

Acquiring Procedural Knowledge in EXPECT

Yolanda Gil, Jim Blythe, Jihie Kim and Surya Ramachandran

University of Southern California / Information Sciences Institute

Marina del Rey, CA 90292

Phone +1 310 448 8794

Fax +1 310 822 0751

{gil,blythe,jihie,surya}@isi.edu

<http://www.isi.edu/expect>

Introduction

Most machine learning approaches for acquiring knowledge from users reduce the user's participation in the learning process to a bare minimum. Learning apprentices, for example, acquire knowledge by analyzing a user's *normal* interaction with a system. Some systems learn from solutions given by the user, while others display a proposed solution that the user can accept, correct, or rate. An advantage of these approaches is that they are non-intrusive, and users do not need to understand the underlying learning process beyond sometimes the fact that it is analyzing their examples. To learn under this limited style of interaction, the learning system must figure out on its own knowledge about what features of the example are important, what new abstract features (not appearing in the examples) need to be defined, what combinations of those features are relevant to the task, and what parts of the example correspond to the user's preferences in that specific case. This approach is impractical in complex tasks where there are many different features in an example, many alternative values of these features, and many combinations thereof can account for user behavior.

An alternative approach is represented by learners that can request information from the user when faced with a situation where they do not know how to proceed. A drawback of this approach is that it puts more burden on users in that they are required to determine what information the learner needs and to figure out what is the best way to provide it, often by designing good examples that facilitate learning. As a system acquires more knowledge about a task, information about which knowledge the system already has and how it is organized may be unknown to the user and often hard to keep track of, posing increasingly stronger requirements on the user.

Ideally, learning systems should place less demands on the user and become increasingly more competent at the learning task as their knowledge bases grow. They should become more responsible for their internal organization of knowledge and of their own learning process, perhaps by asking more specific questions to the user or by pointing out areas in their knowledge base where they detect the need for additional information. In order to achieve this, they must be able to reflect upon the knowledge that they have about the task—at each moment—and understand what ev-

ery piece of knowledge, old and new, contributes to the task. This understanding would allow them to be responsible for ensuring harmonious interactions between any new knowledge provided by the user and existing knowledge, thereby preventing redundancies, inconsistencies, and knowledge gaps that may arise inadvertently. This is the approach taken in our work on EXPECT (Swartout & Gil 1995). In this position paper we summarize selected areas of current research where we focus on end-users, and highlight some of the research questions that arise from them.

Exploiting Interdependency Models to acquire procedural knowledge

A theme of our KA research has been how KA tools can exploit *Interdependency Models* (Swartout & Gil 1995) that relate individual components of the knowledge base in order to develop expectations of what users need to add next. EMED (EXPECT Method Developer) (Kim & Gil 1999; 2000), a knowledge acquisition tool to acquire problem-solving knowledge, exploits the Interdependency Models to guide users by helping them understand the relationships among the individual elements in the knowledge base. Our hypothesis is that Interdependency Models allow users to enter more knowledge faster, particularly for end users.

EMED (Expect Method Developer) was developed to support users in adding new procedural knowledge to a knowledge base, either to modify existing knowledge base or to develop a new one. We analyzed the user tasks during knowledge base development and found several areas where an acquisition tool could help: (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. We then expanded Interdependency Models, developing a set of techniques and principles that could guide users in both knowledge base creation and modification. The expectations result from enforcing constraints in the knowledge representation system, working out incrementally the interdependencies among the different components of the KB. As the user defines new *KB elements* (e.g., new problem-solving knowledge), the KA tool can form increasingly more frequent and more reliable expectations.

When users define new procedural knowledge (problem-

solving methods), EMeD first finds the interdependencies and inconsistencies within that element, such as if any undefined variable is used in the body of the procedure. If there are any errors within a definition, the *Local-Error Detector* displays the errors and it also highlights the incorrect definitions in red 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-method, 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. When the system is missing the knowledge, the *Method Proposer* in EMeD notifies the user and displays the ones needed to be defined. It can also construct an initial sketch of the new method to be defined. 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.

Our evaluations show an average over thirty percent time savings for different user groups, including end users with no formal computer science training. The results also show that KA tool that exploits Interdependency Models helped end users enter more knowledge.

Exploiting background knowledge about tasks

End users who are not programmers are often unable to modify procedural knowledge directly, even with interdependency-based tools. Often they are unsure where to even begin when specifying how to perform some task and understanding a language for procedural information is also an obstacle. We address these issues by guiding the user through the process of adding problem-solving knowledge using background knowledge about a generic task type such as a problem-solving method library. We also use an English-based method editor so that users can modify task knowledge without having to see and understand the EXPECT programming language. Using the combination of these two approaches, we have been able to acquire procedural knowledge from end users who were not programmers that was directly executable (Blythe & Gil 2000).

We use background knowledge about a task area in several different ways to guide users adding new knowledge in that area. One way it is used is to identify which of a set of generic procedures might be appropriate to specialize for a given task. In this case, a tool can help a user to get started with a question-answer dialog that concentrates on features of the task to be performed rather than on programming constructs. Then it can recommend a generic KBS from a

library that goes some of the way to performing the task that the user wants to define. Here the required knowledge includes a set of generic procedures that are useful within a domain, information about appropriateness conditions for each one, about the information they need and about the relative strengths and weaknesses of alternative procedures. This information is in a form that can be operationalized as easily understandable questions for the user.

We also use background knowledge about a procedural task domain to help the user specialize a procedure once it has been selected. Some parts of a procedure should usually not be changed by a naive user since they define the way the generic task is performed. (These are often at the upper levels of a procedure definition.) Other parts are designed to be changed for a specific task, and are often represented as stubs. Knowledge about which parts of a task description can easily be changed and which should stay fixed can allow a tool to guide the user in modifying the procedure. One can also provide knowledge about the way that certain parts of the procedure are expected to be changed.

Notice that this is complementary to the Interdependency Model described in the last section. As a user modifies a generic task, a tool such as EMeD can help the user with information about missing pieces of the modified knowledge and about other parts that may need to be changed for consistency. Here, the background knowledge has solved the problem of how to start defining a task.

We built a problem-solving method library for plan evaluation and a KA tool called PSMTTool that uses background knowledge in the ways we have described to guide users building new KBs for plan evaluation tasks (Blythe & Gil 2000). In experiments in two different domains, end users were able to add new procedural knowledge for evaluating plans that could be directly executed. We found that users could complete on average 60% more tasks when using a full version of PSMTTool than when using an ablated version that did not use background knowledge. They completed tasks on average 3 times as quickly with the full version.

Editing methods with an English-based editor

We also use an English-language based editor that presents problem-solving methods in an English paraphrase. The editor allows the user to select parts of the method as phrases of English and automatically creates a set of probable new fragments that the part can be changed to. These are presented for the user to select from, also in an English paraphrase. Thus users to modify task knowledge without having to see and understand the EXPECT programming language. The power of this editor comes from the combination of hiding the underlying language and providing a browsable list of useful program elements and can be used to specify arbitrary procedural knowledge (Blythe & Ramachandran 1999).

Evaluations with End Users

We wanted to put our KA tools to the test and understand how accessible they are to end users. Could 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? We performed several suites of experiments with different kinds of users, including project assistants and Army officers (Kim & Gil 2000; Blythe & Gil 2000).

With EMeD, we spent 8 hours (two half-day sessions) for training. Users spent roughly half of that time learning EXPECT's language and how to put the procedural knowledge together. 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. 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 procedural knowledge 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 how to find this kind of information would be very useful.

In one experiment with PSMTTool, users worked with the tool after having taken the training for EMeD, with an extra 45 minutes of training on PSMTTool. In a second experiment users only saw PSMTTool and received one hour of training that showed the tool and how to put procedural knowledge together using the English-based editor. These users were able to effectively use the tool to define new tasks, indicating that procedural knowledge can be directly specified by users with very little training. Again, the more complex the task knowledge, the more training may be required.

For each tool we created an ablated version that lacks Interdependency Models in the case of EMeD or background knowledge in the case of PSMTTool. The differences in the number of tasks that users could complete are shown in Figure 1, both by task category and overall. Averaged across both tools, users could complete roughly twice as many tasks with the full versions as with the ablated versions.

Discussion

One important issue is to identify what aspects of a knowledge base modification task are more challenging to end users when using KA tools. Some subjects using EMeD had difficulties in starting the KA tasks, when it does not point to a particular element of the KB to start with. Although they could use the search capability or look up related methods in the tool, this was more difficult for them than the cases when the elements or problems to be resolved were already present in the KB and could be highlighted by the tool. PSMTTool is able to provide more help for starting a task, but only in cases where pre-specified generic task

knowledge can be appropriately modified.

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. Here PSMTTool is helpful because it keeps track of what users are doing in the context of the overall task and lets the user visualise their progress. KA tools that help with this in a broader range of situations would be very helpful.

Another important issue is whether KA tools need to be different for users with different degrees of background in computer science. We did not know whether end users would need a completely different interface from knowledge engineers. It seems that a few improvements to the presentation in order to make the tool easier to use was all they needed. The editors that help the users avoid syntax problems were important. In interviews we found that end users consistently preferred a version that used an English paraphrase, but we have not tested whether this leads to improved performance.

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 we thought would be distracting is organizing procedural knowledge into a hierarchy, but it seemed more useful for the end users than for knowledge engineers.

Although EMeD and PSMTTool are proactive 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.

References

- Blythe, J., and Gil, Y. 2000. Extending the role-limiting approach: Supporting end-users to acquire problem-solving knowledge. Technical report, Expect internal report.
- Blythe, J., and Ramachandran, S. 1999. Knowledge acquisition using an english-based method editor. In *Proc. Twelfth Knowledge Acquisition for Knowledge-Based Systems Workshop*.
- Kim, J., and Gil, Y. 1999. Deriving expectations to guide knowledge-base creation. In *Proc. Sixteenth National Conference on Artificial Intelligence*. AAAI Press.
- Kim, J., and Gil, Y. 2000. Acquiring problem-solving knowledge from end users: Putting interdependency models to the test. In *Proc. Seventeenth National Conference on Artificial Intelligence*. AAAI Press.
- Swartout, W. R., and Gil, Y. 1995. Expect: Explicit representations for flexible acquisition. In *Proc. Ninth Knowledge Acquisition for Knowledge-Based Systems Workshop*.

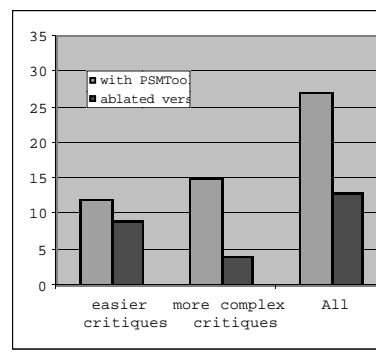
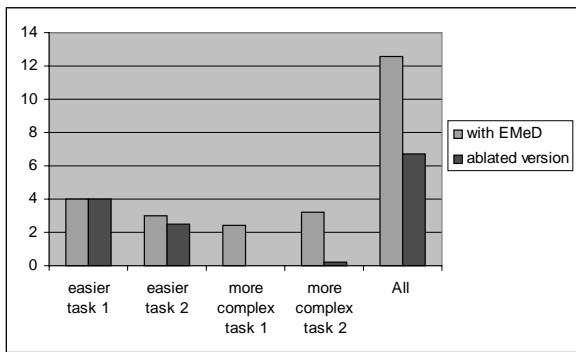


Figure 1: Number of tasks completed by users for EMeD (on the left) and PSMTTool, with the full and ablated versions of the tools.