

APPLICATION OF DISCIPLE TO DECISION MAKING IN COMPLEX AND CONSTRAINED ENVIRONMENTS

MICHAEL BOWMAN¹, GHEORGHE TECUCI¹, and MARION G. CERUTI²

¹ Learning Agents Laboratory, Department of Computer Science, George Mason University
MS 4A5, 4400 University Dr., Fairfax, VA, USA 22030

² Space and Naval Warfare Systems Center
D4121, 53560 Hull Street, San Diego, CA 92152-5001, USA

Abstract

This paper describes Disciple, an Artificial Intelligence based decision aid which subject-matter experts can train and use when making decisions under stressful, complex, and constrained conditions. The tool was developed and used under the Defense Advanced Research Projects Agency's High Performance Knowledge Base and Rapid Knowledge Formation programs. Some domains in which the tool would be applicable are described, with particular emphasis on military battle planning. The paper concludes with a discussion of future trends in decision-support application tools.

Keywords

Automated tools and agents, artificial intelligence, command and control, decision-support system, knowledge-acquisition bottleneck, subject-matter expert

1 Introduction

The paper describes the Disciple system, which was developed by researchers at the George Mason University Learning Agents Laboratory, based on artificial intelligence techniques. It also describes challenges and similarities in decision making under pressure in the military, medical, and manufacturing domains. Disciple could contribute to enhanced decision-making efficiency as a decision aid in these domains, as demonstrated by its application to military battle planning problems.

This paper surveys the contribution of Disciple to the following areas:

- The knowledge acquisition (KA) bottleneck;
- Expert decision making through task reduction;
- Decision aid development methodology;
- Experimental evaluation;
- DARPA's High Performance Knowledge Base (HPKB) program (See, [5], [7] & [10]);
- HPKB Army Course-of-Action (COA) critiquing.

Disciple's history, capabilities, inner workings, future research directions, and publications are described in detail in [11]. More information can be found on the GMU LALAB web page, <http://lalab.gmu.edu>.

2 The knowledge-acquisition bottleneck

For artificial intelligence to become truly useful in practical applications and environments it is necessary to identify, document, and integrate into automated systems the knowledge that people use to solve problems. This process, called knowledge acquisition (KA), has become a bottleneck in the development of AI-based systems because KA has been found to be difficult, labor intensive, and error prone (See, for example, [1] and [9]).

As it is conducted typically today, KA involves direct interaction between a trained knowledge engineer (KE) and a domain expert. The KE attempts to understand the knowledge and methods the expert uses to solve problems, and to represent them in the knowledge base of the system to be built. This knowledge elicitation and representation process is particularly difficult because the form in which the expert expresses his or her knowledge is significantly different from the manner in which it should be represented in the knowledge base. After the knowledge is elicited it has to be verified by the expert and where appropriate, corrected by the KE. This KE-mediated transfer of knowledge between the domain expert and the knowledge base is the major contributor to the knowledge-acquisition bottleneck described above.

This paper proposes a solution to the KA bottleneck. It describes research in the development of a general methodology for modeling and representing an expert's problem-solving process in a knowledge-based agent, through teaching-based ontology formation and rule learning. In this approach, the user teaches the agent to perform various tasks in a way that resembles how the user would teach an apprentice or student, by giving the agent examples and explanations, and by supervising and correcting its behavior. Moreover, this approach

produces a verified and validated knowledge base that can be updated efficiently in response to changes in the environment or to modifications of the application's requirements.

This methodology is designed to accomplish the following functions:

- Helps the domain expert conceptualize the problem-solving process.
- Allows experts to document, order, and justify their decision-making process.
- Facilitates directly the expression of this information to the agent.
- Governs the entire agent-training and knowledge-base development process.
- Facilitates natural interactions that allow the agent to learn complex problem-solving rules.
- Supports the reuse of existing ontologies and the extension of these ontologies for the problem domain.

An important feature of this methodology is that it is based on the task reduction paradigm of problem solving. In the next section we discuss the use of task reduction for expert decision making, showing that a task-reduction based methodology has clear potential for applications to decision making in a wide variety of domains characterized by constraints (e.g. time and safety) and complexity.

3 Expert decision making and task reduction

Experts constantly need to make complex decisions rapidly. The human mind has an enormous capacity to collect information, evaluate situations, compare options, and reach conclusions. Even in the most complex, life-threatening, decision-making environments the human mind still outperforms other tools, such as computers, in many areas. The surgeon wielding a scalpel, the pilot landing an aircraft, and the military commander maneuvering troops on the battlefield all rely primarily on their cognitive processes to direct information collection, analyze data, make decisions and act on those decisions. The ability to make sound decisions rapidly is easy to recognize and demonstrate. However, to capture and explain how and why experts make rapid, complex decisions is much more difficult. (See, for example, [6]). Bad decisions generally are no easier to analyze, explain, and document than are good decisions. When our decisions are faulty, the processes of explaining the factors that led us to a conclusion, and finding the particular factors that led us astray, are very difficult for most people. When asked to explain a decision, most people immediately are able to identify one or two salient features of the information that supports their decision. They

generally find it very difficult, however, to generate rapidly a very thorough list of salient facts or to establish the relationships between them. This is a key factor in the KA bottleneck faced by KEs who build expert, knowledge-based systems.

Despite the complexity of the problem and the variety of approaches available, one methodology that is seen consistently in explaining and documenting the accessible parts of the decision-making processes is task reduction. Kirlik, et al. have suggested that task-simplification strategies based mainly on perception and pattern recognition are fundamental to the novice-expert shift in dynamic decision making [8]. Task reduction is the process of taking a complex problem and reducing it to a series of less and less complex problems until what remains is a relatively simple problem for which we have enough information to reach a conclusion. (See, for example, [6]). Task reduction is a powerful method of documenting and representing decision-making processes, but the approach has documented weaknesses as well as strengths.

The use of task reduction as a problem-solving methodology is well understood and documented in systems engineering. Its use in KA is widespread, but not dominant. One challenge is to introduce the method to non-system engineering or KA SMEs who are initially unfamiliar with it, and induce them to adapt to the requirements of the method. An expert attempting to capture all of the steps and factors in solving a complex technical problem almost never gets the list right the first time. One of the key challenges of our research is the collection and representation of this type of decision-making information. To support our higher-level goals, the methodology of modeling this information must meet the following requirements:

- Appear natural to the expert providing information.
- Be general enough to be used in a variety of domains.
- Be thorough enough to support teaching-based ontology and rule development.

In a wide variety of domains, experts face cognitively demanding tasks that have costly consequences for poor or ineffective performance [3]. Many mishaps in multiple domains have been attributed to human error [3]. Table 1 lists some examples of challenging tasks for which a mistake could produce catastrophic consequences. It also lists some task-reduction techniques that experts use to approach and solve problems. The military domain is explored in detail below, and the other domains, more briefly. Any automated decision-support tool needs to be flexible enough to accommodate various task-reduction techniques of the application domain.

Table 1: Examples of decision making under pressure and task reduction in diverse and challenging situations

<u>Attribute</u>	<u>Military domain</u>	<u>Medical domain</u>	<u>Manufacturing domain</u>
Knowledge Acquisition	Intelligence gathering	Diagnostic tests, examinations	Situation assessment
Source of Decision Pressure	Public perception, expectation of commanding officers, uncertainty of battle, (The "fog of war")	Patient expectation, clinical schedule, progress of disease	Customer expectation, production schedule, marketplace competition, inherent environmental dangers, government regulations
Consequence of Error	Failed military mission, loss of assets, wartime casualties, multiple fatalities	Untreated disease, treating the wrong disease, selecting wrong treatment for correctly diagnosed disease, patient cost and suffering, fatalities	Damage to products or materials, contract cancellations, injury accidents, litigation, financial damage, fatalities
Examples	Developing a COA or plan for a military operation	Diagnosing a disease when multiple diseases are present or when a group of symptoms has a combination of causes	Determining a strategy to transport heavy, expensive, unstable, hazardous and/or fragile materials or products in a factory/shipyard
Task-reduction Techniques	COA sequence according to published doctrine [5]	Step-by-step medical procedure	Safety checklists and engineering Guidelines

4 DARPA HPKB and Disciple

The DARPA High Performance Knowledge Base (HPKB) program ran from 1997 to 1999. (See, for example, [2], [5] and [7].) The goal of HPKB was to produce the technology needed for the rapid construction of large knowledge bases (with many thousands of axioms) that provide comprehensive coverage of topics of interest, are reusable by multiple applications with diverse problem-solving strategies, and are maintainable in rapidly changing environments.

The intention of HPKB was to produce alternative knowledge-base development environments, that combined the necessary foundation-building, knowledge-acquisition, and problem-solving technologies into an integrated development environment, and to use those environments to build reusable knowledge-base components for multiple DARPA application projects.

The organizations participating in HPKB were given the challenge of solving a selection of knowledge-based problems in a particular domain, and then modifying their systems quickly to solve further problems in the same domain. (See, for example, [5], [7] and [10]). The aim of the exercise was to test the claim that, with the latest AI technology, large knowledge bases could be built quickly and efficiently.

The GMU Learning Agents Laboratory's approach to HPKB was based on the Disciple Toolkit (See, for example, [11]). Disciple is a theory, methodology and agent shell for rapid development of knowledge bases and knowledge-based agents by domain experts with limited assistance from knowledge engineers. The Disciple agent shell consists of an integrated set of knowledge acquisition, learning, and problem solving modules for a generic knowledge base structured into two main components: an object ontology that defines the concepts from a specific application domain, and a set of problem solving rules expressed with these concepts. The process of developing a specific Disciple agent, starting from the Disciple shell, relies on importing ontological knowledge from existing knowledge repositories, and on teaching the agent how to perform various tasks. This paradigm allows an expert to teach the agent as though it were a human apprentice by giving the agent specific examples of tasks and solutions; by providing explanations of these solutions; and by supervising the agent as it performs new tasks. From such interactions with the expert Disciple learns general problem solving rules, building, verifying and improving its knowledge base.

The rationale for this approach is that experts can perform the following actions more efficiently: update (vs. create) an ontology; define examples (vs. define rules); understand a formal sentence (vs.

create one); and provide hints (vs. give explanations).

During Disciple-expert interactions, called training episodes, the experts share their expertise with the agent, which continuously is extending and improving its knowledge and performance abilities. The agent's capabilities are achieved by a synergistic integration of the following learning and knowledge acquisition methods: systematic elicitation of knowledge; empirical inductive learning from examples; learning from explanations; and learning by analogy and experimentation.

In our HPKB applications, a military expert taught Disciple to perform various tasks in a way that resembles how the expert would teach a novice. Disciple learns from the expert, building and improving its knowledge base and expanding its problem-solving capability.

To conduct productive, interactive training episodes with Disciple, the experts must understand and document their decision-making process with respect to the examples to be used in the training episode. This general methodology for modeling and representing expert problem solving is described in detail below in the context of one of the HPKB challenge problems.

5 HPKB COA critiquing

The problem domain for one of the HPKB challenge problems was the critiquing of military courses of action (COA). A military COA is a preliminary outline of a plan for how a military unit might attempt to accomplish a mission. The example COAs used for this research and provided by the US Army were specified in standard military formats consisting of a multi-paragraph textual description of the COA and a graphical representation of the COA in the form of a sketch using standardized symbols for units, activities and geo-spatial relationships.

The developed Disciple critiquer identifies strengths and weaknesses of a COA with respect to the principles of war and the tenets of army operations [14]. (See, for example, [5]). According to U.S. Army doctrine, the nine principles of war are objective, offensive, mass, economy of force, maneuver, unity of command, security, surprise, and simplicity. They provide general guidance for the conduct of war at the strategic, operational and tactical levels. The tenets of U.S. Army operations, which describe the characteristics of successful operations are initiative, agility, depth, synchronization and versatility.

The Disciple-COA critiquing agent was developed to act as an assistant to a military commander and staff, helping them to choose the best COA for a particular mission. To develop the agent's

knowledge base, we performed a task based modeling of the principles of war and tenets of army operations. By task based, we mean the consideration of military units and their assigned tasks. Specifically we examined the tasks assigned to units of interest; the ability of units to accomplish assigned tasks as appropriate for the unit type, size and/or condition; and the probability that the completion of the assigned tasks will contribute to the success of the overall mission.

A general, common understanding of the principles of war and the tenets of army operations exists because they are well documented in military literature. They are defined succinctly in [14] with passages such as "At all levels of war, successful application of maneuver requires agility of thought, plans, operations and organizations." A problem is that experts disagree on what agility is and how to utilize it to apply the principle of maneuver in a COA. Application of the principles and tenets, as described in [14] is just the beginning of a good critique of a COA or plan. GMU's goal was to create a Disciple agent that contained the common understanding of the principles and tenets while retaining sufficient flexibility to allow rapid personalization by the experts training and using the agent.

The COA domain represented a change from past uses of Disciple in that it required the development of a question-answering system rather than a planning system. Success of Disciple in this new domain indicates that the learning and knowledge acquisition methods developed are general and even domain independent. Moreover, the participating teams were required to use a common ontology developed by Teknowledge and Cycorp. This was a new challenge for our approach, which up to that point was used only with ontologies developed specifically for Disciple. Therefore, the challenge was particularly well suited to our learning and knowledge-acquisition methods.

6 Agent development methodology

During HPKB, Disciple did not include a modeling, or drawing tool. The modeling described in the following sections can be done with paper and pencil or any automated drawing tool, such as Microsoft Powerpoint. We used paper-and-pencil sketching for real-time modeling during the time-constrained HPKB evaluation of COA critiquing. During less time-constrained modeling we used numerous tools before standardizing on Microsoft Powerpoint for modeling. Using Powerpoint allowed us to cut and paste task names, questions, and answers efficiently into Disciple's interface windows during interactive agent training and

ontology development. An integrated modeling tool has been added to Disciple for the RKF program. (See, for example, [2]). This tool automatically transfers many of the elements of the domain model directly to agent development elements of Disciple.

The importance and usefulness of this methodology for modeling and representing an expert decision-making process, (as expressed in the thorough and accurate task reduction steps, questions and answers that the expert provides), is captured in our high-level research goals. Our objective is to have a domain expert interact directly and independently with the agent-building shell to train an agent to solve complex problems. Experts type natural language text, use mouse clicks to provide hints for explanation generation, and use mouse clicks to identify and select correct explanations. We do not expect an expert to create formal sentences for explanations or explicitly create rules in machine-executable language. This modeling provides the basis for the expert-agent interaction.

An important experimental observation from the research of this methodology is that whereas it was developed primarily to support direct expert-agent interaction, it works equally as well in support of experts working with KEs in the traditional approach to agent and expert-system development. We used the methodology extensively to represent knowledge elements conveyed directly from an expert to a KE, who then used the models with little or no assistance from the expert, as templates for direct creation and modification of rules.

The domain-modeling and representation methodology consists of the steps listed below. Several of the steps are iterative and may be repeated as necessary to provide complete coverage of the problem area. Steps 6, 7 and 8 may occur concurrently or may loop back to and from one another. Steps 9, 10 and 11 comprise the interactive training episodes with the Disciple agent. The following steps are described below in more detail.

1. Identify the high-level problem to be solved.
2. Identify categories of potential solutions.
3. Identify a specific example problem to be solved.
4. Brainstorm potential solutions for the example problem within a category of solution.
5. Select a potential solution to be modeled
6. Identify the complete set of task-reduction steps for that potential solution.
7. Identify a question and answer that justifies progression from one step to another in the task reduction solution path.
8. Identify concepts and features for Disciple's ontology.
9. Use the questions and answers as the basis for hints provided to Disciple that consist of selecting relevant concepts, instances and relationships.
10. Use the questions and answers to identify correct justifications among the justifications provided by Disciple during rule development or refinement.
11. Repeat the process for other solution paths.
12. Check solutions and refine rules for other data sets.

6.1 Identify the high-level problem

For COA critiquing, we identify the top-level problem as: "critique COAs with respect to the principles of war and tenets of army operations." A Disciple agent was designed and developed to concentrate in general terms on this top-level task.

We assume that the Disciple ontology has at least a rudimentary representation of the problem domain. In terms of COA critiquing, this was an ontology of military organizations, military equipment and basic military terminology such as: mission, COA, offense, and defense. (See, for example, [5]). Disciple and this methodology will support starting from "scratch" but with the ontology import and re-use capability of Disciple, this may not be necessary.

6.2 Identify categories of potential solutions

To begin the modeling process, the expert can begin one level down from this top-most task. One HPKB challenge problem was to identify strengths and weaknesses in COAs with respect to the principles of war and the tenets of army operations which represent categories of solutions. Therefore, we identified a high-level task for each of the principles and tenets for a total of 14 high level tasks which will generate essentially 14 separate models. Identifying these high-level (versus top-level) tasks constitutes the first task reduction and leads to step 3.

6.3 Identify a specific example problem to be solved

This is the first step to be revisited multiple times, as it is not likely or necessary to identify all possible solutions to a selected problem on the first try. The top-level task of critiquing COAs reduced to a model for each principle of war and tenet of army

operations. This description focuses primarily on modeling critiques based on the principle of surprise for a particular COA. This amounts to selecting an example problem to be solved.

6.4 Brainstorm potential solutions for the example problem

Assessing a COA with respect to the principle of surprise can involve several possible ways of looking for the element of surprise in a COA. Our experience with modeling surprise caused us to revisit this step three times. In our first brainstorming of potential solutions with respect to surprise we identified two obvious solution paths looking for assigned actions that, by definition, called for the use of surprise or deception. During the HPKB evaluation of COA critiquing, Army experts identified two additional possible solution paths. During our KA experiments the experts identified an additional possible solution path.

At this point in modeling, we recommend that the expert consider and record potential solution paths within a category unconstrained by consideration of how a conclusion might be reached or whether or not available sample data will support providing concrete examples of a solution during training. Determining which solution paths to pursue was determined in steps 5 and 6.

6.5 Select a solution to be modeled

The next step is to select a single solution path to model through completion of at least step 8, such as our example solution based on countering enemy reconnaissance units. Steps 5 through 8 are revisited for each possible solution path that the expert identifies and selects to pursue in agent-training episodes.

The key in this step is that the expert performs this modeling by analyzing a specific COA and anticipates that available training data would include examples of the solutions to be modeled subsequently. Direct expert-agent interactive-training episodes require concrete examples of problem solving by the expert with available data. If the expert wants to teach the agent that a combination of A and B, with specific conditional criteria implies a conclusion of C, the expert must be able to provide the agent with data that match these conditions. Consequently, the expert initially may not be able to teach the agent directly that the lack of certain conditions constitutes a negative example because of the inability to provide a concrete example of missing conditions.

This represents an important difference and limitation compared to the traditional approach of a KE creating rules to represent expert knowledge. In

the traditional environment, which Disciple also can support, the KE can directly craft and alter rules that cover both positive and negative cases, working in the absence of any supporting data. If concrete examples from this solution path are likely to be available for agent training, the expert proceeds to step 6.

6.6 Identify task-reduction steps

Steps 6, 7 and 8 may proceed concurrently, or in an iterative manner. This is the most difficult step in the procedure. The expert may begin this step with a clear idea that if certain things can be found in example data, a conclusion can be reached. The HPKB evaluation process provided a model answer that indicated that when enemy reconnaissance elements are destroyed as a precursor to an operation, the likelihood of achieving tactical surprise is enhanced. Reaching and stating these kinds of conclusions is easier than breaking the conclusion down into the knowledge elements and logical steps that allow us to reach the conclusions, but this step requires the latter.

When given a fairly complete conclusion statement as in this example, a single, complex reduction to an assertion statement is possible. That is, A follows logically, given elements X, Y, and Z. This is fairly typical in the traditional approach to expert-system development where the KE attempts to program expert knowledge as efficiently as possible. However, this approach actually appears to be counter productive in the long run. We have found that simpler, incremental steps yield much more flexibility to reuse steps and branch off in new directions when the expert identifies new potential solution paths or identifies the need for changes based on new data sets.

An assessment task is reduced successively to simpler assessment tasks and ultimately, to assertions on how the COA conforms to the principle of surprise. During this step, the expert begins to identify the critical elements of knowledge that are necessary to reach a conclusion. In the simplest form of this example, the expert tracks the specific COA for which this solution path is being generated.

6.7 Identify a question and answer

The expert frames an explanation of why one task on this solution path reduces to another in the form of a question to be considered, and an answer that justifies the particular reduction. Our objective at this point is to progress beyond the task of "assessing surprise with respect to enemy reconnaissance" only when enemy reconnaissance units are present in a scenario. Therefore, the expert

records a question and answer that fits the situation and includes them in the model. For example, a Q-A pair could be as follows: Question: Is an enemy reconnaissance unit present? Answer: Yes, enemy reconnaissance unit RED-CSOP1 is conducting the reconnaissance action SCREEN1.

The correctness of the question and answer is very important because they directly support ontology development and provide the basis for hints and explanations that the expert will provide the agent during the subsequent ontology development and training episodes as described in steps 9, 10 and 11.

The procedure described above is not initially likely to be entirely natural for an expert. Experts make unstated assumptions and take mental shortcuts when they make decisions. On the first pass at modeling a solution, assumptions and shortcuts are likely to cause omissions of critical knowledge elements. In this example, a military expert may not express the appropriateness of considering only enemy reconnaissance units in this context. This likely would cause faulty training of the agent and subsequently result in the agent generalizing and applying a rule that considered both friendly and enemy reconnaissance units when it critiqued a COA in the same context. Fortunately, our experimental results show that experts quickly learned the requirement to frame detailed and specific questions and answers because the consequences of the omissions rapidly became apparent during subsequent checking of the agents' performance. An important strength of this approach is that Disciple agents learn not only domain-specific problem solving from interaction with the expert, but eventually will learn to anticipate and deal with omissions by the expert. The agent learns to make the same assumptions and mental shortcuts that the expert makes, mostly by using analogy.

An additional, critical purpose of the questions and answers in the model is their use in the justification for the eventual assertion. Typically, computer users will challenge a solution provided by an automated system. In the case of this military domain, it would be surprising if an experienced military user did not challenge the validity of many answers because of the subjectivity of some facets of COA critiquing. The Disciple problem solver has been enhanced to store the questions and answers provided by the expert and create a variety of justification trees with them when a rule fires and results in an assertion of a strength or weakness in a COA. Furthermore, Disciple recognizes the presence of actual, exact instances in the questions and answers and generalizes them when they are stored.

When the subsequent justification tree is built for an assertion, Disciple replaces the generalized instance names with the actual instance names from the current data set into the justification. As a result, when the expert user wants an explanation of how the Disciple COA critique agent arrived at an assertion, with a mouse click or two, a justification is generated and displayed containing exact references to the current data set.

To support various degrees of experience in users, Disciple generates three versions of the justification trees with different levels of detail in the justification. A domain expert may be interested in just the basic questions and answers displayed in one version of the justification, whereas a KE may want the most detailed version that lists the rules that fired and the values that were assigned to variables in different states. Whereas this justification can explain how a correct conclusion was reached, the more powerful use of the justification is that it helps to determine where the agent went wrong and helps to focus corrective retraining when an assertion is incorrect.

7 Experimental trials and results

Our HPKB evaluation results are documented in [7] and [13]. In summary, these results show that Disciple-based agents built by teams of SMEs and KEs using early versions of the methodology, were highly effective in solving complex problems and produced very high knowledge-acquisition rates.

We have also conducted a knowledge-acquisition experiment at the US Army Battle Command Battle Laboratory at Fort Leavenworth, KS, in August 1999 [12]. The purpose of the experiment was to determine the extent to which SMEs with no knowledge-engineering experience could train Disciple to critique a COA. The experiment did not require the experts to use GMU's modeling and representation methodology to design solutions from "scratch." Rather, SMEs modified and used models prepared by other military experts for our HPKB evaluations, and tested whether these models were appropriate and sufficient to support SME attempts to develop and train Disciple-based agents.

The SMEs were four, active duty, US Army combat arms officers with 16 to 22 years of military service. The experts received about 16 hours of background-information briefings that included the following topics: 1. discussion of artificial intelligence, GMU's research goals and experiment design; 2. demonstrations and practice with Disciple and its interfaces; and 3. explanation and practice using and modifying the task-reduction models as a paradigm for problem solving.

During the experiment, the military SMEs each separately taught Disciple to critique a COA with respect to the principles of offensive and security. Starting with a conceptual modeling of the critiquing process for these two principles, they each, independently succeeded in developing a knowledge base in one day.

The training for the principle of offensive took place in the morning and, in the case of Expert #4, it consisted of 101 minutes of expert-Disciple interactions. During this time, Disciple learned 14 tasks and 14 rules. The training for security took place in the afternoon and consisted of 72 minutes of expert-Disciple interactions. During this time Disciple learned 14 tasks and 12 rules. The KE provided very limited training assistance.

The knowledge-acquisition rates obtained during the experiment were very high (~ 9 tasks and 8 rules/hour expert).

We consider this KA experiment to be one of the most significant accomplishments of our research. To our knowledge, it is the first time a SME with no prior knowledge engineering experience has succeeded in extending a significant knowledge base in a very short period of time, without any significant support from a KE.

8 Conclusion

Our experimental results show that we have developed a general-purpose methodology and tool for expert knowledge acquisition based on apprenticeship multi-strategy learning in a mixed-initiative framework. This methodology enhances the ability of a domain expert with very little knowledge engineering experience, to build a knowledge base efficiently. The Disciple methodology accomplishes six major functions: 1. helps the domain expert conceptualize the problem solving process; 2. allows an expert to document, order and justify their decision making process; 3. facilitates directly the expression of this information to the agent; 4. governs the entire agent training and knowledge base development process; 5. facilitates natural interactions that allow the agent to learn complex problem solving rules, extend and correct the domain knowledge base; and 6. Supports the reuse of existing ontologies and the extension of these ontologies for the problem domain.

Modeling and representation of expert knowledge are essential elements of solving the knowledge-acquisition bottleneck and making artificial intelligence truly useful in practical applications. The methodology described in this paper provides an organized, effective, and repeatable process for modeling how an expert might solve problems in a given domain, oriented to

specific example problems. This methodology has been used successfully to model problem solving in the military domain and it successfully supports the development of intelligent agents for military decision makers.

9 Directions for future research

The process of creating solution trees has been integrated into the Disciple shell. The elicitation of expert knowledge for the development of these diagrams can be assisted by a Disciple-based intelligent agent designed and trained for that purpose. Modeling and ontology-acquisition activities can be automated, particularly with regard to the direct capture of ontology elements consisting of concepts, objects and features. Disciple learned simple rules in the experiment described above; however, learning more complex rules, in the context of a more general representation language, will require improved methods.

Other research projects can focus on the expansion of Disciple to applications in other domains, such as the medical and manufacturing domains listed in Table 1. For example, a Disciple knowledge base could be augmented to include diagnostic rules and procedures, as well as information about diseases, their characteristics and statistics. In manufacturing, loss of expertise in a private company could be minimized by having experienced personnel enter their knowledge in the knowledge base on a regular basis. Thus, when skilled workers retire, the company could retain some of their knowledge. This facility could be used to train newly hired personnel, or current employees who want to upgrade their skills.

One of the most difficult and often vexing problems that senior military leaders face at the strategic level of war is the determination and analysis of the center of gravity for an opposing force. Clausewitz [4] introduced the concept of a center of gravity as "the hub of all power and movement, on which every thing depends". Military professionals have debated the meaning of Clausewitz's words for many years. As one of the RKF challenge problems, domain experts from the U.S. Army War College and researchers from the GMU LALAB are extending Disciple in an attempt to create intelligent agents that will support the identification of candidate strategic centers of gravity for historic and modern crisis and wartime scenarios.

Acknowledgments

The research on the Disciple approach was done in the GMU Learning Agents Laboratory and was sponsored by AFOSR under grant no. F49620-97-1-

0188 and grant no. F49620-00-1-0072, and by DARPA, AFRL, AFMC, USAF, under agreement number F30602-00-2-0546. The research of Marion Ceruti was sponsored by DARPA. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA, AFRL, AFOSR, or the U.S. Government. Mihai Boicu, Dorin Marcu, Cristina Cascaval and other members of the LALAB contributed to the development of Dicile-COA.

References

- [1] D. Bamber, Knowledge Acquisition for the Development of Expert Systems: An Analysis, Naval Ocean Systems Center *Technical Document 1322*, 54 pp., July 1980.
- [2] M. Burke, Rapid Knowledge Formation (RKF) Program Description, <http://dtsn.darpa.mil/iso/programtemp.asp?mode=331>
- [3] J.A. Cannon-Bowers, E. Salas, and J.S. Pruitt, "Establishing the boundaries of a paradigm for decision-making research," *Human Factors*, vol. 38, no.2, pp. 193-205, 1996.
- [4] C. von Clausewitz, (1832), *On War*, trans. M. Howard and P. Paret, p. 595, Princeton University Press, Princeton, NJ, 1976.
- [5] M.G. Ceruti, C. Anken, A.D. Lin, and S.H. Rubin, Applications of High-Performance Knowledge-Based Technology, *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, vol. 3 of 5, pp. 1965-1971, October 2000.
- [6] M.G. Ceruti and M. Bowman, *Expertise, Pattern Management and Decision Making: Challenges in Human Informatics*, International Institute of Informatics and Systemics (IIIS) Fifth World Multi-conference on Systemics, Cybernetics and Informatics, (SCI'2001), July 2001.
- [7] P. Cohen, R. Schrag, E. Jones, A. Pease, A. Lin, B. Starr, D. Easter, D. Gunning, and M. Burke, The DARPA High Performance Knowledge Bases Project, *AI Magazine*, vol. 19, no.4, pp.25-49, Winter 1998.
- [8] A. Kirlik, N. Walker, A.D. Fisk, and K. Nagel, Supporting perception in the service of dynamic decision making. *Human Factors*, vol. 38, no.2, pp. 288-299, 1996.
- [9] S.H. Rubin, M.H. Smith, and L. Trajkovic, Randomizing the Knowledge Acquisition Bottleneck, *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pp. V-790 – V-795, 1999.
- [10] K. Schreiner, HPKBs and Beyond, *IEEE Intelligent Systems and Their Applications*, vol. 14, no. 2, pp. 53-63, March/April 1999.
- [11] G. Tecuci, 1998. *Building Intelligent Agents: An Apprenticeship Multistrategy Learning Theory, Methodology, Tool and Case Studies*. London, England: Academic Press.
- [12] G. Tecuci, M. Boicu, M. Bowman, D. Marcu, P. Shyr, and C. Cascaval, An Experiment in Agent Teaching by Subject Matter Experts, *International Journal of Human-Computer Studies* 53:583-610, 2000.
- [13] G. Tecuci, M. Boicu, M. Bowman, and D. Marcu, with a preface by M. Burke, An Innovative Application from the DARPA Knowledge Bases Programs: Rapid Development of a Course of Action Critiquer, *AI Magazine*, vol. 22, no.2, Summer 2001.
- [14] US Army Field Manual, FM100-5, *Operations*, Washington D.C., 1993.