

Intelligent Agents in the Command Post

Lieutenant Colonel Michael Bowman, US Army;
Gheorghe Tecuci; and Mihai Boicu

A COMMANDER'S ABILITY to make good decisions under adverse conditions can decide a battle. Seventy-two hours into an operation, he may be running on catnaps and caffeine.

At the tactical operations center, the operations officer briefs three possible courses of action (COAs) for a new mission. The operations officer recommends the first COA with conviction, then explains its advantages. Something about the COA is troubling the commander. Perhaps fatigue, stress or some other distraction is preventing him from recalling the lesson learned from other battlefields or training events that would affect his decision. The commander makes his decision but has a feeling that there is a better way—he needs more time, more information or a clearer head.

The decisions commanders, staff officers and warfighters make are driven by education, training, experience and personal preference. A lifetime of learning allows them to set goals, assess conditions, identify and evaluate alternatives, and make quick, complex decisions. The human mind can make remarkable decisions under extraordinary conditions—an ability technology lacks. However, the mind is subject to adverse effects by environmental factors such as fatigue, stress and hunger. It is well documented that decisions made under these conditions are generally inferior to those made by well-rested, fed, comfortable and relaxed leaders. The US military strives to select commanders based on their proven ability to make good decisions under adverse conditions.

The opening scenario provides a good example of decision making under adverse conditions. Consider the same adverse conditions—with another decision-making tool for commanders.

The intelligent agent combines the things a computer does best—sorting and sifting through massive data while remaining immune to fatigue, stress or environmental factors. . . . It provides concise, relevant and explainable information the commander can consider when making decisions. This is how we can use intelligent agents in the command post of the future.

A computerized intelligent agent is programmed with lessons learned from classrooms and training. The software runs on the same computers that are used to type memos and create briefing slides. It accesses the data in the battle command systems and planning tools, including powerful networked computer systems, handwritten notes and sketches.

The intelligent agent is oblivious to time, temperature and a commander's physical and mental condition. Based on all available information, it evaluates the COAs and lists each strength, weakness and issue in a matter of seconds. The commander quickly scans the list, discards some COAs and considers others until he reaches the element critical to the decision. Based on his judgment, his staff's recommendations and a few key factors that he might otherwise not have had the energy or awareness to identify, he decides.

The strengths, weaknesses or issues might be based on planning considerations learned by a lieutenant in basic course; discovered by a captain during an after-action review; or noted during senior service college. They might be based on new enemy tactics observed during a battle yesterday. The intelligent agent combines the things a computer

does best—sorting and sifting through massive data while remaining immune to fatigue, stress or environmental factors—with the things a human can do best, such as learning from experience. The agent combines doctrine and tactics with lessons learned throughout military history. It does not replace the commander or make the commander's decisions; it provides concise, relevant and explainable information the commander can consider when making decisions. This is how we can use intelligent agents in the command post of the future.

Learning Agents Laboratory

Automated decision support systems, expert systems and intelligent agents are not new but have played a limited role in military command and control or support systems. Even with today's rapid growth in computing power and connectivity, most software products that claim to be intelligent do not solve complex, real-world problems. Government, industrial and academic researchers have made recent progress in moving intelligent systems from hype to reality.

A novel approach to creating and using intelligent agents to solve complex military problems is under way at the George Mason University (GMU) Learning Agents Laboratory (LALAB). The research goal is to develop methods and tools that will allow users with minimal computer skills to build, teach and maintain intelligent software agents easily.¹ This approach allows users to teach the agent to perform tasks as they would teach an apprentice or student. The agent is given examples and explanations; then its behavior is supervised and corrected.

This research was done as part of the Defense Advanced Research Projects Agency (DARPA) High-Performance Knowledge Base (HPKB) project with additional support from the US Air Force Office of Scientific Research and the US Army Battle Command Battle Laboratory. The HPKB project's goal was to produce the technology for rapidly constructing large knowledge bases that comprehensively cover topics of interest, are reusable by multiple applications with a variety of problem-solving strategies and are maintainable in rapidly changing environments.²

The organizations participating in HPKB were challenged to solve knowledge-based problems in a particular domain and then modify their systems quickly to solve further problems in the same domain. The exercise tested the claim that large knowledge bases can be built quickly and efficiently with the latest artificial intelligence (AI) technology. The

When asked to explain a decision, most people point out one or two fragments of key information. They generally find it difficult to generate quickly a thorough list of salient facts or establish the relationships between those facts. Furthermore, thoroughly explaining how one reached a specific decision often falls far short of providing a general solution.

work continues as part of the DARPA Rapid Knowledge Formation project with Air Force and Army support. DARPA's independent evaluations and experiments at the BCBL show that with the right approach, intelligent agents can solve complex real-world problems and be rapidly built and easily maintained.

The Traditional Approach and Bottleneck

Whether the domain is nuclear physics or grocery shopping, we are all experts at something and continually make decisions, sometimes instantly. We learn this from birth so that our decision-making acumen is based on experience, training and habits of a lifetime. Recognizing and demonstrating the ability to make rapid, sound decisions is easy; capturing and explaining how and why we make the decisions is more difficult. Bad decisions are generally no easier to explain and document than good decisions. When asked to explain a decision, most people point out one or two fragments of key information. They generally find it difficult to generate a thorough list of salient facts quickly or establish the relationships between those facts. Furthermore, thoroughly explaining how one reached a specific decision often falls far short of providing a general solution.

Consider the task of buying a loaf of bread. Take a moment to capture all of the information necessary to complete the task:

- Funds to purchase the bread.
- A place to purchase the bread.
- Transportation to reach the bread.
- The type of bread.
- The method of payment used for the purchase.

What about "paper or plastic?" We almost never complete the list on the first try, and neither will experts when they attempt to capture all of the steps and factors in complex problem solving.

A major barrier to building intelligent systems that solve human problems is called the "knowledge-

acquisition bottleneck," which results when transferring knowledge from the expert to the computer. The traditional approach to knowledge acquisition and agent development involves interaction between

A major barrier to building intelligent systems that solve human problems is called the "knowledge-acquisition bottleneck," which results when transferring knowledge from the expert to the computer.

Our goal was to create a tool in the Disciple COA critique agent that contained a common understanding of principles and tenets but was flexible enough to allow the experts training and using the system to personalize them rapidly.

a trained knowledge engineer and a domain expert. The knowledge engineer must learn what the expert knows and how the expert uses that knowledge to solve problems. The knowledge engineer uses various tools and techniques but relies primarily on personal observation and interview.

Indirectly transferring knowledge from an expert through a knowledge engineer and finally to a computer system is particularly difficult because experts express their knowledge differently from the way it must be represented in the intelligent system. After providing background knowledge, the expert must verify and correct it where appropriate. Such an indirect transfer makes acquiring usable knowledge slow, painful and inefficient.

Consider the development of a hypothetical expert system. Joe is a nuclear engineer, has worked at his profession for 30 years and is his plant's leading expert. Plant managers hire an information technology (IT) firm to develop an agent to assist, or even replace, Joe. The IT firm assigns a knowledge engineer to elicit, document and formalize Joe's knowledge and problem-solving process. The knowledge engineer takes a year to understand and capture what Joe knows. The captured knowledge summarizes that when the red light on his control panel flashes, he presses button number five to prevent the plant from exploding. The knowledge engineer then takes another year to complete the knowledge base that represents Joe's knowledge and to program and debug a system that uses that knowledge to solve appropriate problems.

After the second year, the knowledge engineer returns to the nuclear plant with his completed system and a bill for two years' work. To his surprise, he finds that Joe has retired, and the plant has updated Joe's control panel with a version that no longer includes a red flashing light or button number five. The system he has spent two years building is irrelevant. In the end the system is unused, the plant has wasted time and money, and the IT firm's reputation has taken a blow. Everyone involved with this kind of system knows that there has to be a better way. Acknowledged weaknesses in this traditional approach include:

- Limited ability to reuse previously developed knowledge.
- Slow, inefficient knowledge acquisition.
- Slow knowledge adaptation.
- Limited scalability in building agents.³

Acquiring Knowledge for Decision Making

How can a subject matter expert (SME), with no knowledge engineering experience and very limited support from a knowledge engineer, train an intelligent agent to solve problems? To answer this question we developed the Disciple theory, methodology and learning shell, in which the SME teaches the agent to perform various tasks that an expert would teach an apprentice. The agent learns directly from the expert, building and refining its knowledge base. Over several years increasingly advanced intelligent agents have emerged from the Disciple family.⁴

The knowledge base of a Disciple agent consists of an ontology that defines the terms from the application domain and a set of rules expressed with these terms. The Disciple strategy is to replace the difficult engineering tasks required to build a knowledge base with simpler tasks that the SME can perform.

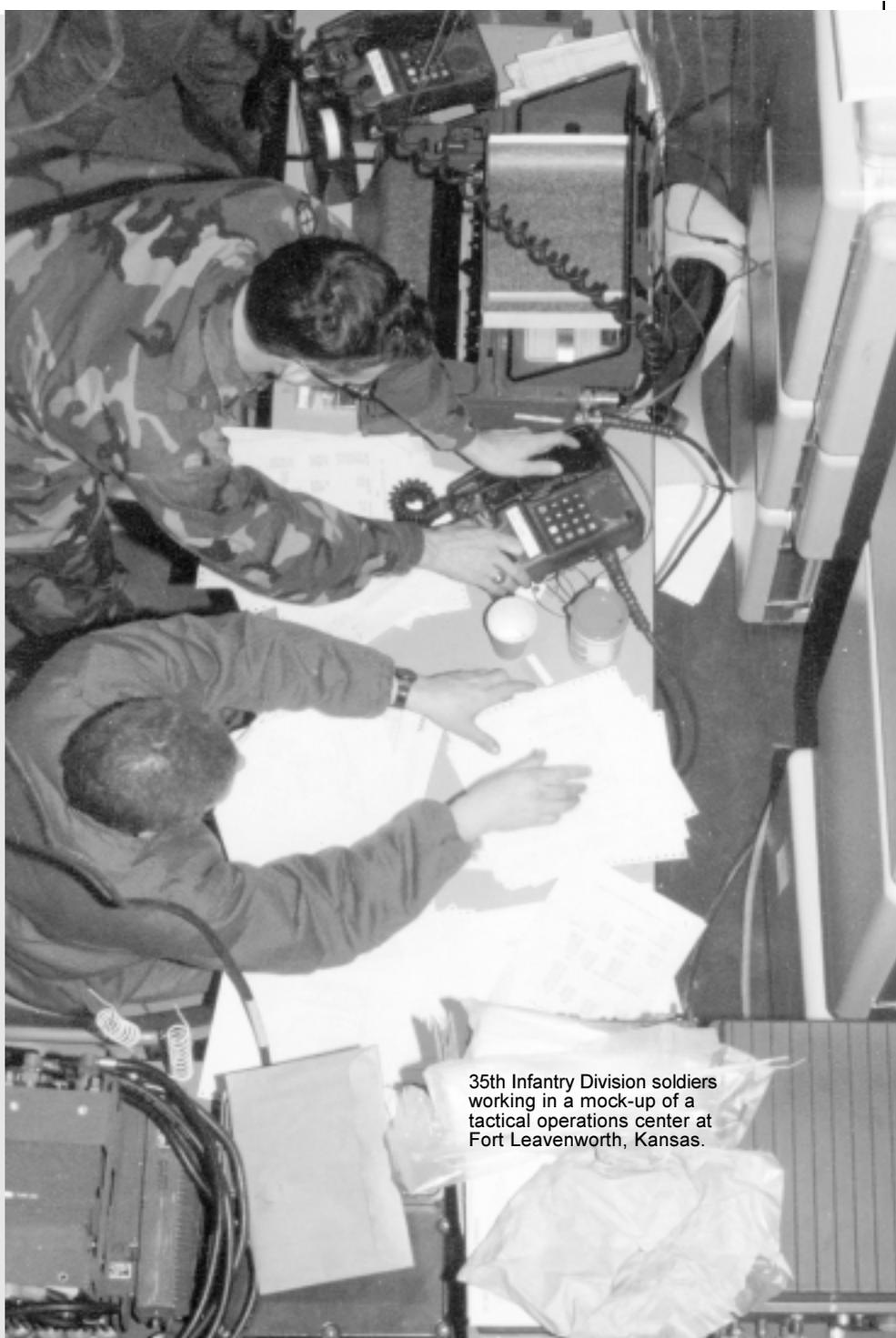
Building a knowledge base requires creating an ontology that defines the application domain, defining problem-solving rules or methods, then verifying and updating them. These tasks require creating formal sentences and formal explanations. In the Disciple approach, these tasks are replaced with simpler tasks that an SME can perform with limited support from a knowledge engineer. Instead of creating an ontology, the expert need only update and extend an initial ontology imported from existing repositories of knowledge. Instead of defining a complex problem-solving rule, the ex-

pert only defines a specific example of a problem-solving episode because Disciple will learn a rule from that example. Instead of debugging a complex problem-solving rule, the expert critiques specific examples of problem-solving episodes, and Disciple will update the corresponding rule accordingly. Most of the time, the expert will not need to create formal sentences, just understand the sentences Disciple generates. The expert will not need to provide formal explanations, just informal hints so Disciple can generate plausible explanations. The expert chooses the correct ones.

Disciple

A challenge for the second phase of HPKB was critiquing COAs for ground combat operations. The participating teams used a common ontology that the research organizations participating in HPKB developed. This was a new challenge since we had used only ontologies Disciple had developed directly, which were particularly well suited to our learning and knowledge-acquisition methods.

The US Army provided the COAs in a standard military format consisting of a multi-paragraph description and a graphic representation of the COA using standardized symbols for units, activities and geospatial relationships. The Disciple agent identifies COA strengths and weaknesses with respect to the principles of war and tenets of Army operations.⁵ To develop the agent's knowledge base, we performed a task-based modeling of these principles and tenets. Task-based modeling relates military units to their assigned tasks by determining whether tasks are appropriate for the unit type, size or condition, or whether completion



35th Infantry Division soldiers working in a mock-up of a tactical operations center at Fort Leavenworth, Kansas.

[Intelligent agent] software runs on the same computers that are used to type memos and create briefing slides. It accesses the data in the battle command systems and planning tools, including powerful networked computer systems, hand-written notes and sketches.

of the tasks contributes to mission success.

A common understanding of the principles of war and tenets of operations is well-documented in military literature. US Army Field Manual (FM) 100-5, *Operations*, states: "At all levels of war, successful application of maneuver requires agility of

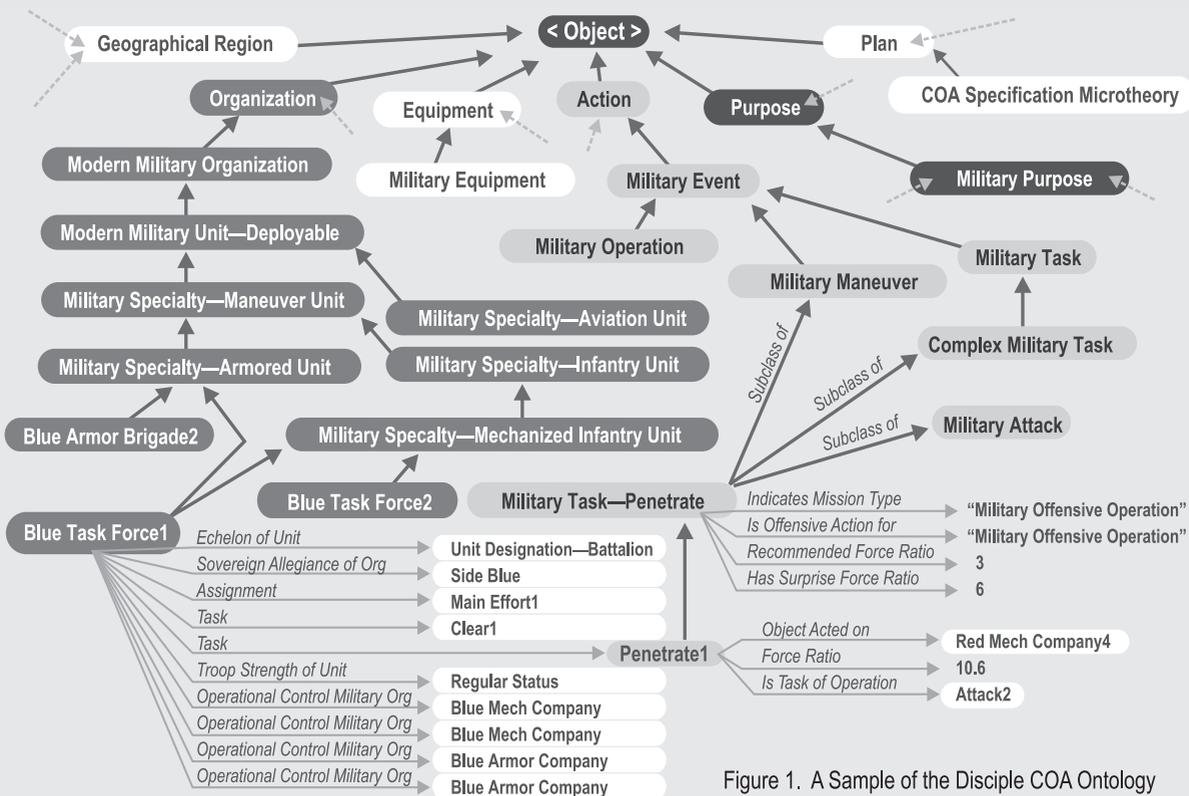


Figure 1. A Sample of the Disciple COA Ontology

thought, plans, operations and organizations.”⁶ Experts disagree on the interpretation and application of such a phrase, and the principles and tenets are just the beginning of a critique. Our goal was to create a tool in the Disciple COA critique agent that contained a common understanding of the principles and tenets but was flexible enough to allow the experts who train on and use the system to personalize them rapidly.

Building Disciple Agents

We have developed a methodology for an SME to build an intelligent agent that has the following characteristics:

- Natural to the expert providing the information.
- General enough to use in a variety of domains.
- Thorough enough to support teaching-based ontology development.
- Flexible enough to support explanation generation and multistrategy learning.

While Disciple includes the complete set of knowledge-engineering tools necessary to develop and train intelligent agents, common sense and experiments show that developing agents that solve complex problems is most effective and efficient when the agent already possesses background knowledge. For example, if we train an agent to

identify whether a COA contains the element of surprise, we do not explain what a tank or a tank company is. Background knowledge is available from multiple sources, and Disciple can import this knowledge so a domain expert does not start from scratch to train agents. When an agent-training episode requires providing new background knowledge, a suitable Graphical User Interface allows the expert to do so.

Background knowledge, such as unit hierarchy, organization and equipment, was imported from the CYC knowledge base for Disciple to use. Information for each training or critique came from the COA sketch and text.⁷ The Artificial Intelligence Applications Institute and Northwestern University participated in the HPKB project and developed the tools that converted the sketch and text to data files. This information is collectively called an ontology. A Disciple ontology includes objects, features and tasks, all represented as frames, according to the Open Knowledge Base Connectivity knowledge model.⁸ A domain expert training a Disciple agent is not required to work directly with the ontology structure or knowledge base, but generally understanding the structure and information in the knowledge base makes the expert a more efficient trainer.

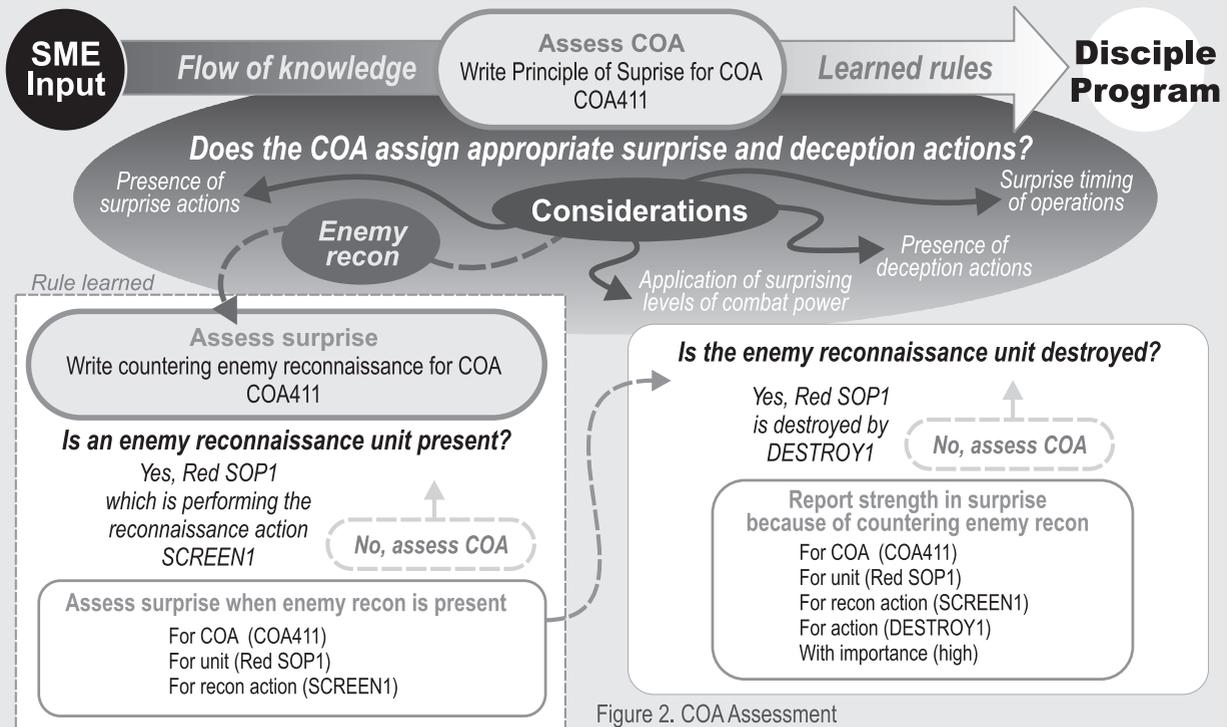


Figure 2. COA Assessment

The figure presents a fragment of the object ontology used to model the COA domain. The upper part of the figure represents the top level of the object ontology that identifies the types of concepts represented. The left part shows a fragment of the hierarchy of the modern military organization. The levels of this hierarchy are specific military units, corresponding to a specific COA to be critiqued by Disciple. Each concept and instance of the object hierarchy is described by specific features and values.

Our methodology for explaining and documenting decision making or problem solving to a Disciple agent is task reduction. Task reduction takes a complex problem and reduces it into a series of simpler tasks until what remains is a problem that contains enough information to reach a conclusion.

Whether teaching a human or a machine to solve a problem, teachers must know how to solve a problem before they can explain the process. Make no mistake about it—analyzing, documenting and checking decision-making details is not a natural process for experts unless they also routinely teach aspiring experts. Teaching a Disciple agent consists of identifying the sequence of steps (task reduction) to Disciple and explaining when and why you move from one step to the next. The explanations could

The expert defines a specific example of a problem-solving episode because Disciple will learn a rule from that example. Instead of debugging a complex problem-solving rule, the expert critiques specific examples of problem-solving episodes, and Disciple will update the corresponding rule accordingly.

be a series of considerations or questions to ask at each step and the corresponding answers that would lead to the next step (task). The questions and answers are expressed in natural language, and when combined with a diagram representing the problem-solving steps, they serve as a script for the interaction between the expert and Disciple during an agent-training episode.

Disciple learns a general plausible version space task-reduction rule as the expert indicates each task-reduction step (represented by a task, a question, its answer and a subtask). This complex “if-then” structure indicates the conditions under which the task from the “if” part of the rule can be reduced to the task (or tasks) from the “then” part of the rule. However, instead of a single applicability condition, a rule specifies a plausible space of hypotheses for its condition. A plausible upper-bound condition and

a plausible lower-bound condition represent this plausible version space. During further learning from the expert, these two bounds will converge. In addition to conditions necessary for the rule to apply, the rule may have several “except when” conditions that should not hold for the rule to apply. The rule may also have “except for” conditions (that specify instances that are negative exceptions to the rule) and “for” conditions (that specify positive exceptions). Much of the power of the Disciple approach comes from the plausible version space rule.

The expert never has to see or deal with these problem-solving rules, only with specific examples. In the traditional approach, experts must explain this reasoning to knowledge engineers who manually encode this knowledge into problem-solving rules and debug them. With Disciple, the expert directly teaches the agent that learns and refines such rules by itself.

Training a Disciple agent is an iterative process of providing examples and explanations to the agent, letting the agent attempt to solve problems and explaining why its solutions are right or wrong. Using the Graphic User Interface, the expert identifies the problem to be solved. Disciple will attempt to solve the problem by applying plausible version space rules that reduce the problem to simpler ones. The expert examines these solutions and has three options. If the expert does not find a correct solution, he provides a solution and helps the agent to understand it, and Disciple learns a new plausible version space rule. If the expert finds a suitable solution among those proposed, Disciple generalizes the rule that produced that solution. If the expert finds an incorrect solution, he rejects it and helps Disciple to understand why the solution is wrong. This helps Disciple isolate the rule that produced the wrong solution so it does not generate such a solution in the future. This way, Disciple continuously learns from the expert while extending and improving its knowledge base.

One of the major strengths of the Disciple approach is that the expert does not have to conduct perfect or comprehensive agent training. Training flaws show up immediately when Disciple solves problems on its own, and the expert merely examines Disciple's solutions and explains where it went wrong. Extremely important, Disciple learns through example. Training an agent to critique COAs requires examples of COAs—both an advantage and disadvantage. The obvious disadvantage is

Disciple learns a general plausible version space task-reduction rule as the expert indicates each task-reduction step. This complex “if-then” structure indicates the conditions under which the task from the “if” part of the rule can be reduced to the task (or tasks) from the “then” part of the rule. . . . In the traditional approach, experts must explain this reasoning to knowledge engineers who manually encode this knowledge into problem-solving rules and debug them.

that example data is required. The advantage is that most experts naturally examine an example and offer explanations based on that example.

Experimental Results

The Disciple methodology and the developed Disciple agents were tested with other systems as part of DARPA's annual HPKB program evaluations. The experiment results show that the Disciple-based agents, built by teams of experts and knowledge engineers, were highly effective in solving complex problems and produced high knowledge-acquisition rates, outperforming the other systems developed in HPKB for the same challenge problems.⁹

After July 1999 we concentrated our efforts on designing and conducting a one-week knowledge-acquisition experiment at the US Army Battle Command Battle Lab, Fort Leavenworth, Kansas. The experiment demonstrated that military experts can directly teach Disciple agents how to critique a COA using several principles of war and tenets of Army operations.

Our SMEs were four US Army, active duty combat arms officers with 16 to 22 years of military service, who had previous knowledge-engineering experience. The Discipline experiment contained three phases—a training phase for the experts (days 1-3), an experiment phase (day 4) and a joint discussion of the experiment (day 5). The expert training phase presented the purpose of the experiment, the history and status of AI, structure and content of the COA ontology, and practical exercises using Disciple.

During the experimental phase, the officers trained Disciple agents to critique COAs on principles of the offense and security, starting with a knowledge base containing the complete ontology of objects and features—but no rules. Their train-

Developing agents that solve complex problems is most effective and efficient when the agent already possesses background knowledge. For example, if we train an agent to identify whether a COA contains the element of surprise, we do not explain what a tank or a tank company is.

ing episodes with the agents lasted approximately three hours, and they did not receive any significant assistance from knowledge engineers. All experts extended the knowledge base of Disciple-COA with 28 tasks and 26 rules, following a model for COA critiquing they received at the beginning of the experiment. At the end of the experiment, their surveys gave high scores to Disciple's usability and usefulness.

This experiment was the biggest accomplishment of our research in 1999. It is the first time an SME with no prior knowledge-engineering experience has trained an intelligent agent to solve a com-

plex problem and extended a significant knowledge base. Moreover, this occurred quickly, without a knowledge engineer's significant support.

As part of DARPA's HPKB program, the Disciple approach to intelligent-agent development was conducted and concluded that:

- It significantly speeds up the process of building and updating an intelligent agent and its requisite high-performance knowledge base.
- It enables domain experts to rapidly learn problem-solving knowledge with limited assistance from knowledge engineers.
- It provides problem-solving knowledge sufficient for the agent to generate highly correct solutions.

Our long-term vision is to develop a capability that will allow typical computer users to build and maintain intelligent agents and knowledge bases as easily as they use personal computers for text processing or electronic mail. This research intends to change intelligent agents from being programmed by a knowledge engineer to being taught by an SME. 🐼

NOTES

1. Gheorghe Tecuci, *Building Intelligent Agents: An Apprenticeship Multistrategy Learning Theory, Methodology, Tool and Case Studies* (San Diego, CA: Academic Press, 1998).
2. Paul Cohen et al., "The DARPA High-Performance Knowledge Bases Project," *AI Magazine* (Winter 1998), 25-49.
3. Tecuci.
4. Ibid.; Technical Aspects of Disciple are available at <www.lalab.gmu.edu>.
5. Ibid.
6. US Army Field Manual 100-5, *Operations* (Washington, DC: US Government Printing Office, 1993).

7. Douglas B. Lenat, "CYC: A Large-Scale Investment in Knowledge Infrastructure," *Communications of the ACM* (November 1995), 33-38.
8. Vinay Chaudhri et al., "OKBC: A Programmatic Foundation for Knowledge Base Interoperability," in *AAAI-98 Proceedings* (Menlo Park, CA: American Association for Artificial Intelligence [AAAI] Press, 1998), 600-607.
9. Michael Bowman, Gheorghe Tecuci and Mihai Boicu, "A Methodology for Modeling and Representing Expert Knowledge that Supports Teaching-Based Intelligent Agent Development," in *Proceedings of the Seventeenth National Conference on Artificial Intelligence and the Twelfth Conference on Innovative Application of Artificial Intelligence* (Menlo Park, CA: AAAI Press, July 2000).

Lieutenant Colonel Michael Bowman, US Army, is a student at the US Army War College, Carlisle Barracks, Pennsylvania. He received a B.S. from Ouachita Baptist University and an M.S. from the US Naval Postgraduate School and is a Ph.D. candidate at George Mason University. He has held various command and staff positions, including product manager, Communications and Intelligence Support Systems, Fort Belvoir, Virginia; chief, Automation and Software, Deputy Chief of Staff for Research, Development and Acquisition, Headquarters, US Army Materiel Command, Alexandria, Virginia; and joint systems integration officer, Defense Intelligence Agency, Washington, DC.

Gheorghe Tecuci is professor of computer science and director, Learning Agents Laboratory, Department of Computer Science, George Mason University, Fairfax, Virginia. He received an M.S. from Polytechnic University of Bucharest, Romania. He has earned Ph.D. degrees from the University of Paris-South, France, and the Polytechnic University of Bucharest. He has served in various research and staff positions, including associate professor of computer science, School of Information Technology and Engineering, George Mason University; and senior researcher, Research Institute for Informatics, Bucharest, Romania. He has published more than 100 scientific papers and five books on artificial intelligence.

Mihai Boicu is a Ph.D. candidate at the School of Information Technology and Engineering, George Mason University. He received an M.S. from the University of Bucharest, Romania. He has served in various research and staff positions, including graduate research assistant, School of Information Technology and Engineering, George Mason University; and tenured teacher, Computer Science Department, Research Institute for Informatics, Bucharest, Romania. He has published more than 15 papers on artificial intelligence.