of actions to remove inconsistencies in the KB.

2. Users will make **less mistakes** during a KA task using the KA tools.

   Rationale: Our KA tools detect missing knowledge and inconsistencies in the KB and they also support users in fixing them.

3. The reduction in completion time and number of mistakes will be more noticeable for **less experienced users**.

   Rationale: Less experienced users will be the most benefited from the tool's thorough guidance in making changes to a KB. The tools will also be able to resolve the inconsistencies that arise during the modification of the KB using strategies that may be unknown to less experienced users but are well-known to more experienced ones.

4. The reduction in time will also be more noticeable for **users lacking a detailed knowledge** of the KBS implementation.

   Rationale: Our tools detect interactions with already existing knowledge. Our tools also reveal the existence of KB elements that can be reused or adapted and that the users may not be aware of. This should be particularly noticeable when the KB is large.

5. The KA tools will be useful for a **broad range of domains and knowledge acquisition scenarios**.

   Rationale: Our tools are based in general domain-independent principles.

Given these claims, we were able to state specific and measurable hypotheses to be proved or disproved with experiments that were feasible given 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 in combination with the EXPECT basic KA tool than for subjects using the EXPECT basic KA tool 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 useful to 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 current state-of-the-art 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 had 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 for the first hypothesis and a very different experiment for the second hypothesis.

In practice, many hypotheses are hard to evaluate because they imply experiments that may be unfeasible due to lack of time and other resources.

In order to show the benefits of a tool or technology, a useful way to design an experiment is to perform a comparison with some baseline tool. In this case, we have to choose carefully the baseline tool so that the only difference between the two tools is the presence or absence of the technology to be evaluated. Otherwise, we may not be able to determine if the differences in performance were due to the technology itself or to some other factors (e.g., a different interface design or interaction style). Comparing the performance of users using a KA tool with users using an editor to enter knowledge is only useful if the hypothesis is that using a KA tool is better than not using it at all, which is normally not a hypothesis that one questions in an experiment.

We often use tool *ablation* experiments, where the baseline tool results from eliminating some capability of the KA tool. For example, to test the benefits of additional expectations computed by EMeD, we compared EMeD against a basic KA tool that consisted of the same EMeD interface where a number of expectation features had been disabled. That is, both tools provided a similar user interface environment. We have also used knowledge ablation experiments, where the ablated KA tool has only access to a subset of the original knowledge base. The group of subjects that is given the ablated tool serves as the *control group*. We have found these experiments to be the most useful and compelling kind to test claims about KA approaches.

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.

A central concern is minimizing the effects of individual differences(Calfee, 1975). One 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. We typically find that the number of subjects tends to be small. Thus, 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). This kind of design 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.

In some experiments we have used several groups of users, each group of users having different background and skills, particularly with respect to their familiarity and expertise with knowledge base development and knowledge acquisition tools and techniques.

2.3.2 KA Tasks and Scenarios

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. Adding problem-solving knowledge is a very different task from adding instances, even if they are both KA tasks. It is important to design the experiments 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.

An important issue in within subject experiments is that if one were to give a subject the same exact KA task to do with the two versions of the tool there would most probably be a *transfer effect*. This means that the user would 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.

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 in itself. The knowledge bases that we used are available to other researchers by contacting the authors.

2.3.3 Experimental procedure

After determining the kind of experiment to be carried out, the hypotheses, the type of users, and the nature of the KA task, we can 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 (if any), how many tasks 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 task (e.g., the final KB correctly solves some given problems, perhaps according to a gold standard).

Our experiments followed the same general procedure distributed in three stages:

Stage 1: Familiarization

The subjects attend a presentation that introduced the knowledge bases and either ETM or EMeD.

Stage 2: Practice

The subjects 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 tool. The purpose of this practice is to make subjects familiar with the tools, the domain, and the procedure of the experiment.

Stage 3: Test

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

2. Answer a feedback questionnaire regarding their impressions and difficulties in using each version of the tool. Each question is given numerical range (1 to 5), so that the answers are comparable across subjects.

For each test scenario, the subjects start by analyzing the specifications. Then, they perform the given scenario until the system gives the correct results in the sample problem. During the test, the experimenter can only assist subjects in clarifying the instructions.

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. In these cases subjects are asked to complete a KA task and make sure that EXPECT does not report any errors. In some other cases, the subjects are asked to test the KBS with a given set of problems, and make sure they obtain the expected results. In addition, after each experiment, we check the modifications made 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*. 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. Voicing what they think and what they are doing seemed fine.

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

- Time to complete the KA task.
- Automatic log of changes to the KB (e.g., a new subgoal was added).

- Automatic log of errors in the KB after each change to the KBS (e.g., a problem solving goal cannot be achieved). These errors are detected automatically by the EXPECT framework and hence are available in both versions of the KA tool.

- Automatic log of the features of the KA tool used during the experiment (e.g., user follows a suggestion proposed by the tool).

- 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. Subjects often 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

A careful design of what to collect can enhance the quality and utility of the collected measurements. In one experiment we treated the edition of a KB element (with a text editor) as a single KB modification action and we only recorded its begin and end time. However, while editing a KB element and before closing the text editor, a subject might perform several editions which did not get individually recorded in our logs. For example, the subject might modify several different parts of a KB element, make a mistake and then fix it, and then spend some time deciding how to proceed. This deficiency in our instrumentation precluded us from isolating the time incurred in modifying individual aspects of the problem solving knowledge.

[Insert table 2 here]

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 2 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.

Collecting fine grained data is very useful because it not only proves/disproves the hypotheses, but it also helps to explore the potential causes of certain experiment outcomes because there might be unforseen factors that affect the outcome. For example, the analysis of the sequence of changes performed to the KB revealed that some subjects spent considerable time for irrelevant activities.

2.5 Analyzing the Data

This section lists the kinds of analysis we have performed on the data acquired.

2.5.1 Normalizing Variances

Our experience indicates that raw data needs to be carefully examined and sometimes be normalized in order to perform meaningful comparisons.

In some of our 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.

Depending on the hypotheses, to compensate for these differences we can average the time by the number of KB elements that have been created or modified during the task, instead of comparing the total time for completing the assigned KA task.

2.5.2 Depurating Data

Some other times the data relevant to the experiment have to be isolated or depurated because uncontrolled factors not related to the hypotheses distort the measurements. In our experiments with ETM, the merits the tool were more clearly evidenced after we divided the time it took to complete the assigned KA task in two parts: before and after the first KB modification. The purpose of ETM is to guide users in following up on the side effects of changes to the KB, hence the merits of ETM will not be evident until the subjects perform at least one change. The time it took for the different subjects to perform their first modification varied considerably, possible because this first modification included the time for understanding and planing the assigned KA task. Discriminating between the time before and after the first modification helped to focus the evaluation of ETM to the features that are more meaningful to our research.

[Insert figure 1 here]

Isolating the data relevant to our claims might require a detailed analysis of the gathered data. The following analysis of some data gathered in one of the ETM evaluations illustrates this step. Figure 1 compares the subjects performance at each step of an assigned KA task. A detailed analysis of this figure reveals some anomalies that affected the outcome of the experiment. For example, this figure shows that Subject 3 in the control group spent some extra amount of time analyzing incorrect KB answers and fixing inconsistencies caused by his/her own mistakes (7 minutes in Change 4 and 1 minute in Change 5). ETM was not designed to prevent or help users with this kind of mistakes. Recovering from these mistakes takes a significant amount of time. Hence, it seems unfair to compare the performance of subjects when some of them made mistakes that were not related to the use of ETM. It turns out that, even without considering this extra time the time spent by Subject 3 does not come close to the time of the subjects that used ETM. Thus, the data still helped validate our claim that subjects would take less time, but the difference in terms of the acquisition rates between both tools turned out to be less than if the users had not made these mistakes by chance.

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. 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 ETM:

- 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 the basic and the enhanced versions of the KA tools 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 among the KA tool features being evaluated.

- Misuse of the domain structures: 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.

- Misconceptions of the domain model: Subjects approached the KA task in a wrong way because they had misunderstood some aspects of the domain. For example, one subject wrongly believed that instance of SEAPORTS would point out to its LOCATION. However, this was not the relation that was represented in our model but its inverse (i.e., LOCATIONS pointed to its SEAPORTS instead of SEAPORTS to its LOCATIONS). Some of these errors were detected along the evolution of the KA task 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 and/or omission of requested changes. Subjects performed a sequence of wrong modifications because they had misunderstood the assigned task or oversaw some changes. Locating and repairing these errors was very difficult and sometimes required to undo some modifications made erroneously by the user.

In (Tallis, 1999) we present a detailed analysis of the cases that included mistakes and suggest 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.

2.5.3 Assessing the Evidence for or against Hypotheses and Claims

[Insert table 3 here]

Once the data has been collected (and possibly depurated) it has to be confronted with the hypotheses that had been formulated. For example, in one of our experiments involving EMeD we obtained the results described in Table 3 which allowed us to draw the conclusions described below. The first column of Table 3 shows the average time to complete tasks for each user group. We had (1) four knowledge engineers who had not used EMeD before but were familiar with EXPECT (2) two knowledge engineers not familiar with EXPECT, (3) four users not familiar with AI but had formal training in computer science, and (4) two users with no formal training in AI or CS. The results for different user groups are shown separately to contrast the results. The second column shows the average number of problem-solving methods added. The last column shows the average time to build one problem solving method, normalizing the variances as described above. The last row in the table summarizes the results.

Based on these analyses we were able to draw the following conclusions:

- **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.

We also find very useful to analyze the data in detail, looking for interesting and unexpected phenomena. For example, the results show that subjects needed to add slightly less KB elements with EMeD. We may use this additional finding to explore some other hypotheses in the future, such as the effect of EMeD on the quality of the output KBS. Also, we may investigate why the ratios did not improve as predicted for the less experienced users.

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, 1999) 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.

3 Lessons Learned

Our evaluations were done through a multiple iteration of the steps described in section 2. The first attempt to evaluate a KA tool have usually failed but provided very useful lessons in designing the subsequent experiments. The following list summarizes some of the lessons that we have learned from our experience. Although other KA experiments may have different goals and claims, we would like share our lessons so that they can avoid the same mistakes.

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

- Use ablation experiments. This helps to explain the benefits of additional features effectively and to provide compelling results.

- 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 two different mechanisms: a text editor or a menu based interface. Having both mechanisms to perform a modification did not add any value to the experiment, however it introduced unnecessary variability that complicated the analysis of the data.

- 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 the subjects to use a construct that is hard to understand yet subjects received little training for it. Since the tool to be evaluated did not provide any special support to handle this construct it would have been better to avoid the need for that command in the experiment.

- Isolate as much as possible the data that is relevant to the hypotheses. For example, we were able to evaluate the ETM hypotheses more effectively by focusing our analyses in the actions that followed the first modification performed to the KB. The purpose of ETM is to guide users in following up on the side effects of changes to the KB, hence the merits of ETM would not be evident until the subjects perform at least one change.

- Ensure that subjects understand the assigned KA task. We found that the best way to do this is to ask the subjects to repeat out loud what they understood they needed to do. These subjects had less problems in executing the assigned tasks than the subjects that simply nod and say they understand.

- 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.

- 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.

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

A few relevant user 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) tested 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, 1994; Schreiber & Birmingham, 1996; Shadbolt *et al.*, 1999) 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. 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). 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(Olson & Moran, 1998). 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).

5 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. By sharing our experiences with the KA community we hope to contribute to make this field more experimental and perhaps more scientific.

Acknowledgments

We would like to thank the past and present members of the EXPECT research group for their comments on this work. We also thank everyone that patiently participated in our experiments, making the evaluations reported here possible. 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

Basili, V., Selby, R. W., & Hutchens, D. H. (1986). Experimentation in software engineering. *IEEE Transactions in Software Engineering*, SE-12(7):733–743.

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

Blake, C. & Merz, C. (1998). UCI repository of machine learning databases.

Blythe, J., Kim, J., Ramachandran, S., & Gil, Y. (2001). An integrated environment for knowledge acquisition. In *Proceedings of the Intelligent User Interface Conference (IUI-2001)*, Sante Fe, NM, USA.

Calfee, R. C. (1975). *Human Experimental Psychology.* New York, Holt, Rinehart & Winston.

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

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 (KAW-95)*, pp. 13–1–13–21, Banff, Alberta, Canada.

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 (AAAI-94)*, pp. 520–526, Seattle, WA, USA.

Gil, Y. & Linster, M. (1995). Dimensions to analyze applications. In *Proceedings of the Ninth Knowledge-Acquisition for Knowledge-Based Systems Workshop (KAW-95)*, pp. 16–1 – 16–15, 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 (AAAI-96)*, pp. 469–476, Portland, OR, USA.

Gil, Y. & Tallis, M. (1997). A script-based approach to modifying knowledge-based systems. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, pp. 377–383, 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. (1999). Deriving expectations to guide knowledge base creation. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, pp. 235–241, Orlando, FL, USA.

Kim, J. & Gil, Y. (2000a). Acquiring problem-solving knowledge from end users: Putting interdependency models to the test. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*, pp. 223–229, Austin, TX, USA.

Kim, J. & Gil, Y. (2000b). User studies of an interdependency-based interface for acquiring problem-solving knowledge. In *Proceedings of the Intelligent User Interface Conference (IUI-2000)*, pp. 165–168, New Orleans, LA, USA.

Kim, J. & Rosenbloom, P. S. (1996). Learning efficient rules by maintaining the explanation structure. In *Proceedings of the Thirteenth National conference on Artificial Intelligence (AAAI-96)*, pp. 763–770, Portland, OR, USA.

Linster, M. (1994). Sisyphus '91/92: Models of problem solving. *Int. Journal of Human-Computer Studies*, 40(2):189–192.

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 *Proceedings of INTERCHI-93*, pp. 277–285.

McDermott, D. (2000). The 1998 ai planning systems competition.

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 (AIPS-94)*, pp. 323–328, Chicago, IL, USA.

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*, 44:275–280.

Self, J. (1993). Special issue on evaluation. *Journal of Artificial Intelligence in Education*, 4(2/3):129–413.

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

Swartout, W. & Gil, Y. (1995). EXPECT: Explicit representations for flexible acquisition. In *Proceedings of the Ninth Knowledge-Acquisition for Knowledge-Based Systems Workshop (KAW-95)*, pp. 34–1 – 34–19, Banff, Alberta, Canada.

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 (AAAI-99)*, pp. 242–249, Orlando, FL, USA.

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 (AAAI-90)*, pp. 693–700, Boston, MA, USA.

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*, pp. 735–744.

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 complex KBS modifications. KBS modifications usually require changing several related parts of the KB. 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.

[Insert figure 2 here]

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 KBSs. 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 KBSs. Figure 2 shows its interface.

EMeD (EXPECT Method Developer)

Successful approaches to developing knowledge acquisition tools use expectations of what the user has to add or may want to add, based on how new knowledge fits within a knowledge base that already exists (Davis, 1979; Marcus & McDermott, 1989; Birmingham & Klinker, 1993; Eriksson *et al.*, 1995). When a knowledge base is first created or undergoes significant extensions and changes, these tools cannot provide much support.

We performed an analysis of the KB creation task, investigating why creating a knowledge base is hard and what sources of expectations that KA tools can exploit in order to guide users.

- *Difficulty in designing and creating many KB elements* $\Rightarrow$ *Guide the users to avoid errors and letting them look up related KB elements.*

- *Many pieces of knowledge are missing at a given time:* $\Rightarrow$ *compute surface relationships among KB elements to find incomplete pieces and create expectations from them*

- *Difficulty in predicting what pieces of knowledge are related and how* $\Rightarrow$ *use surface relationships to find unused KB elements and propose potential uses of the elements*

- *Inconsistencies among newly defined KB elements* $\Rightarrow$ *help users find them early and propose fixes*

[Insert figure 3 here]

Through the analysis, we 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 interdependencies among the different components of the KB. 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. EMeD (Kim & Gil, 1999; Kim & Gil, 2000b; Kim & Gil, 2000a) is the tool we implemented to support users in adding problem-solving knowledge. Its interface is shown in Figure 3.

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 was learned 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 experimental design.

- 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 2: 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.

User Groups	Avg. time to complete tasks (min)	Avg. number of methods added	Time/Method (min)
Experienced EXPECT users:With ablated tool	54.50	6.25	8.72
Experienced EXPECT users:With EMeD	38.25	6.00	6.38
Experienced KBS users:With ablated tool	65.50	6.50	10.08
Experienced KBS users:With EMeD	39.00	5.00	7.80
Non-Experienced CS users:With ablated tool	88.00	5.50	16.00
Non-Experienced users:With EMeD	55.75	5.25	10.62
End users:With ablated tool	133.50	5.50	24.27
End users:With EMeD	102.00	6.50	15.69
All:With ablated tool	80.67	5.92	13.63
All:With EMeD	54.83	5.67	9.67

TABLE 3: EMeD Results.

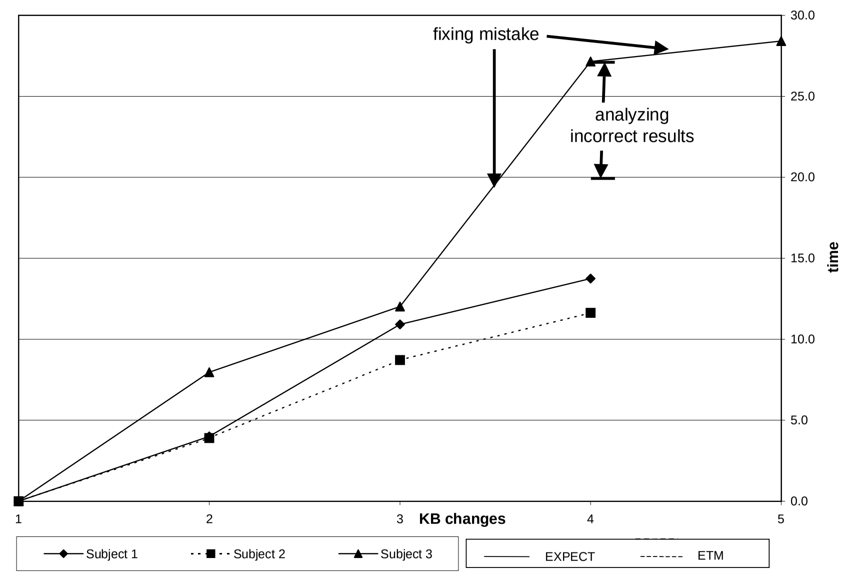
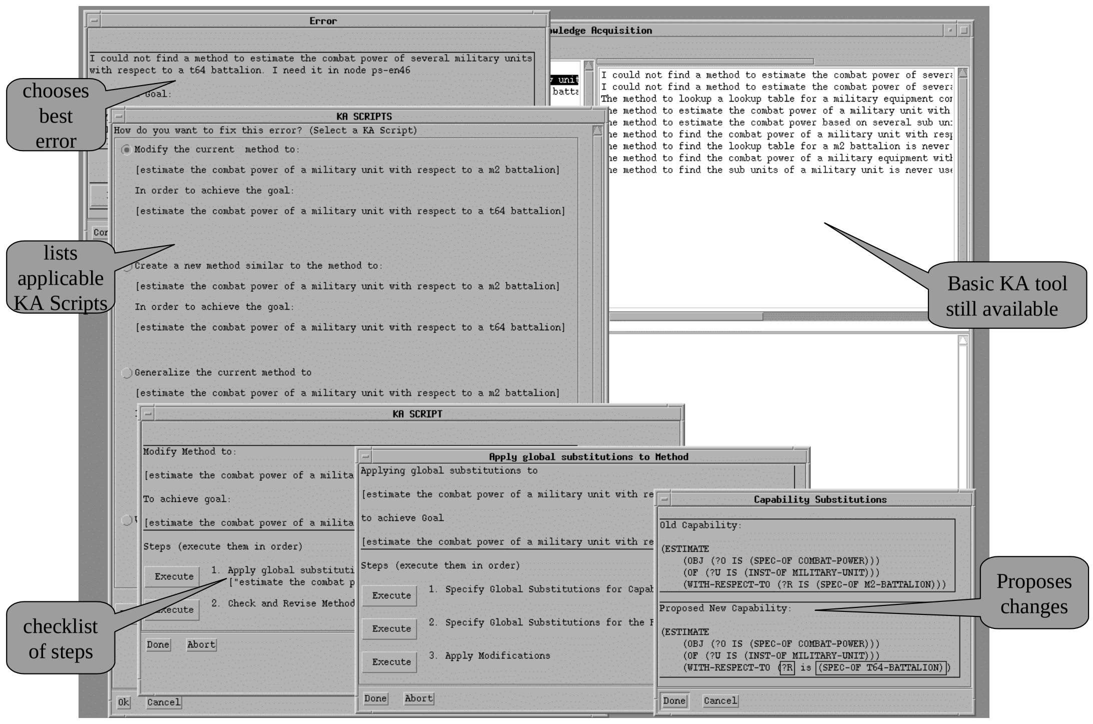


FIGURE 1: Time increments at each step for one of the ETM scenarios. Time increments are measured as the elapsed time between the finalization of two consecutive commands



I could not find a method to estimate the combat power of several military units
with respect to a t64 battalion. I need it in node ps-en46

How do you want to fix this error? (Select a KA Script)

Modify the current method to:
[estimate the combat power of a military unit with respect to a m2 battalion]
In order to achieve the goal:
[estimate the combat power of a military unit with respect to a t64 battalion]

Create a new method similar to the method to:
[estimate the combat power of a military unit with respect to a m2 battalion]
In order to achieve the goal:
[estimate the combat power of a military unit with respect to a t64 battalion]

Generalize the current method to
[estimate the combat power of a military unit with respect to a m2 battalion]

```
I could not find a method to estimate the combat power of severa
I could not find a method to estimate the combat power of severa
The method to lookup a lookup table for a military equipment cor
he method to estimate the combat power of a military unit with
he method to estimate the combat power based on several sub un:
he method to find the combat power of a military unit with resp
he method to find the lookup table for a m2 battalion is never
he method to find the combat power of a military equipment witl
he method to find the sub units of a military unit is never use
```

Modify Method to:
[estimate the combat power of a milita
To achieve goal:
[estimate the combat power of a milita
Steps (execute them in order)
1. Apply global substituti
["estimate the combat p
2. Check and Revise Method

Applying global substitutions to
[estimate the combat power of a military unit with re
to achieve Goal
[estimate the combat power of a military unit with re
Steps (execute them in order)
1. Specify Global Substitutions for Capab
2. Specify Global Substitutions for the F
3. Apply Modifications

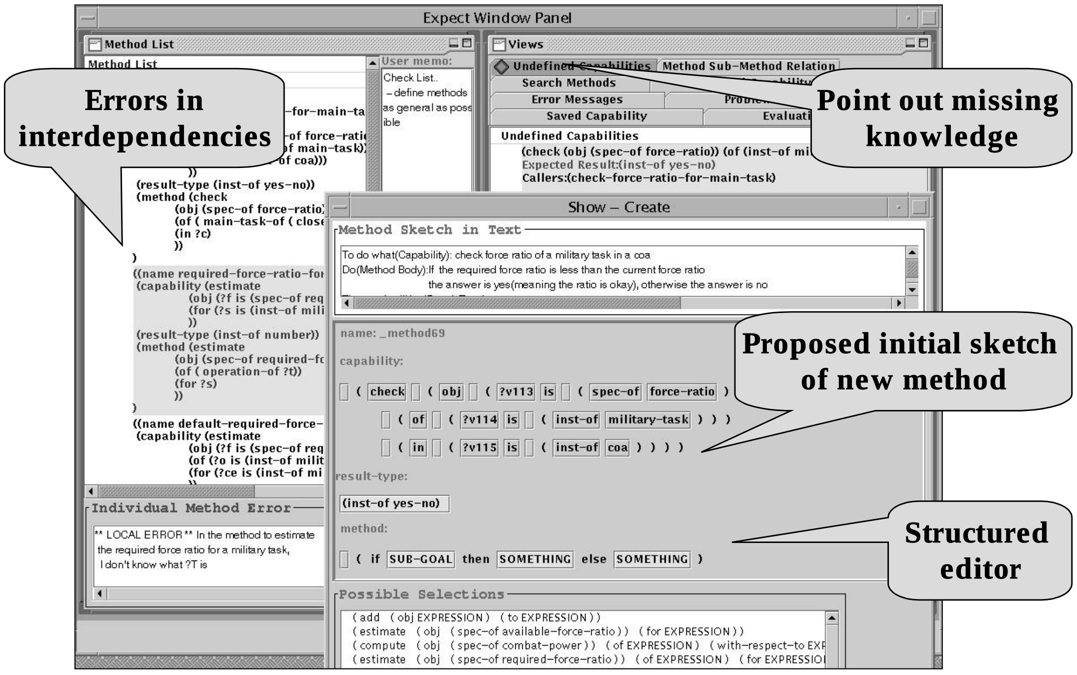
Old Capability:

```
(ESTIMATE
    (OBJ (?O IS (SPEC-OF COMBAT-POWER)))
    (OF (?U IS (INST-OF MILITARY-UNIT)))
    (WITH-RESPECT-TO (?R IS (SPEC-OF M2-BATTALION)))
```

Proposed New Capability:

```
(ESTIMATE
    (OBJ (?O IS (SPEC-OF COMBAT-POWER)))
    (OF (?U IS (INST-OF MILITARY-UNIT)))
    (WITH-RESPECT-TO (?R is (SPEC-OF T64-BATTALION)))
```

FIGURE 2: ETM User Interface.



User memo:
Check List..
- define methods as general as possible

Undefined Capabilities
(check (obj (spec-of force-ratio)) (of (inst-of mi
Expected Result:(inst-of yes-no)
Callers:(check-force-ratio-for-main-task)

```
        ))
 (result-type (inst-of yes-no))
 (method (check
         (obj (spec-of force-ratio)
         (of ( main-task-of ( close
         (in ?c)
         ))
)
((name required-force-ratio-for
 (capability (estimate
             (obj (?f is (spec-of req
             (for (?s is (inst-of mili
             ))
 (result-type (inst-of number))
 (method (estimate
         (obj (spec-of required-fo
         (of ( operation-of ?t))
         (for ?s)
         ))
)
((name default-required-force-
 (capability (estimate
             (obj (?f is (spec-of req
             (of (?o is (inst-of milit
             (for (?ce is (inst-of mi
```

** LOCAL ERROR ** In the method to estimate the required force ratio for a military task, I don't know what ?T is

To do what(Capability): check force ratio of a military task in a coa
Do(Method Body):If the required force ratio is less than the current force ratio
the answer is yes(meaning the ratio is okay), otherwise the answer is no

```
name: _method69
capability:
( check ( obj ( ?v113 is ( spec-of force-ratio )
    ( of ( ?v114 is ( inst-of military-task ) ) )
    ( in ( ?v115 is ( inst-of coa ) ) ) )
result-type:
(inst-of yes-no)
method:
( if SUB-GOAL then SOMETHING else SOMETHING )
```

```
( add ( obj EXPRESSION ) ( to EXPRESSION ) )
( estimate ( obj ( spec-of available-force-ratio ) ) ( for EXPRESSION ) )
( compute ( obj ( spec-of combat-power ) ) ( of EXPRESSION ) ( with-respect-to EXP
( estimate ( obj ( spec-of required-force-ratio ) ) ( of EXPRESSION ) ( for EXPRESSIO
```

FIGURE 3: EMeD User Interface.