In *Proceedings of the 14th International Conference on Knowledge Engineering and Knowledge Management,* EKAW 2004, 5-8th October 2004 - Whittlebury Hall, Northamptonshire, UK, Springer-Verlag, 2004.

Parallel Knowledge Base Development by Subject Matter Experts

Gheorghe Tecuci, Mihai Boicu, Dorin Marcu,
Bogdan Stanescu, Cristina Boicu, Marcel Barbulescu

MSN 4A5, Learning Agents Center, George Mason University,
4400 University Drive, Fairfax, VA 22030, USA
{tecuci, mboicu, dmarcu, bstanesc, ccascava}@gmu.edu,
mbarb@cs.gmu.edu, http://lac.gmu.edu,
tel: 1 703 993-1722, fax: 1 703 993-1710

Abstract. This paper presents an experiment of parallel knowledge base development by subject matter experts, performed as part of the DARPA's Rapid Knowledge Formation Program. It introduces the Disciple-RKF development environment used in this experiment and proposes design guidelines for systems that support authoring of problem solving knowledge by subject matter experts. Finally, it compares Disciple-RKF with the other development environments from the same DARPA program, providing further support for the proposed guidelines.

1 Introduction

Traditionally, a knowledge-based system is built by a knowledge engineer (KE) who needs to acquire the knowledge from a subject matter expert (SME) and to encode it into the knowledge base. This is a very difficult process because the SMEs express their knowledge informally, using natural language, visual representations, and common sense, often omitting many essential details they regard as being obvious. In order to properly understand an SME's problem solving knowledge and to represent it in a formal, precise, and "complete" knowledge base, the knowledge engineer needs to become himself/herself a kind of SME. Therefore this process is very difficult, error-prone, and time-consuming, being known as the knowledge acquisition bottleneck in system development.

One solution to this problem, pursued by the DARPA's Rapid Knowledge Formation program (2000-03), is the development of knowledge bases directly by SMEs, the central objective of this program being to enable distributed teams of SMEs to enter and modify knowledge directly and easily, without the need of prior knowledge engineering experience (http://cerberus.cyc.com/RKF/).

This paper presents an experiment of parallel knowledge base development by SMEs, and proposes guidelines for the design of systems that support authoring of problem solving knowledge by SMEs. It briefly introduces the Disciple-RKF integrated development environment used in this experiment, and discusses the main

phases of the experiment. Finally, it compares Disciple-RKF with the other two systems developed in the DARPA's RKF program, KRAKEN and SHAKEN [6], which provide further experimental support for the proposed design guidelines.

2 Disciple-RKF knowledge base development environment

The Disciple approach is the result of 20 years of research on developing a theory and associated methodologies and tools for knowledge base development [8, 4, 9, 2, 3]. Disciple-RKF, the implementation of the most recent version of the Disciple approach, is an agent shell with a knowledge base structured into an object ontology that describes the entities from an application domain, and a set of task reduction and solution composition rules expressed with these objects. The main functional components of Disciple-RKF are:

- A problem solving component based on task reduction. It includes a mixed-initiative (step-by-step) problem solver that allows the user and the agent to collaborate in the problem solving process, and an autonomous problem solver. It also includes a modeling assistant that helps the user to express his/her contributions to the problem solving process.
- A learning component for acquiring and refining the knowledge of the agent, allowing a wide range of operations, from ontology import and user definition of knowledge base elements (through the use of editors and browsers), to ontology learning and rule learning.
- A knowledge base manager which controls the access and the updates to the knowledge base. Each module of Disciple-RKF accesses the knowledge base only through the functions of the knowledge base manager.
- A window-based graphical user interface.

The development of the knowledge base of a Disciple-RKF agent is based on importing ontological knowledge from existing knowledge repositories, and on teaching the agent how to perform various tasks, in a way that resembles how an SME teaches a human apprentice. For instance, to teach the agent, the SME may formulate a specific problem and show the agent the reasoning steps to solve it, helping the agent to understand each of them. Each problem solving step represents an example from which the agent learns a general rule. As Disciple learns new rules from the SME, the interaction between the SME and Disciple evolves from a teacher-student interaction, toward an interaction where both collaborate in solving a problem. During this mixed-initiative problem solving phase, Disciple learns not only from the contributions of the SME, but also from its own successful or unsuccessful problem solving attempts, which lead to the refinement of the learned rules. This process is based on:

- *mixed-initiative problem solving*, where the SME and the agent solve problems in cooperation and the agent learns from the contributions of the SME;
- *integrated teaching and learning*, where the agent helps the SME to teach it (for instance, by asking relevant questions), and the SME helps the agent to learn (for instance, by providing examples, hints and explanations); and

- *multistrategy learning*, where the agent integrates different strategies, such as learning from examples, from explanations, and by analogy, to learn general concepts and rules.

3 Parallel knowledge base development experiment

A knowledge base development experiment with Disciple-RKF was conducted during the Military Applications of Artificial Intelligence (MAAI) course, taught at the US Army War College, in Spring 2003. This was a 10 week, 3 hours/week course, attended by 13 colonels and lieutenant colonels from different military services. The students, who had no prior knowledge engineering experience, were introduced to the Disciple approach, and used Disciple-RKF to jointly develop an agent for the determination of the centers of gravity (COG) of the opposing forces from a conflict. The concept of center of gravity is fundamental to military strategy, denoting the primary source of moral or physical strength, power or resistance of a force [7]. The most important objective of a force is to protect its own center of gravity, while attacking the center of gravity of its enemy.

Our approach to center of gravity determination, developed with the experts from the US Army War College, consists of two main phases: *identification* and *testing*. During the identification phase, center of gravity candidates from different elements of power of a force (such as government, military, people, economy) are identified. For instance, a strong leader is a center of gravity candidate with respect to the government of a force. Then, during the testing phase, each candidate is analyzed to determine whether it has all the critical capabilities that are necessary to be the center of gravity. For example, a leader needs to be protected, stay informed, communicate (with the government, the military, and the people), be influential (with the government, the military, and the people), be a driving force, have support (from the government, the military, and the people), and be irreplaceable. For each capability, one needs to determine the existence of the essential conditions, resources and means that are required by that capability to be fully operative, and which of these, if any, represent critical vulnerabilities. The testing of the critical capabilities is based on a general theory developed by Strange [7].

Figure 1 provides an overview of the performed experiment. Before starting the experiment, the Disciple-RKF agent was trained to identify leaders as center of gravity candidates. The knowledge base of this agent contained the definitions of 432 concepts and features, 29 tasks and 18 task reduction rules. However, the agent had no knowledge of how to test the identified candidates.

We then performed a joint domain analysis and ontology development with all the SMEs, by considering the example of testing whether Saddam Hussein, in the Iraq 2003 scenario, has all the required critical capabilities to be the center of gravity for Iraq. We determined which are the critical requirements for these capabilities to be operational, and which are the corresponding critical vulnerabilities, if any. Based on this domain analysis, we extended the ontology of Disciple-RKF with the definition of 37 new concepts and features identified with the help of the SMEs.

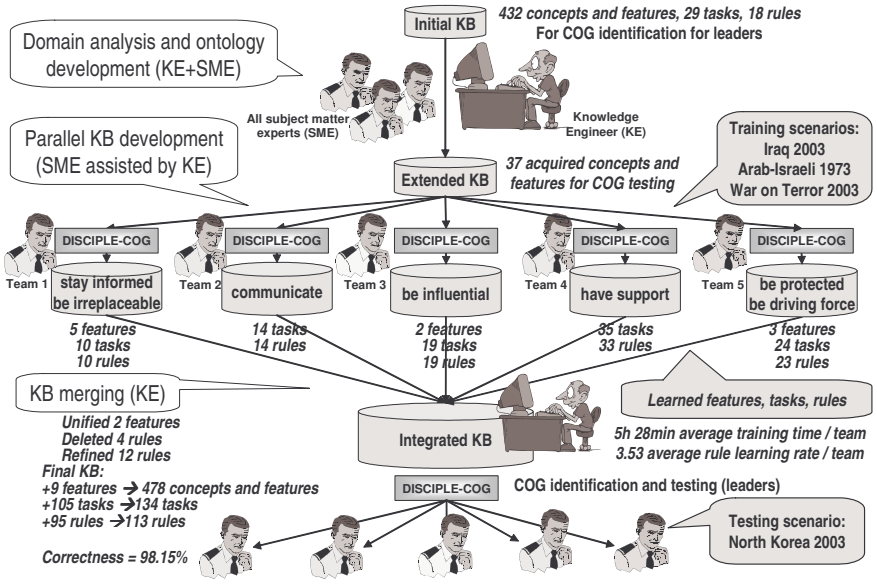

Fig. 1. Experiment of rapid knowledge base development by SMEs

The 13 SMEs were grouped into five teams (of 2 or 3 SMEs each), and each team was given a copy of the extended Disciple-RKF agent. After that, each team trained its agent to test whether a leader has one or two critical capabilities, as indicated in Figure 1. For instance, Team 1 trained its agent how to test whether a leader has the critical capabilities of staying informed and being irreplaceable. The training was done based on three scenarios (Iraq 2003, Arab-Israeli 1973, and War on Terror 2003), the SMEs teaching Disciple-RKF how to test each strategic leader from these scenarios. As a result of the training performed by the SMEs, the knowledge base of each Disciple-RKF agent was extended with new features, tasks, and rules, as indicated in Figure 1. For instance, the knowledge base of the agent trained by Team 1 was extended with 5 features, 10 tasks and 10 rules for testing whether a leader has the capabilities to stay informed and be irreplaceable. The average training time per team was 5 hours and 28 minutes, and the average rule learning rate per team was 3.53 rules/hour. This included the time spent in all the agent training activities (i.e., scenario specification, modeling SME's reasoning, task formalization, rule learning, problem solving, and rule refinement).

During each class the SMEs were introduced to the Disciple theory and tools corresponding to a particular agent training activity (e.g. scenario specification). The SMEs were supervised by knowledge engineers who were asked not to offer help, unless it was requested, and were not allowed to do SME's work. The SMEs were helped in the initial phases of learning to use a tool, after it was demonstrated by the course's instructor.

After the training of the 5 Disciple-RKF agents, their knowledge bases were merged by a knowledge engineer, who used the knowledge base merging tool of

Disciple-RKF. The knowledge engineer also performed a general testing of the integrated knowledge base, in which we included 10 new features, 102 new tasks, and 99 new rules (all acquired in less than 6 hours). During this process two semantically equivalent features were unified, 4 rules were deleted, and 12 other rules were refined by the knowledge engineer. The other 8 features and 83 rules learned from the SMEs were not changed. Most of the modifications were done to remove rule redundancies, or to specialize overly general rules.

Next, each SME team tested the integrated agent on a new scenario (North Korea 2003), and was asked to judge the correctness of each reasoning step performed by the agent, but only for the capabilities for which that SME team performed the training of the agent. The result was a very high 98.15% correctness.

In addition to the above experiment, agent training experiments were also conducted during the Spring 2001 and Spring 2002 sessions of the MAAI course, which were attended by 25 military experts [12]. However, *the Spring 2003 experiment is the first one ever that also included the merging of the developed knowledge bases into a functional agent*, demonstrating a capability for rapid and parallel development of a knowledge base by SMEs, with limited assistance from knowledge engineers. One should also mention that this was also the only experiment of parallel knowledge base development and integration, performed in the DARPA's RKF program.

The design of this experiment provides the following general guideline for the parallel development of a knowledge base by the SMEs.

Guideline 1: *Structure the knowledge base development process as follows:*

1) *Partition the application domain into sub-domains that are as independent as possible, and assign each partition to an SME.*
2) *Develop an object ontology for the entire domain (i.e., a hierarchical representation of the domain objects and their properties and relationships).*
3) *Develop the knowledge base for the top-level reasoning of the agent, allowing the agent to reduce any input problem solving task to a set of subtasks from the defined sub-domains.*
4) *Provide each SME with a copy of the agent, to author problem solving knowledge for his/her sub-domain. This will result in several parallel extensions of the object ontology, and in several sets of rules.*
5) *Integrate the knowledge bases of all the agents by merging all the extended ontologies into a shared ontology, and by keeping the developed rules in separate partitions. Any input problem solving task for the final agent is reduced to subtasks from different sub-domains. Then each subtask is solved using the rules learned from a single SME (or SME team).*
6) *Test the agent with the integrated knowledge base.*

Notice that the SMEs needed to agree on a common ontology, but not on how to solve a given subtask, significantly simplifying the knowledge base integration process. The next sections discuss two of the most important phases of this process and their corresponding guidelines.

4 Modeling the reasoning of an SME

In order to teach an agent how to solve problems, the SME has first to be able to make explicit the way he or she reasons. Our experience shows that *this is the single most difficult agent training activity for the SME*. In the following we will briefly describe the Disciple approach to this challenging problem, and then present several design guidelines that help simplifying it.

We have developed a very simple and intuitive modeling language in which the SME expresses the way he/she is solving a specific problem, using natural language, as if the SME would think aloud. The SME follows a task reduction paradigm, guided by questions and answers, successively reducing a complex problem solving task to simpler tasks, finding the solutions of the simplest tasks, and successively combining them into the solution of the initial task. The Disciple-RKF modeling language is illustrated in Figure 2 which includes a sequence of two reduction steps.

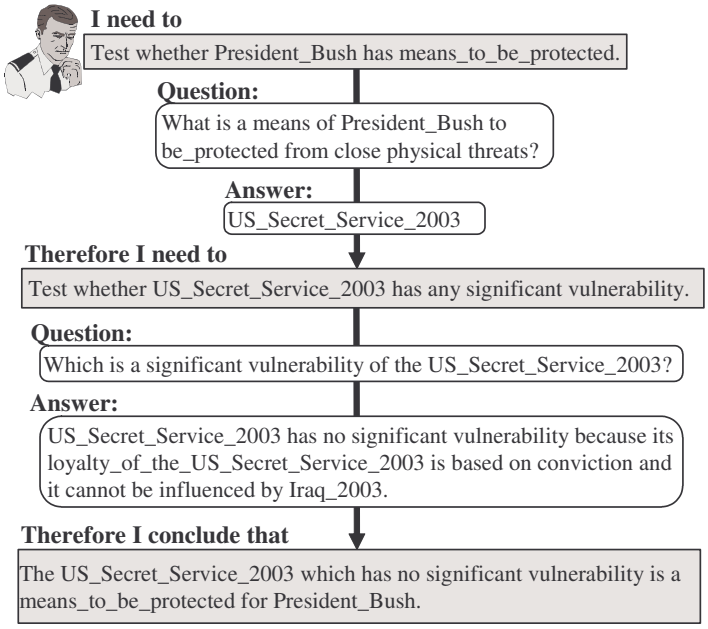

Fig. 2. A sequence of two task reduction steps

The question associated with the task from the top of Figure 2 considers some relevant piece of information for solving that task. The answer identifies that piece of information and leads the SME to reduce this task to a simpler task (or, in other cases, to several simpler tasks). Alternative questions correspond to alternative approaches to solve the current problem solving task. Several answers to a question correspond to several potential solutions. The modeling language includes many helpful guidelines for the SME, such as [3]:

Guideline 2: *Ask small, incremental questions that are likely to have a single category of answer (but not necessarily a single answer). This usually means ask who, or what, or where, or what kind of, or is this or that etc., not complex questions such as who and what, or what and where.*

Higher-level design guidelines, supported by our experiments, are briefly described below.

Guideline 3: *Train the SMEs to express their reasoning using the problem solving paradigm of the knowledge base development environment.*

In our experiment, the problem solving strategy of task reduction based on questions and answers was discussed and illustrated in action planning, course of action critiquing, and center of gravity identification. A general approach to center of gravity testing was also discussed, as presented at the beginning of section 3.

Guideline 4: *Allow the SMEs to express their reasoning in natural language, but provide them with helpful and non-disruptive mechanisms for automatic identification of the knowledge base elements in their phrases.*

For instance, Disciple-RKF includes a Modeling Assistant with an effective word completion capability. When the SME types a few characters of a phrase, such as, "`means to be protected`" (see Figure 2), the assistant proposes all the partially matching names from the knowledge base, ordered by their plausibility, to be used in the current context, including "`means_to_be_protected`." The SME selects this name only because it is simpler than typing it. However, now the system also partially "understands" the English sentence entered by the SME, which will significantly facilitate the follow-on process of language to knowledge transformation.

Guideline 5: *Do not ask the SMEs to provide general problem solving rules. Ask them to express how to solve specific problems.*

The modeling language and the associated Modeling Assistant help the SME to express how to solve a specific problem, thus providing examples of problem solving steps from which Disciple-RKF learns general rules, as will be discussed in section 5.

Guideline 6: *Provide non-disruptive mechanisms for helping the SMEs to express their reasoning process.*

For instance, at each step in the modeling process, the Modeling Assistant shows the SME both all the allowable user actions, and the recommended ones (such as "Copy and modify the current Task to define a Subtask"). Also, when possible, the Modeling Assistant automatically performs the selected action. Moreover, using analogical reasoning with previously learned rules, this assistant may suggest a partially instantiated pattern for the current question to be asked, or for the answer to the question.

The usefulness of this guideline is supported by the comparative analysis of the modeling process performed during the Spring 2002 agent training experiment (when the SMEs were not supported by the Modeling Assistant), and the modeling process performed during the Spring 2003 experiment (when the SMEs were helped by the Modeling Assistant). In Spring 2003, the modeling process was considered more

natural, was faster and more correct, due to the support provided by the Modeling Assistant [2].

The 13 SMEs who participated in the Spring 2003 experiment (see Figure 1) evaluated the difficulty in modeling their own reasoning process, using the task reduction paradigm. For instance, on a 5-point scale (strongly agree, agree, neutral, disagree, strongly disagree), 8 of them strongly agreed, 2 agreed, 3 were neutral, and none disagreed with the statement "*The task reduction paradigm implemented in Disciple is a reasonable way of expressing in detail the logic involved in identifying and testing strategic COG candidates.*" Moreover, 10 experts strongly agreed, 2 agreed, 1 was neutral, and none disagreed with the statement "*Subject matter experts that are not computer scientists can learn to express their reasoning process using the task reduction paradigm, with a reasonable amount of effort.*"

5 Rule learning

After the SME expresses his/her reasoning as a sequence of task reduction steps, as illustrated in Figure 2, the SME needs to help the agent to learn a general task reduction rule from each task reduction step. Rule learning is a mixed-initiative process between the SME (who knows why the reduction is correct and can help the agent to understand this) and the Disciple-RKF agent (that is able to generalize the task reduction example and its explanation into a general rule, by using the object ontology as a generalization language). This process is based on a communication protocol which takes into account that:

- it is easier for an SME to understand sentences in the formal language of the agent than it is to produce such formal sentences; and
- it is easier for the agent to generate formal sentences than it is to understand sentences in the natural language used by the SME.

Part of this process is illustrated in Figure 3. The left hand side of Figure 3 shows the task reduction example from the top part of Figure 2. This example is in natural language, except for the phrases with underscores which have already been recognized as representing elements from the Disciple's ontology, as discussed in the previous section. The right hand side of Figure 3 is the same task reduction example in a structured form. This structured form is generated by Disciple-RKF with the help of the SME. First Disciple-RKF proposes formalizations of the tasks from the example, by rephrasing the unstructured form of the task into a general task name (which does not contain any specific object), and one or several task features that identify the specific objects from the task. The SME can accept the formalization proposed by Disciple-RKF or, in rare cases, can modify it so that the task name is more understandable.

The natural language question and its answer from a task reduction step are intended to represent the SME's reason (or explanation) for performing that reduction. But they are in natural language and the SME has to help Disciple-RKF to "understand" them. The agent will use analogical reasoning with previously learned rules, as well as general heuristics, to hypothesize the meaning of the question-answer pair. It will generate plausible explanation fragments (ordered by their plausibility), and the

SME will select those that best express this meaning. The SME may also help the agent to propose the right explanation pieces by proving hints, such as pointing to a relevant object that should be part of the explanation. Each explanation fragment proposed by the agent is a relationship (or a relationship chain) involving instances, concepts and constants from the task reduction step and from the knowledge base. For instance, the right hand side of Figure 3 shows the explanation fragments selected by the SME, from those proposed by the agent.

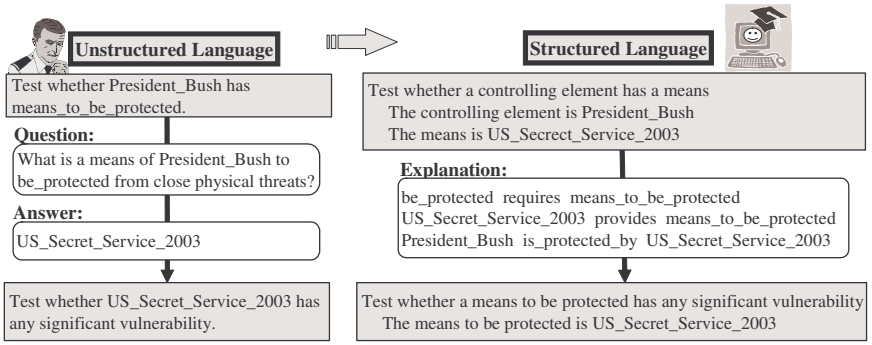

Fig. 3. Mixed-initiative language to knowledge translation

Once the explanations have been identified, the agent generates the task reduction rule shown in Figure 4. This rule has an informal structure, shown at the top of Figure 4, and a formal structure, shown at the bottom of Figure 4. Compare the informal structure of the learned rule with the example from the left hand side of Figure 3. As one can see, this rule is generated by simply turning the constants from the example into variables. Therefore, the informal structure of the rule preserves the natural language of the SME, and is used in agent-user communication. This rule should be interpreted as follows: If the task to be solved is T_1, I am asking the question Q, and if the answer is A, then I can reduce T_1 to T_{11}.

The formal structure of the learned rule, shown at the bottom of Figure 4, corresponds to the formalized version of the example from the right hand side of Figure 3. This formal structure is used in the actual problem solving process, and is interpreted as follows: If the task to be solved is FT_1, and the rule's applicability condition is satisfied, then I can reduce FT_1 to FT_{11}. Notice that both FT_1 and FT_{11} are formal task expressions. Moreover, instead of a single applicability condition, the rule in Figure 4 has a plausible version space for the exact condition, because the rule is only partially learned. This rule, generated from a single example and its explanation, will be further refined. The plausible lower bound of the version space is the least general generalization of the objects from the example and the explanation, based on the object ontology (including the definitions of the features). Similarly, the plausible upper bound is the most general generalization. For instance, "`US_secret_service`" was generalized to "`protection_service`" in the lower bound, and to "`agent`" in the upper bound.

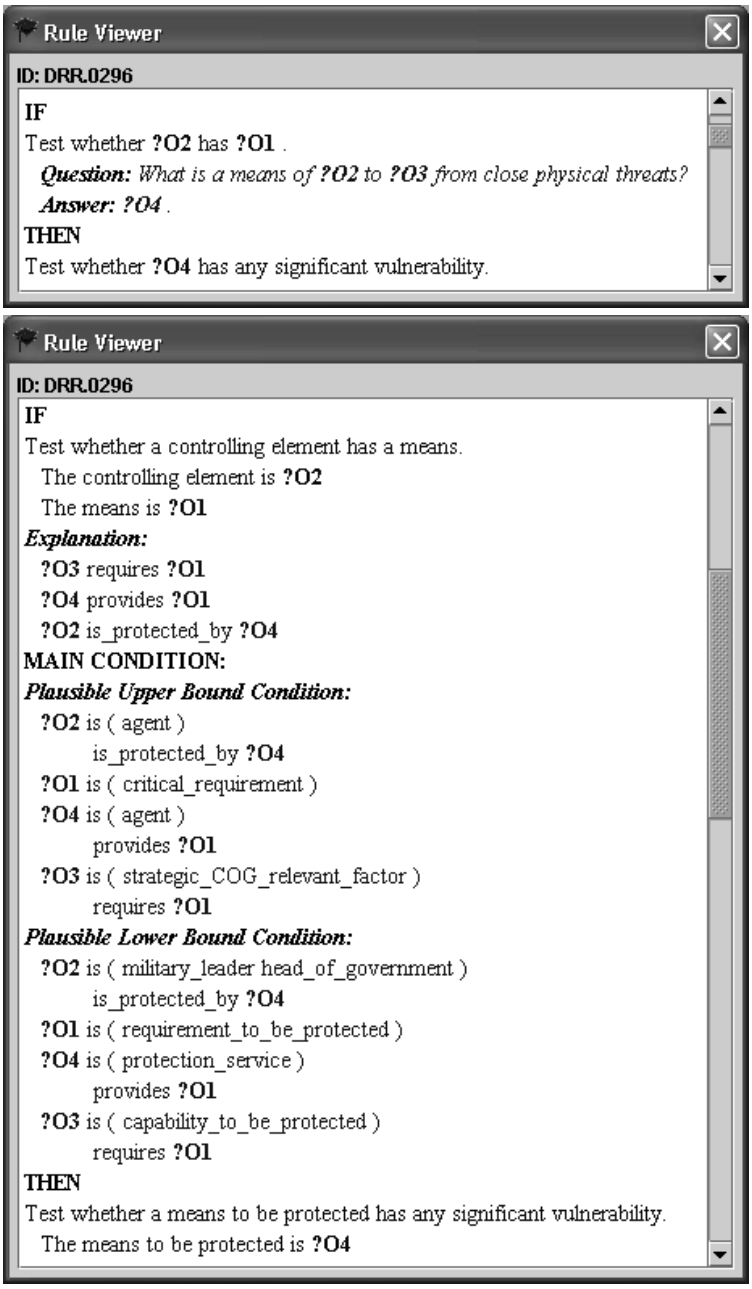

Fig. 4. A learned rule

Once a rule is learned, it is used by Disciple-RKF in problem solving. For instance, the rule in Figure 4 was used to identify the means to be protected for the other leaders from the training scenarios. The corresponding reductions that were accepted by the user were used as positive examples by the agent to further generalize the plausible lower bound condition of the rule. Those that were rejected were used as negative examples to specialize the rule's plausible upper bound. If the agent and the SME identified an explanation of why a reduction was wrong, then a corresponding "except-when" plausible version space condition was added to the rule. In the future, the rule will be applicable only if the except-when condition will not be satisfied. During this process, the agent may learn complex rules with quantificators and negation. More details on the learning methods used are given in [9, 2].

The high accuracy of the learned rules in the performed experiment (see Figure 1) shows that the above approach to rule learning from SMEs is very successful, leading to the learning of good quality rules, in a very short time, and from a small number of examples.

The performed experiments support the following guidelines for the design of systems that help SMEs to author problem solving knowledge.

Guideline 7: *Provide the SME with easy to use features for helping the agent to understand the natural language phrases of the SME.*

For instance, the SME helped Disciple-RKF to generate the correct meaning of the SME's phrases by pointing to relevant objects, or by guiding the refinement of abstract structures.

Guideline 8: *Implement mechanisms to automatically generalize the specific examples provided by the SMEs into general knowledge pieces.*

These mechanisms are illustrated by the rule learning process of Disciple-RKF.

Guideline 9: *Verify the general rules learned by asking the SME to judge the results of their applications in other cases, and use SME's critiques to automatically refine them.*

Because it is much easier for an SME to judge concrete cases than general pieces of knowledge, in our approach, the SME does not even see the rules, but only their application to specific situations.

Guideline 10: *Implement mechanisms to automatically check the learned knowledge pieces, and to correct them by asking clarification questions about specific examples.*

For instance, the agent may determine that a variable from the THEN part of a rule is not linked to any variable from the IF part of the rule. This is indicative of a rule which was learned based on an incomplete explanation, causing the agent to reinitiate the explanation generation process. Sometimes, the missing explanation is so obvious to the SME that it is simply ignored as, for instance, the following one: "`US_2003 has_as_government government_of_US_2003.`" The agent will automatically select such an explanation, if it provides a link to an unconstrained variable. Sometimes, even when all the rule's variables are linked, the number of rule instances may still be very large. In such a case the agent will attempt to identify which vari-

ables are the least constrained, and will attempt to further constrain them, by proposing additional explanation pieces.

Guideline 11: *The knowledge authoring approach should be incremental, allowing imperfections from the SME.*

For instance, the explanation pieces selected by the SME do not need to constitute a complete explanation of a task reduction step. In such a case the learned rule will generate wrong reductions which will reveal to the SME what factors were not considered, and what new explanation pieces should be added. Another common imperfection is for the SME to select too many explanation pieces. This leads to a rule which is less general than it should be, but it is correct nevertheless. The agent may then automatically combine and generalize similar rules.

At the end of the Spring 2003 experiment, 7 SMEs strongly agreed, 4 agreed, 1 was neutral and 1 disagreed with the statement "*I think that a subject matter expert can use Disciple to build an agent, with limited assistance from a knowledge engineer*". This is a powerful experimental support for the Disciple approach.

6 Related research and conclusions

In addition to Disciple-RKF, two other systems for acquiring expert problem solving knowledge were developed in the DARPA's RKF program, KRAKEN and SHAKEN [6]. KRAKEN was developed by the Cycorp team, which also included researchers from Northwestern University, Stanford University, Information Science Institute, University of Edinburgh, Teknowledge, and SAIC. SHAKEN was developed by the SRI team, which also included researchers from University of Texas at Austin, Boeing, Stanford University, MIT, Northwestern University, University of Massachusetts, University of Western Florida, Information Science Institute, PSR, and Pragati.

In the DARPA's RKF program, KRAKEN and SHAKEN were evaluated on the course of action (COA) critiquing challenge problem [6], while Disciple-RKF was evaluated on the center of gravity (COG) challenge problem. However, a previous version of Disciple, Disciple-COA, developed as part of the DARPA's HPKB program, was also evaluated on the course of action critiquing problem, as discussed in [10, 11]. As opposed to Disciple-COA and Disciple-RKF, which use rule chaining, the course of action critiquing of both KRAKEN and SHAKEN is more limited, being based on a single rule, where the antecedent represents some characteristics of the course of action, and the consequent is a critique of the course of action. Therefore, the kind of problem solving knowledge acquired by Disciple was more complex than that acquired by KRAKEN and SHAKEN.

The approach taken by KRAKEN and SHAKEN to the authoring of problem solving knowledge from SMEs was radically different from that of Disciple-RKF. The philosophy behind both systems was to develop advanced tools that would allow the SMEs to directly author general problem solving rules. The main difference between the two systems was in the type of the rule editing tools used. KRAKEN used text-oriented tools, its key strategy for facilitating SME-authoring being natural language presentation and a knowledge-driven acquisition dialog with natural language under-

standing, supported by the large Cyc knowledge base [5]. Notice here the relationship with *Guideline 4* on the use of natural language.

SHAKEN provided the SME with a graph (concept map) editor to represent the antecedents and the consequent of a rule pattern. This graph editor facilitated the SME's use of the objects and relationships from the knowledge base of SHAKEN (as also suggested in the second part of *Guideline 4*). Such a graph was then automatically translated into a formal rule, so that the SMEs did not need to be trained in formal logic [1].

It is interesting to notice that the conclusions of the experiments performed with KRAKEN and SHAKEN indirectly support the Disciple approach to acquiring problem solving knowledge, and several of the design guidelines stated in the previous sections. For instance, one of these conclusions is: "For purposes of rule elicitation, the focus on a particular scenario rendered the initial rule articulation much more manageable for the SME. It would have been more difficult to articulate rules from more universally acceptable general principles initially" [6]. This justifies *Guideline 5* which is also based on the idea that it is easier for an SME to reason with specific scenarios.

However, "an unfortunate consequence of the scenario focus is that the SMEs occasionally tended to overly restrict a rule by including unnecessary details from a particular scenario" making necessary for the rule to be further generalized [6].

Moreover, sometimes the generalization performed by the SME was incorrect in the sense that it was not really a generalization of the studied scenario. The proposed solution to this problem is to have the rule "examined for generalization, either by a system tool, or in consultation with a knowledge engineer" [6]. Our experiments with Disciple have also revealed the SME's tendency to provide more specific explanations of the problem solving episodes considered, as stated in the explanations for *Guideline 11*. However, as opposed to SHAKEN and KRAKEN, the generalization of the examples is performed by the Disciple learning agent and not by the SME, and it is correct (with respect to the current ontology). Therefore *Guideline 8* suggests that the generalizations should be performed by the agent.

Because the SMEs cannot test the correctness of a rule by direct examination, these rules acquired by KRAKEN and SHAKEN were tested on the concrete scenarios that inspired them in the first place, allowing the SMEs to further improve the rules. This supports *Guideline 9* and the approach taken in Disciple-RKF where the SMEs do not even see the rules learned by the system, but only the results of their application. Moreover, a Disciple rule learned from a scenario is guaranteed to correctly work for that scenario.

A global conclusion of the developers of KRAKEN and SHAKEN is that "given representational subtleties, especially those associated with negation and quantification, fully automated elicitation of any arbitrary complex rule remains somewhat elusive" [6]. In essence, these experiments confirmed that it is difficult for an SME to formulate general rules, and these rules are very likely to be incomplete and only partially correct (which justifies *Guideline 5*).

On the contrary, Disciple-RKF agents successfully acquired a significant number of accurate problem solving rules from the SMEs because the Disciple approach requires the SMEs to do what they know best (i.e. to solve specific problems), and not

to perform knowledge engineering tasks, which proved to be very difficult for them, even with the very powerful knowledge engineering tools offered by KRAKEN and SHAKEN.

Maybe the most significant factor in the comparison of Disciple-RKF and, in general, of the Disciple approach to acquiring and integrating problem solving knowledge from SMEs (on one hand), and the approaches illustrated by KRAKEN and SHAKEN (on the other hand), is that the Disciple team was the only team from the DARPA's RKF program that has successfully conducted a parallel knowledge base development and integration experiment.

The deployment and evaluation of Disciple-RKF have also revealed several limitations of this approach and have provided numerous ideas for improvement. For instance, while the subject matter expert has an increased role and independence in the agent development process, the knowledge engineer still has a critical role to play. The knowledge engineer has to assure the development of a fairly complete and correct object ontology. The knowledge engineer also has to develop a generic modeling of the expert's problem solving process based on the task reduction paradigm. Even guided by this generic modeling, and using natural language, the subject matter expert has difficulties in expressing his reasoning process. Therefore more work is needed to develop methods for helping the expert in this task, along the path opened by the Modeling Advisor.

The experimentations also revealed that the mixed-initiative reasoning methods of Disciple-RKF could be significantly empowered by developing the natural language processing capabilities of the system.

Finally, because the expert who teaches Disciple-RKF has no formal training in knowledge engineering or computer science, the knowledge pieces learned by the agent and the knowledge base itself will not be optimally represented, and will require periodic revisions by the knowledge engineer. Examples of encountered problems with the knowledge base are: semantic inconsistencies within a rule, proliferation of semantically equivalent tasks, and the violation of certain knowledge engineering principles. It is therefore necessary to develop mixed-initiative knowledge base reformulation and optimization methods to identify and correct such problems in the knowledge base.

Acknowledgments

This research was sponsored by DARPA, AFRL, AFMC, USAF, under agreement number F30602-00-2-0546, by the AFOSR under grant no. F49620-00-1-0072, and by the US Army War College. Several persons supported this effort, including Jerome Comello, William Cleckner, Murray Burke, William Rzepka, Douglass Campbell, David Brooks, and Christopher Fowler. We are also grateful to the anonymous reviewers for their insightful comments.

References

1. Barker, K., Blythe, J., Borchardt, G., Chaudhri, V.K., Clark, P.E., Cohen, P, Fitzgerald, J., Forbus, K., Gil, Y., Katz, B., Kim, J., King, G., Mishra, S., Morrison C., Murray, K., Otstott, C., Porter, B., Schrag, R.C., Uribe, T., Usher, J., Yeh, P.Z.: A Knowledge Acquisition Tool for Course of Action Analysis. In: Proc. of the 15th Innovative Applications of Artificial Intelligence Conference. AAAI Press, Menlo Park, California, USA (2003) 43-50
2. Boicu, M.: Modeling and Learning with Incomplete Knowledge, PhD dissertation. George Mason University, Fairfax, Virginia, USA (2002)
3. Bowman, M.: A Methodology for Modeling Expert Knowledge that Supports Teaching Based Development of Agents, PhD dissertation. George Mason University, Fairfax, Virginia, USA (2002)
4. Dybala, T.: Shared Expertise Model for Building Interactive Learning Agents, *Ph.D. Dissertation,* Department of Computer Science, George Mason University, Fairfax, Virginia (1996)
5. Lenat, D.B.: CYC: A Large-scale Investment in Knowledge Infrastructure. Communications of the ACM, 38(11), (1995) 33-38
6. Pool, M., Murray, K., Fitzgerald, J., Mehrotra, M., Schrag, R., Blythe J., Kim, J., Chalupsky, H., Miraglia, P., Russ, T., Schneider, D.: Evaluating Expert-Authored Rules for Military Reasoning. In: Proc. of the 2nd Int. Conf. on Know-ledge Capture. ACM Press, Florida, USA (2003) 69-104
7. Strange, J.: Centers of Gravity & Critical Vulnerabilities: Building on the Clausewitzian Foundation So That We Can All Speak the Same Language. Quantico, Virginia, USA, Marine Corps University (1996)
8. Tecuci, G.: DISCIPLE: A Theory, Methodology and System for Learning Expert Knowledge, *Thèse de Docteur en Science,* University of Paris-South (1988)
9. Tecuci, G.: Building Intelligent Agents: An Apprenticeship Multistrategy Learning Theory, Methodology, Tool and Case Studies. Academic Press, London (1998)
10. Tecuci G., Boicu M., Bowman M., Marcu D., Shyr P., and Cascaval C.: An Experiment in Agent Teaching by Subject Matter Experts. *International Journal of Human-Computer Studies*, 53 (2000) 583-610
11. Tecuci G., Boicu M., Bowman M., and Marcu D., with a commentary by Burke M.: An Innovative Application from the DARPA Knowledge Bases Programs: Rapid Development of a High Performance Knowledge Base for Course of Action Critiquing. *AI Magazine,* 22, 2. AAAI Press, Menlo Park, California (2001) 43-61
12. Tecuci G., Boicu, M., Marcu, D., Stanescu, B., Boicu, C., and Comello, J.: Training and Using Disciple Agents: A Case Study in the Military Center of Gravity Analysis Domain. AI Magazine 23(4) (2002) 51–68