# Modifying Knowledge Bases using Scripts

**Marcelo Tallis** and **Yolanda Gil**
Information Sciences Institute and Computer Science Department
University of Southern California
4676 Admiralty Way, Marina del Rey, CA 90292
{tallis, gil}@isi.edu

Most current Knowledge Acquisition (KA) approaches provide tools that are specifically tailored to a predefined library of problem-solving methods (Klinker *et al.* 1991; Puerta *et al.* 1992). These tools support users in populating knowledge bases (KBs) with domain-specific knowledge that the methods need. However, these tools do not support users in modifying the individual problem-solving methods beyond substituting one method for another one in the library. Hence, KA tools that support users in making a wider range of changes to problem-solving methods are needed.

We are currently developing EXPECT (Gil 1994), a tool that supports users in making a broad range of changes to knowledge-based systems (KBSs), including changes to problem-solving methods. EXPECT represents problem-solving methods explicitly, and its KA tool reasons about them to guide KA. Problem solving knowledge in EXPECT is structured in small-size methods. A method is a procedure for achieving a problem-solving goal. This procedure may include subgoals which are invocations to other methods. Methods goals have parameters that are bound to the values of the parameters of the subgoals that call them.

Modifications to problem-solving methods rarely consist of a single change. Instead, they require a sequence of related changes to different pieces of knowledge. Failing to perform the complete sequence may leave the KB in an incoherent state. For example, suppose the user wants to add a new parameter to a method. This modification requires several individual modifications to different methods: (1) adding the declaration of the new parameter; (2) using the new parameter in the method's procedure; (3) adding the new parameter to the subgoals in other methods that called the modified method; (4) modifying parameter declarations of the methods called by any modified subgoal of the method's procedure. The omission of any of these changes will probably leave the KB incoherent. For example, if the user declares a new parameter in a method (change 1) but does not change the subgoals of other methods that call the modified method (change 3), then the parameters of these subgoals will not match the parameters of the modified method and the subgoals will not be achieved.

Determining the sequence of changes required to complete a KB modification is a laborious and difficult task. Moreover, the identification and thorough execution of this sequence is necessary to maintain the integrity of the KB. Hence, it becomes essential that a KA tool assists the user in identifying and executing these modifications sequences.

We approach this problem by providing the tool with knowledge of commonly occurring sequences of changes which we call *KA scripts*. An example of a KA script, which corresponds to changes 1 and 3 of the example is: (1) Add the declaration of a new parameter to a method, (2) Add that new parameter to the subgoals of other methods that call the modified method. Other KA scripts take care of the other changes in the example. These KA scripts are used by a KA tool to help a user to resolve side-effects of changes already made and complete the modification that he or she has started.

We have identified 75 KA scripts by analyzing both our formal representation of problem-solving methods and example KA scenarios. Through this analysis we addressed issues like what information from the context of the modifications can be used to guide the user in completing KA scripts, and how KA scripts interact with one another. Then ten of these KA scripts were implemented and usability tests conducted with promising results.

## References

Gil, Y. 1994. Knowledge refinement in a reflective architecture. In *Proceedings of AAAI'94*.

Klinker, G.; Bhola, C.; Dallemagne, G.; Marques, D.; and McDermott, J. 1991. Usable and reusable programming constructs. *Knowledge Acquisition* 3(2):117–135.

Puerta, A.; Egar, J.; Tu, S.; and Musen, M. 1992. A multiple-method knowledge-acquisition shell for the automatic generation of knowledge-acquisition tools. *Knowledge Acquisition* 4(2):171–196.