



```

((name method1)
 (capability (check (obj (?f is (spec-of force-ratio)))
  (of (?t is (spec-of main-task)))
  (in (?c is (inst-of COA))))))
(result-type (inst-of yes-no))
(method (check (obj (spec-of force-ratio))
  (of (main-task-of (close-statement-of ?c))))))

((name method2)
 (capability (check (obj (spec-of (force-ratio)))
  (of (?t is (inst-of military-task))))))
(result-type (inst-of yes-no))
(method (is-less-or-equal
  (obj (estimate (obj (spec-of required-force-ratio))
    (for ?t)))
  (than (estimate (obj (spec-of available-force-ratio))
    (for ?t))))))

((name method3)
 (capability (estimate (obj (?f is (spec-of required-force-ratio)))
  (for (?s is (inst-of military-task))))))
(result-type (inst-of number))
(method ...))

((name method4)
 (capability (estimate (obj (?f is (spec-of available-force-ratio)))
  (for (?t is (inst-of military-task))))))
(result-type (inst-of number))
(method ...))

```

Figure 1: Examples of EXPECT Problem-Solving Methods.

kinds of interfaces and interaction modalities would be appropriate and in what ways should they be different from those that knowledge engineers find useful?

This paper reports on a study to evaluate our KA tools with domain experts (end users) who extended a knowledge base in their area of expertise. This study was conducted as part of an evaluation of the DARPA High Performance Knowledge Bases program (Cohen *et al.* 1998). We also present our experimental design and the preliminary study with users with varying degrees of background in AI and computer science, which was performed before the evaluation. We analyze the results in terms of our initial questions and hypotheses, and extract some general conclusions that motivate future directions of KA research.

## EMeD: Exploiting Interdependency Models to Acquire Problem-Solving Knowledge

*EMeD* (EXPECT Method Developer) (Kim & Gil 1999) is a knowledge acquisition tool that allows users to specify problem-solving knowledge. This section summarizes the functionality of the tool, further details and comparison with other tools are provided in (Kim & Gil 1999).

EMeD is built within the EXPECT framework (Gil & Melz 1996; Swartout & Gil 1995). EXPECT's knowledge base contains ontologies that describe the objects in a domain, and problem-solving methods that describe how tasks are achieved. Tasks are specified as goal hierarchies, where a goal is broken into smaller subgoals all the way down to primitive or basic tasks. The problem-solving methods specify how the decomposition takes place. EXPECT provides a rich language that was developed with understandability and intelligibility in mind, since it was used to generate adequate explanations for knowledge-based systems

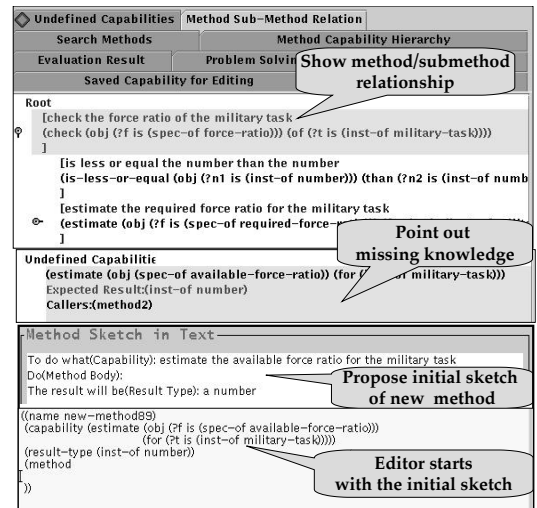


Figure 2: The Method Proposer of the EMeD Acquisition Interface.

(Swartout, Paris, & Moore 1991). Figure 1 shows some examples of EXPECT methods. Each problem-solving method has a *capability* that describes what the method can achieve, a *result type* that specifies the kind of result that the method will return upon invocation, and a *method body* that specifies the procedure to achieve the capability. The method body includes constructs for invoking subgoals to be resolved with other methods, retrieving values of concept roles, and control constructs such as conditional expressions and iteration. The arrows in the figure indicate some interdependencies, where a head of an arrow points to a sub-method which can solve a given subgoal. For example, the second method shown in the Figure 1 checks the force ratio of a given military task by comparing its required force ratio and the available force ratio. The result should be yes or no depending on whether the required ratio is less than the available ratio.

EXPECT derives an *Interdependency Model* (IM) by analyzing how individual components of a knowledge base are related and interact when they are used to solve a task. An example of interdependency between two methods is that one may be used by the other one to achieve a subgoal in its method body. Two methods can also be related because they have similar capabilities. EMeD exploits IM in three ways: (1) pointing out missing pieces at a given time; (2) predicting what pieces are related and how; (3) detecting inconsistencies among the definitions of the various elements in the knowledge base.

When users define a new problem-solving method, EMeD first finds the interdependencies and inconsistencies within that element, such as if any undefined variable is used in the body of the method. If there are any errors within a method definition, the *Local-Error Detector* displays the errors and it also highlights the incorrect definitions so that the user can be alerted promptly. The *Global-Error Detector* analyzes the knowledge base further and detects more subtle errors that occur in the context of problem solving.

By keeping the interdependencies among the problem-solving methods and factual knowledge, and analyzing interdependencies between each method and its sub-methods, the *Method Sub-method Analyzer* in EMeD can detect missing links and can find undefined problem-solving methods that need to be added. EMeD highlights those missing parts and proposes an initial version of the new methods, as shown in Figure 2. In this example, a method for checking the force ratio for an assigned task needs to compare the available force ratio (i.e., ratio between blue units and red units) with the force ratio required for that task. When the system is missing the knowledge for the available ratio (i.e., missing method4), the *Method Proposer* in EMeD notifies the user with a red diamond (a diamond shown in Figure 2 on the top) and displays the ones needed to be defined. It can also construct an initial sketch of the capability and the result type of the new method to be defined. What the new method has to do (capability of the method) is to estimate the available force ratio for a given military task. Since we are computing a ratio, the result type suggested is a number (method sketch in Figure 2). Users can search for existing methods that can achieve a given kind of capability using the *Method-Capability Hierarchy*, a hierarchy of method capabilities based on subsumption relations of their goal names and their parameters.

Finally, EMeD can propose how the methods can be put together. By using the Method Sub-method Analyzer for analyzing the interdependencies among the KB elements, it can detect still unused problem-solving methods and propose how they may be potentially used in the system.

## Experimental Design

As described in the introduction, current KA research lacks evaluation methodology. In recognition of the need for evaluation, the community started to design a set of standard task domains that different groups would implement and use to compare their work. These Sisyphus experiments (Schreiber & Birmingham 1996; Sisyphus 2000) 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. 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). Other KA work evaluated the tool itself. TAQL's performance was evaluated by comparing it with some basic data that had been reported for other KA tools (Yost 1993). There were some user studies on ontology editors (Terveen 1991). In contrast with our work, these evaluations were done with knowledge engineers. Also since the experiments were not controlled studies, the results could not be causally linked to the features in the tools.

Our research group has conducted some of the few user studies to date (Tallis & Gil 1999; Kim & Gil 1999), and as a result we have proposed a methodology (Tallis, Kim, & Gil 1999) that we use in our own work. It turns out that the lack of user studies is not uncommon in the software sciences (Zelkowitz & Wallace 1998). In developing a

methodology for evaluation of KA tools, we continue to draw from the experiences in other areas (Self 1993; Basili, Selby, & Hutchens 1986; Olson & Moran 1998).

Our goal was to test two main hypotheses, both concerned with Interdependency Models (IMs):

**Hypothesis I:** A KA tool that exploits IMs *enables users to make a wider range of changes* to a knowledge base because without the guidance provided with IMs users will be unable to understand how the new knowledge fits with the existing knowledge and complete the modification.

**Hypothesis II:** A KA tool that exploits IMs *enables users to enter knowledge faster* because it can use the IMs to point out to the user at any given time what additional knowledge still needs to be provided.

There are three important features of our experiment design:

- In order to collect data comparable across users and tasks, we used a controlled experiment. Thus, we designed modification tasks to be given to the participants based on typical tasks that we encountered ourselves as we developed the initial knowledge base.
- Given the hypotheses, we needed to collect and compare data about how users would perform these tasks under two conditions: with a tool that exploits IMs and with a tool that does not (this would be the control group). It is very important that the use of IMs be the only difference between both conditions. We designed an ablated version of EMeD that presented the same EMeD interface but did not provide any of the assistance based on IMs.
- Typically, there are severe resource constraints in terms of how many users are available to do the experiments (it typically takes several sessions over a period of days). In order to minimize the effect of individual differences given the small number of subjects, we performed within-subject experiments. Each subject performed two different but comparable sets of tasks (each involving the same kind of KA tasks but using a different part of the knowledge base), one with each version of the tool.

In order to determine when a KA task was completed, the subjects were asked to solve some problems and examine the output to make sure they obtained the expected results. In addition, after each experiment, we checked by hand the knowledge added by the subjects.

Participants were given different combinations of tools and tasks and in different order, so as to minimize transfer effects (i.e., where they would remember how they did something the second time around).

EMeD was instrumented to collect data about the user's performance, including actions in the interface (e.g., commands invoked and buttons selected), the knowledge base contents at each point in time, and the time at which each user action takes place. These provide objective measurements about task completion time and the use of specific features. Since this data was insufficient to understand what things users found hard and difficult to do with the

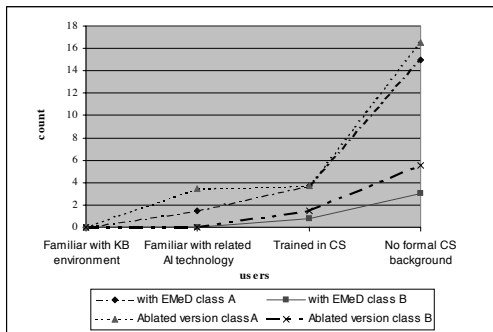


Figure 3: Average number of hints given to each group of subjects during the preliminary user study.

tool or why a certain action was not taken, we collected additional information during the experiment. We asked users to voice what they were thinking and what they were doing and recorded them in transcripts and in videotapes (during the experiments with domain experts). We also prepared a questionnaire to get their feedback, where instead of questions with free form answers we designed questions that could be answered with a grade from 1 (worst) to 5 (best).

### Preliminary Study

Since it is expensive to run user studies and hard to get domain experts in the field, we wanted to filter out distractions which are unrelated with our claim, such as problems with the tool that are not related to Interdependency Models. We also wanted to understand whether our interface and KA tool are appropriate for end users and how different types of users interact with it, so that we can improve our tools and our experimental methodology. For these reasons, we performed a preliminary study before the actual evaluation with domain experts.

The study used a spectrum of users that had gradually less background in AI and CS (Kim & Gil 2000). 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 but that had experience with knowledge-based systems, (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.

Since a major goal of this preliminary study was to understand our KA tool, we allowed the subjects to ask for hints when they were not able to make progress in the task (this was not allowed in the final evaluation). These hints allow us to categorize the basic types of difficulties experienced by users and adjust the tool based on them.

Figure 3 shows the number of hints given to the subjects in this study. More hints were always needed with the ablated version. The number of hints increases dramatically when subjects lack CS background. We analyzed all the hints, and separated them into two major categories. Class A hints consist of simple help on language and syntax, or clarification of the tasks given. Since syntax errors are unrelated to our claims about IMs, we developed a Structured Editor for the new version of EMeD (version 2) that

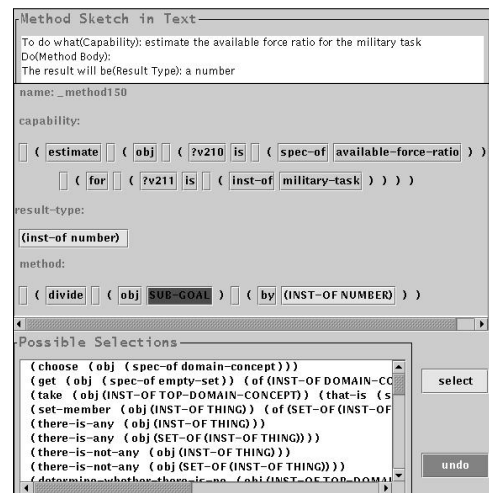


Figure 4: Structured Editor.

guides users to follow the correct syntax. Figure 4 shows the new editor which guides the users to follow the correct syntax. Users can build a method just using point and click operations without typing.

Class B hints were of a more serious nature. For example, users asked for help to compose goal descriptions, or to invoke a method with the appropriate parameters. Although the number of times these hints were given is smaller and the number is even smaller with EMeD, they suggest new functionality that future versions of EMeD should eventually provide to users. The subjects indicated that sometimes the tool was showing too many items, making it hard to read although they expected this would not be a problem after they had used the tool for a while and had become used to it. Since these presentation issues were affecting the results of the experiment and are not directly evaluating the IMs, the new version of EMeD (version 2) has more succinct views of some of the information, showing details only when the user asks for them. Other hints pointed out new ways to exploit IMs in order to guide users and would require more substantial extensions to EMeD that we did not add to the new version. One area of difficulty for subjects was expressing composite relations (e.g., given a military task, retrieve its assigned units and then retrieve the echelons of those assigned units). Although EMeD helped users in various ways to match goals and methods, in some cases the users still asked the experimenters for hints and could have benefited from additional help. The fundamental difficulties of goal composition and using relations still remained as questions for the real experiment.

In addition to improving the tool, we debugged and examined our experimental procedure, including tutorial, instrumentation, questionnaire, etc., especially based on the results from the fourth group.

We found out how much time end users would need to learn to use our tools. The tutorial given to the users was done with simpler sample tasks from the same knowledge base. The training time was significantly longer and harder

for the subjects with no technical background (2 hours for knowledge engineers and 7.5 hours for the project assistants). More details of this study are discussed in (Kim & Gil 2000), showing that even the end users were able to finish complex tasks, and that the KA tool saves more time as users have less technical background.

As described above, we extended our tool based on the pre-test results, creating a new version of EMeD (version 2). The next section describes the evaluation with domain experts with this new version of EMeD.

## Experiment with Domain Experts

The participants in this experiment were Army officers facilitated by the Army Battle Command Battle Lab (BCBL) at Ft Leavenworth, KS. They were asked to use our KA tools to extend a knowledge based system for critiquing military courses of action. Each subject participated in four half-day sessions over a period of two days. The first session was a tutorial of EXPECT and an overview of the COA critiquer. The second session was a tutorial of EMeD and a hands-on practice with EMeD and with the ablated version. In the third and fourth sessions we performed the experiment, where the subjects were asked to perform the modification tasks, in one session using EMeD and in the other using the ablated version. Only four subjects agreed to participate in our experiment, due to the time commitment required.

An important difference with the previous study is that during this experiment subjects were not allowed to ask for hints, only clarifications on the instructions provided. As soon as a participant would indicate that they could not figure out how to proceed, we would terminate that part of the experiment. In order to collect finer-grained data about how many tasks they could complete, we gave each subject four knowledge base modification tasks to do with each version of the KA tool. The reason is that if we gave them one single task and they completed almost but not all of it then we would not have any objective data concerning our two initial hypotheses. The four tasks were related, two of them were simpler and two more complex. The easier tasks required simple modifications to an existing method (e.g., generalize the existing methods that compute the required force ratio for “destroy” tasks into methods that can compute the ratio for any military tasks in general). The more complex tasks required adding new methods, such as the second method shown in Figure 1.

## Results and Discussion

The main results are shown in Figure 5. Figure 5-(a) shows the average time to complete tasks (for the completed tasks only). None of the subjects was able to do the more complex tasks with the ablated version of EMeD. Where data is available (the easier tasks), subjects were able to finish the tasks faster with EMeD. Figure 5-(b) shows the number of tasks that the subjects completed with EMeD and with the ablated version, both by task category and overall. The solid part of the bars show the number of tasks completed. We show with patterned bars the portion of the uncompleted tasks that was done when the subjects stopped and gave up

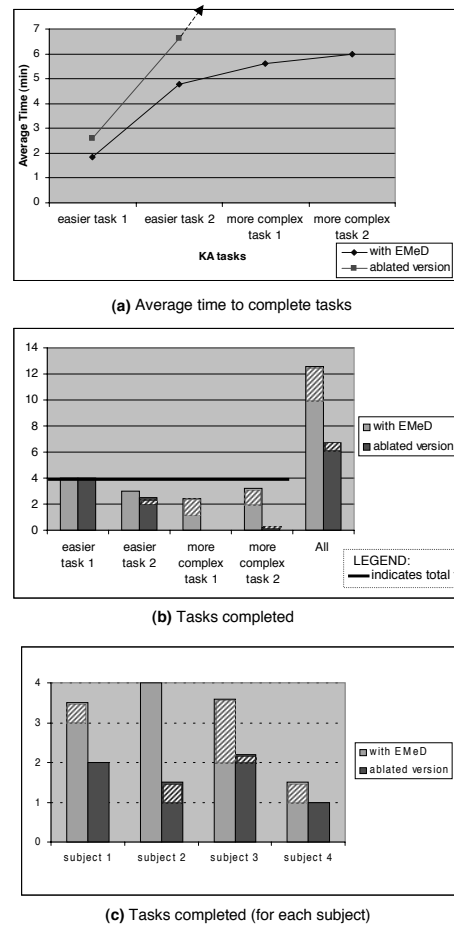


Figure 5: Results of the evaluation with domain experts.

(we estimated this based on the portion of the new knowledge that was added). Figure 5-(c) shows the same data but broken down by subject<sup>1</sup>. The results show that on average subjects were able to accomplish with EMeD almost twice as many tasks as they accomplished with the ablated version. The results support our claims that Interdependency Models can provide significant help to end users in extending knowledge bases.

It would be preferable to test additional subjects, but it is often hard for people (especially domain experts) to commit the time required to participate in this kind of study. Given the small number of subjects and tasks involved it does not seem appropriate to analyze the statistical significance of our results, although we have done so for some of the initial experiments with EMeD with a t-test showing that they were significant at the 0.05 level with  $t(2)=7.03$ ,  $p < .02$ . Gathering data from more subjects may be more reassuring than using these tests for validation.

Our experience with these experiments motivates us to share a few of the lessons that we have learned about knowl-

<sup>1</sup>We had noticed early on that Subject 4 had a different background from the other three, but unfortunately we were not able to get an alternative subject.

Functionality	Avg No. invocations	Usefulness rating	No. Users who used it
Method Proposer	10.5 (1.25)	4.7	4
Method Sub-method Analyzer	8.5	4.3	4
Method-Capability Hierarchy	2.75	4.5	2
Global-Error Detector	3	3.3	4

Table 1: Average Use of EMeD’s Functionality.

edge acquisition research:

- **Can end users use current KA tools to modify the problem-solving knowledge of a knowledge-based system? How much training do they need to start using such KA tools? Would they be able to understand and use a formal language?**

As we described earlier, we spent 8 hours (two half-day sessions) for training. They spent roughly half of that time learning EXPECT’s language and how to put the problem-solving methods together to solve problems. The rest of the time was spent learning about the KA tool and its ablated version. We believe that this time can be reduced by improving the tool’s interface and adding on-line help. We also recognize that more training may be needed if users are expected to make much more complex changes to the knowledge base. At the same time, if they did not need to be trained on how to use an ablated version of the tool they would not need to learn as many details as our subjects did.

Our subjects got used to the language and could quickly formulate new problem-solving methods correctly. They did not seem to have a problem using some of the complex aspects of our language, such as the control structures (e.g., if-then-else statement) and variables. It took several examples to learn to express procedural knowledge into methods and sub-methods and to solve problems. EMeD helps this process by automatically constructing sub-method sketches and showing the interdependencies among the methods. Retrieving role values through composite relations was also hard. Providing a better way to visualize and to find this kind of information would be very useful.

As a result of this experiment, we believe that with current technology it is possible to develop KA tools that enable end users to add relatively small amounts of new problem solving knowledge, and that they can be trained to do so in less than a day.

- **How much do Interdependency Models help? What additional features should be added to our KA tools?**

Overall, the Interdependency Models exploited via different features in EMeD were useful for performing KA tasks. Table 1 shows the average use of each of the Components of EMeD, in terms of the number of times the user invoked them<sup>2</sup>. The subjects were very enthusiastic about the tool’s capabilities, and on occasion would point

<sup>2</sup>We show the number of times the users selected them, except for the Method Proposer where we show the number of times the system showed it automatically as well as the number of times selected (in parenthesis) when applicable.

out how some of the features would have helped when they were using only the ablated version.

According to the answers to the questionnaire, using EMeD it was easier to see what pieces are interrelated. That is, visualizing super/sub method relations using Method Sub-method Analyzer was rated as useful (4.3/5). Also detecting missing knowledge and adding it was easier with EMeD’s hints. Highlighting missing problem-solving methods and creating initial sketch based on interdependencies (by Method Proposer) were found to be the most useful (4.7/5).

The Structured Editor used in this version of EMeD provided very useful guidance, and there were less errors for individual method definitions. The Local-Error Detector was not used for the given tasks.

- **What aspects of a modification task are more challenging to end users?**

Almost everyone could do simple modifications, which required that the subjects browse and understand the given methods to find one method to be modified and then changing it.

Some subjects had difficulties starting the KA tasks, when EMeD does not point to a particular element of the KB to start with. Although they could use the search capability in EMeD or look up related methods in the Method-Capability Hierarchy, this was more difficult for them than when the tool highlighted relevant information.

Typically, a KA task involves more than one step, and sometimes subjects are not sure if they are on the right track even if they have been making progress. A KA tool that keeps track of what they are doing in the context of the overall task and lets them know about their progress would be very helpful. Some of the research in using Knowledge Acquisition Scripts to keep track of how individual modifications contribute to complex changes (Tallis & Gil 1999) could be integrated with EMeD.

- **How do KA tools need to be different for different kinds of users**

We did not know whether end users would need a completely different interface altogether. It seems that a few improvements to the presentation in order to make the tool easier to use was all they needed. We did not expect that syntax errors would be so problematic, and developing a structured editor solved this problem easily. On the other hand, we were surprised that end users found some of the features useful when we had expected that they would cause confusion. For example, a feature in the original EMeD that we thought would be distracting and disabled is organizing problem-solving methods into a hierarchy. However, the feedback from the end users indicates that they would have found it useful.

Although EMeD is pro-active in providing guidance, we believe that some users would perform better if we used better visual cues or pop-up windows to show the guidance. As the users are more removed from the details,

the KA tool needs to do a better job at emphasizing and making them aware of what is important.

## Conclusions

In this paper, we presented an evaluation of a KA tool for acquiring problem-solving knowledge from end users who do not have programming skills. We described the experimental procedure we have designed to evaluate KA tools, and how we refined the design with a preliminary user study with users with gradually less background in AI and computer science. The KA tool that we tested exploits Interdependency Models, and the results show that it helped end users to enter more knowledge faster. We also discussed additional lessons that we have learned that should be useful to other knowledge acquisition researchers.

## Acknowledgments

We gratefully acknowledge the support of DARPA with grant F30602-97-1-0195 as part of the DARPA High Performance Knowledge Bases Program. We are indebted to the many subjects, especially the military officers from the Army Battle Command Battle Lab (BCBL) at Ft Leavenworth, KS, who participated in the experiments for their time and their patience. We would like to thank Surya Ramachandran, Marcelo Tallis, and Jim Blythe for their help during the experiment. We also would like to thank Jon Gratch for helpful comments on earlier drafts.

## References

- Bareiss, R.; Porter, B.; and Murray, K. 1989. Supporting start-to-finish development of knowledge bases. *Machine Learning* 4:259–283.
- Basili, V.; Selby, R. W.; and Hutchens, D. H. 1986. Experimentation in software engineering. *IEEE Transactions in Software Engineering* SE-12(7).
- Cohen, P.; Schrag, R.; Jones, E.; Pease, A.; Lin, A.; Starr, B.; Gunning, D.; and Burke, M. 1998. The DARPA High Performance Knowledge Bases Project. *AI Magazine* 19(4).
- Cypher, A. 1993. *Watch what I do: Programming by demonstration*. MIT Press.
- Eriksson, H.; Shahar, Y.; Tu, S. W.; Puerta, A. R.; and Musen, M. 1995. Task modeling with reusable problem-solving methods. *Artificial Intelligence* 79:293–326.
- Fikes, R.; Farquhar, A.; and Rice, J. 1997. Tools for assembling modular ontologies in Ontolingua. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, 436–441.
- Gil, Y., and Melz, E. 1996. Explicit representations of problem-solving strategies to support knowledge acquisition. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*.
- Kim, J., and Gil, Y. 1999. Deriving expectations to guide knowledge base creation. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, 235–241.
- Kim, J., and Gil, Y. 2000. User studies of an interdependency-based interface for acquiring problem-solving knowledge. In *Proceedings of the Intelligent User Interface Conference*, 165–168.
- Marcus, S., and McDermott, J. 1989. SALT: A knowledge acquisition language for propose-and-revise systems. *Artificial Intelligence* 39(1):1–37.
- Mitchell, T.; Mahadevan, S.; and Steinberg, L. 1985. LEAP: A learning apprentice for VLSI design. In *Proceedings of the 1985 International Joint Conference on Artificial Intelligence*.
- Olson, G. M., and Moran, T. P. 1998. Special issue on experimental comparisons of usability evaluation methods. *Human-Computer Interaction* 13.
- Runkel, J. T., and Birmingham, W. P. 1995. Knowledge acquisition in the small: Building knowledge-acquisition tools from pieces. *Knowledge Acquisition* 5(2):221–243.
- Schreiber, A. T., and 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).
- Sisyphus. 2000. Sisyphus projects. <http://ksi.cpsc.ucalgary.ca/KAW/Sisyphus/>.
- Swartout, W., and Gil, Y. 1995. EXPECT: Explicit representations for flexible acquisition. In *Proceedings of the Ninth Knowledge-Acquisition for Knowledge-Based Systems Workshop*.
- Swartout, W. R.; Paris, C. L.; and Moore, J. D. 1991. Design for explainable expert systems. *IEEE Expert* 6(3).
- Tallis, M., and Gil, Y. 1999. Designing scripts to guide users in modifying knowledge-based systems. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*.
- Tallis, M.; Kim, J.; and Gil, Y. 1999. User studies of knowledge acquisition tools: Methodology and lessons learned. In *Proceedings of the Twelfth Knowledge-Acquisition for Knowledge-Based Systems Workshop*.
- Terveen, L. 1991. *Person-Computer Cooperation Through Collaborative Manipulation*. Ph.D. Dissertation, University of Texas at Austin.
- Wielinga, B. J.; Schreiber, A. T.; and Breuker, A. 1992. KADS: a modelling approach to knowledge acquisition. *Knowledge Acquisition* 4(1):5–54.
- Yost, G. R. 1993. Knowledge acquisition in Soar. *IEEE Expert* 8(3):26–34.
- Zelkowitz, M., and Wallace, D. 1998. Experimental models for validating computer technology. *IEEE Computer* 31(5):23–31.