In Jain L., (ed.), Advanced Information Systems in Defense and Related Applications, Springer Verlag, 2002.

9 Military Applications of the Disciple Learning Agent

To Mihai Draganescu, at his 75th anniversary.

G. Tecuci[1,2] and M. Boicu[1]

[1] Learning Agents Laboratory, Computer Science Department, MS 4A5, George Mason University, 4400 University Dr, Fairfax, VA 22030, USA
[2] Center for Strategic Leadership, US Army War College, 650 Wright Ave, Carlisle Barracks, PA 17013, USA

9.1 Introduction

This chapter presents an overview of the Disciple approach and its military applications. Disciple is a theory, methodology, and family of agent shells for the development of intelligent agents by subject matter experts, with limited assistance from computer scientists or knowledge engineers [1]. A subject matter expert interacts directly with a Disciple agent, to teach it to solve problems, in a way that is similar to how the expert would teach a human apprentice, by giving the agent examples and explanations, as well as by supervising and correcting its behavior. The agent learns from the expert by generalizing the examples and the explanations to build its knowledge base. The Disciple approach integrates methods for mixed-initiative problem solving, teaching, and multistrategy learning, exploiting the complementariness between human and automated reasoning, and creating a synergism between the expert that has the knowledge to be formalized and the agent that knows how to formalize it.

In the last few years the development of the Disciple approach has been a direct result of its application to three military challenge problems used in the DARPA's "High Performance Knowledge Bases" and "Rapid Knowledge Formation" programs [2, 3]:

1. The Workaround challenge problem - planning how a convoy of enemy vehicles can circumvent or overcome obstacles in their path, such as damaged bridges or cratered roads, in order to perform target selection [4].
2. The Course of Action challenge problem - critiquing military courses of actions with respect to the principles of war and the tenets of army operations, in order to assist a military commander to select the best course of action [5, 6].
3. The Center of Gravity challenge problem - identifying and testing strategic center of gravity candidates in military conflicts [7, 8].

This chapter introduces each of these innovative military applications of Artificial Intelligence, and the corresponding Disciple agent that was built to address them.

9.2 Center of Gravity Analysis

9.2.1 The Center of Gravity Problem

The military literature distinguishes between three levels of conflicts: a strategic level focusing on winning wars, an operational level focusing on winning campaigns, and a tactical level focusing on winning battles [9, 10]. One of the most difficult problems that senior military leaders face at the strategic level is the determination and analysis of the centers of gravity for friendly and opposing forces. Originally introduced by Clausewitz in his classical work "On War" [11], the center of gravity is now understood as representing "those characteristics, capabilities, or localities from which a military force derives its freedom of action, physical strength, or will to fight" [9]. It is recognized that if a combatant eliminates or influences the enemy's strategic center of gravity, then the enemy will lose control of its power and resources and will eventually be defeated. Similarly, if the combatant does not adequately protect his own strategic center of gravity, he will be defeated [12]. In spite of the apparently simple definition of the center of gravity, its determination requires a wide range of background knowledge, not only from the military domain, but also from the economic, geographic, political, demographic, historic, international, and other domains. In addition, the situation, the adversaries involved, their goals, and their capabilities can vary in important ways from one scenario to another. When performing this analysis, experts rely on their own professional experience and intuitions, without following a rigorous approach.

Correctly identifying the centers of gravity of the opposing forces is of highest importance in any conflict. Therefore, in the education of strategic leaders at all the US senior military service colleges, there is a great emphasis on the centers of gravity analysis [13]. Recognizing these difficulties, the Center for Strategic Leadership of the US Army War College started in 1993 an effort to elicit and formalize the knowledge of a number of experts in center of gravity. This research resulted in a COG monograph [12], which provided a basis for the application of the Disciple approach to this high value application domain, and to the development of the Disciple-RKF/COG instructable agent. Disciple-RKF/COG is used in a sequence of two courses taught regularly at the US Army War College, "Case Studies in Center of Gravity Analysis," and "Military Applications of Artificial Intelligence." In the first course (the COG course), the students become familiar with Disciple-RKF/COG as end-users, using it as an aid for learning about center of gravity analysis, and for developing a report containing a case study analysis. In the second course (the MAAI course), the students use Disciple-RKF/COG as subject matter experts, teaching it their own problem solving expertise in center of gravity analysis. In the next section we will illustrate the use of Disciple in these two courses.

9.2.2 A Disciple Agent for COG Analysis

In the "Case Studies in Center of Gravity Analysis" course, a personal copy of Disciple guides the student to identify, study and describe the aspects of a scenario (such as the 1945 US invasion of the island of Okinawa) that are relevant for COG analysis. The student-agent interaction takes place as illustrated in Figure 9.1. The left part of the window is a table of contents, whose elements indicate various aspects of the scenario. When the student selects one such aspect, Disciple asks specific questions intended to acquire from the student a description of that aspect, or to update a previously specified description. All the student's answers are in natural language. Taking the Okinawa_1945 scenario as our example, Disciple starts by asking for a name and a description of the scenario, and then asks for the opposing forces. Once the student indicates Japan_1945 and US_1945 as opposing forces, Disciple includes them in the table of contents, together with their characteristics that the student needs to specify (see the left hand side of Figure 9.1). Then, the student may click on any of these aspects (e.g. "Industrial capacity" under "Economic factors" of Japan_1945) and the agent guides the student in specifying it. The student's specification may prompt additional questions from Disciple, and a further expansion of the table of contents. An orange, yellow, or white circle marks each title in the table of contents, indicating respectively that all, some, or none of the corresponding questions of Disciple have been answered. The student is not required to answer all the questions and Disciple can be asked, at any time, to identify and test the strategic center of gravity candidates for the current specification of the scenario.

The top part of Figure 9.2 shows the solution viewer of Disciple that was customized for the COG domain. In the left hand side Disciple lists the strategic center of gravity candidates that it has identified based on the description of the Okinawa_1945 scenario. In the case of Japan_1945, these strategic COG candidates are Emperor Hirohito, Japanese Imperial General Staff, the military of Japan, and the industrial capacity of Japan. When a candidate is selected in the left hand side of the viewer, its (abstract or detailed) justification for identification or for testing can be displayed in the right hand side of the viewer. The top part of Figure 9.2 shows the abstract justification for the identification of Emperor Hirohito as a strategic COG candidate, and consists of a sequence of questions and answers. Emperor Hirohito is identified as a strategic COG candidate for Japan_1945 in the Okinawa_1945 scenario because Japan is a single-member force and Emperor Hirohito is the main controlling element of its government. After being identified as a candidate, Emperor Hirohito is analyzed based on various elimination tests, but he passes all of them. This testing is shown in bottom part of Figure 9.2. It has already been established that Emperor Hirohito controls the government of Japan. Because he is the commander in chief of the military, he can also impose his will on the military of Japan. Moreover, Emperor Hirohito could also impose the people of Japan to accept the unconditional surrender of Japan, which is the main strategic goal of the US. Being able to impose his will on the Clausewitz's trinity of power (government, military and

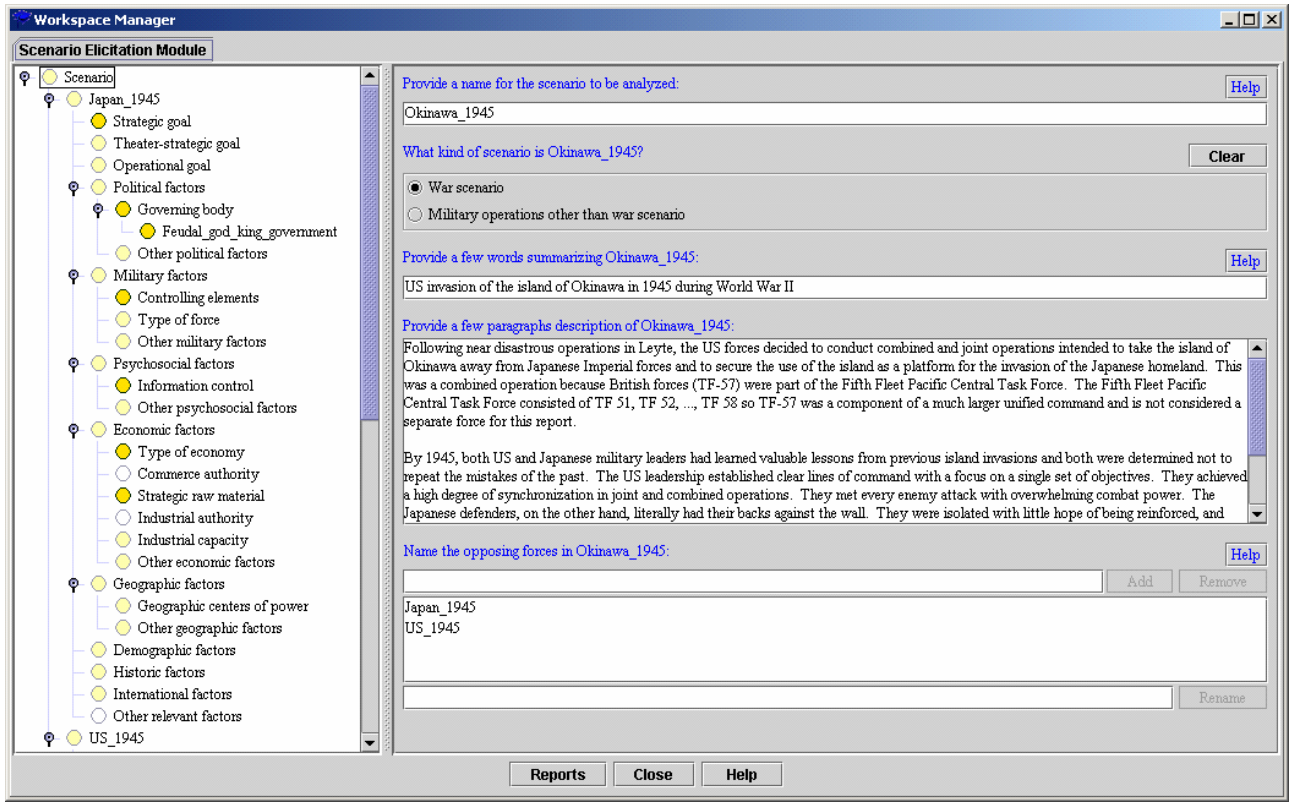


Fig. 9.1. Scenario specification interface

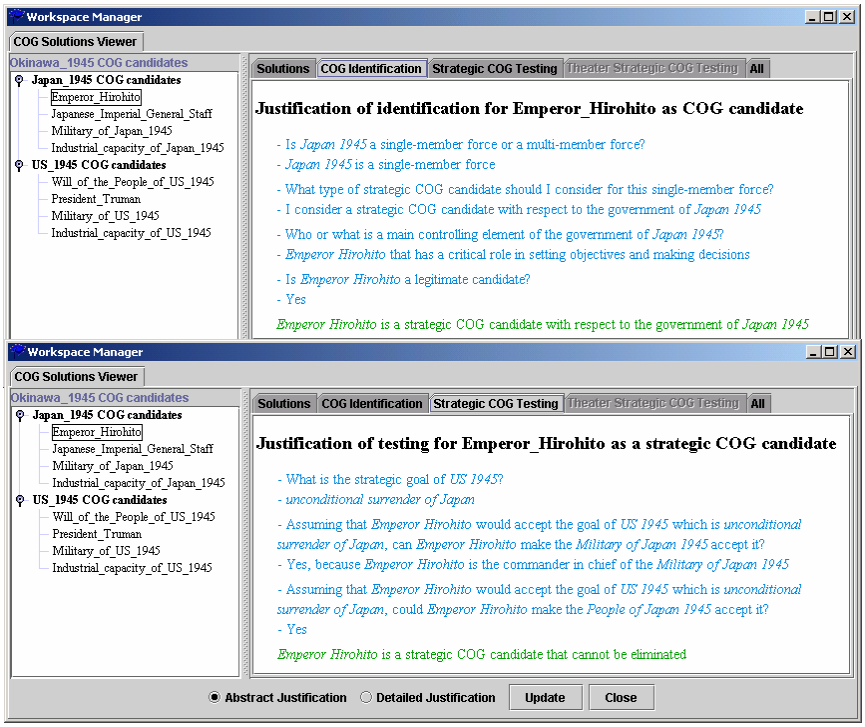


Fig. 9.2. Solution viewer interface

people), Emperor Hirohito is very likely to be the strategic center of gravity of Japan in 1945.

As another example, consider the industrial capacity of Japan_1945, which is another source of strength, power and resistance because it produces the war materiel and transports of Japan. Disciple, however, eliminates this strategic COG candidate because the military and the people of Japan_1945 are determined to fight to death and not surrender even with diminished war materiel and transports.

In the example scenario portrayed here, Disciple eliminates all but two candidates for Japan -- Emperor Hirohito and the Japanese Imperial General Staff -- and suggests that the student should select one of them as the strategic Center of Gravity of Japan in 1945. It is important to point out that this example is only a representative approach to the analysis of Japan's center of gravity for the Okinawa campaign. We recognize that subject matter experts often differ in their judgments as to the identification and analysis of center of gravity candidates for any particular scenario. The important point for agent development is that the Disciple agent can accommodate the preferences of the expert that teaches it, as will be discussed in the next section.

As briefly illustrated above, Disciple guides the student to identify, study and describe the relevant aspects of the opposing forces in a particular scenario. Then Disciple identifies and tests the strategic center of gravity candidates, as illustrated in Figure 9.2. After that Disciple generates a draft analysis report, a fragment of which is shown in Figure 9.3.

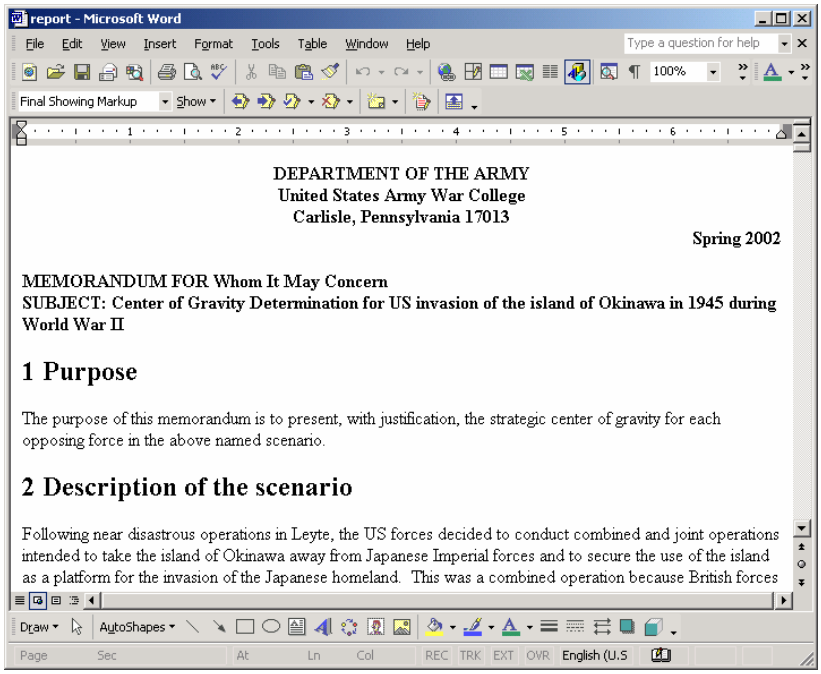


Fig. 9.3. Draft COG analysis report generated by Disciple

The student needs to finalize the report generated by Disciple. He is required to critically analyze Disciple's logic, correct or complete it, or even reject it and provide an alternative line of reasoning. This is productive for several reasons. First Disciple generates its proposed solutions by applying general reasoning rules and heuristics learned from another COG expert, to a new scenario described by the student. This scenario description incorporates the subjective judgments of the student, and may also be incomplete. Therefore the agent may not generate all the solutions that the expert would have generated, and some of the solutions and justifications provided by the agent may not always be entirely correct or complete. Secondly, as mentioned before, COG analysis is influenced by personal experiences and subjective judgments, and the student (who has unique military experience and biases) may have a different interpretation of certain facts.

This requirement for the critical analysis of the solutions generated by Disciple is an important educational component of military commanders that mimics

military practice. Commanders have to critically investigate several courses of actions proposed by their staff and to make the final decision on which one to use.

During the 2001 and 2002 academic years, Disciple was successfully used in both the Winter and Spring sessions of the COG course. As a result of this initial success, the USAWC decided to continue and expand the integration of Disciple in this course for the next academic year and beyond. At the end of the courses the students complete detailed evaluation forms about Disciple and its modules, addressing a wide range of issues, ranging from judging its usefulness in achieving course's objectives, to judging its methodological approach to problem solving, and to judging the ease of use and other aspects of various modules. For instance, on a 5-point scale, from strongly disagree to strongly agree, 9 students of the Winter-2002 session agreed and the other 4 strongly agreed with the statement "The use of Disciple is an assignment that is well suited to the course's learning objectives." To our knowledge, this is the first time that an intelligent agent for the strategic COG identification and testing has been developed.

The next section presents the use of Disciple in the "Military Applications of Artificial Intelligence" course.

9.2.3 Agent Development with Disciple-RKF

Several of the students that took the COG course in the Winter 2001 session, together with additional students, took the "Military Applications of Artificial Intelligence" course in the Spring 2001 session. In this course the students were given a general overview of Artificial Intelligence, as well as an introduction to Disciple-RKF. These students used the agent not as end-users, but as subject matter experts charged with developing their own COG agents using Disciple. The students were organized in five two-person teams. Each team was given the project to train a personal Disciple agent according to its own reasoning in COG identification for its historical scenario. All five teams succeeded in developing working agents, with each team addressing one of the following scenarios: 1) the capture of the Leyte Island by the US forces in 1944; 2) the Inchon landing during the Korean War in 1950; 3) the Falklands war between Argentina and Britain in 1982; 4) the stabilization mission in the Grenada Island in 1983; and 5) the US invasion of Panama in December 1989. In the following we will present how the Disciple-RKF shell is used to develop such an agent.

Generally, a knowledge-based agent includes two main components, a knowledge base and a problem solving engine. The knowledge base contains the data structures representing the entities from the expert's application domain such as objects, relations between objects, classes of objects, laws, actions, processes and procedures. The problem solving engine consists of the programs that manipulate the data structures in the knowledge base in order to solve problems in a way that is similar to how the expert solves them.

Disciple-RKF is a tool for building knowledge-based agents. It consists of a general problem solving engine, an empty knowledge base, and a general learning engine. A subject matter expert can train Disciple-RKF to solve problems in a way

that resembles how the expert would teach a student or an apprentice. For instance, the expert defines a specific problem, helps the agent to understand each reasoning step toward the solution, and supervises and corrects the agent's behavior, when it attempts to solve new problems. During such mixed-initiative interactions the agent learns from the expert by employing complementary learning methods, building and refining its knowledge base to represent the problem solving expertise of the human expert. This knowledge base has 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.

In general, the process of developing a knowledge-based agent with Disciple-RKF consists of two major stages: 1) the development of the object ontology by a knowledge engineer and a subject matter expert, and 2) the training of Disciple by the subject matter expert.

In the first development stage a knowledge engineer works with a subject matter expert to specify the type of problems to be solved by the Disciple agent, to clarify how these problems could be solved by Disciple, and to develop an object ontology. A fragment of the object ontology developed for the COG domain is shown in Figure 9.4. The object ontology consists of hierarchical descriptions of the types of objects (called concepts) from the domain, specifying their properties and relationships. The objects are represented as frames, according to the knowledge model of the Open Knowledge Base Connectivity (OKBC) protocol [14]. Some of the top level objects from this hierarchy are scenario, agent,

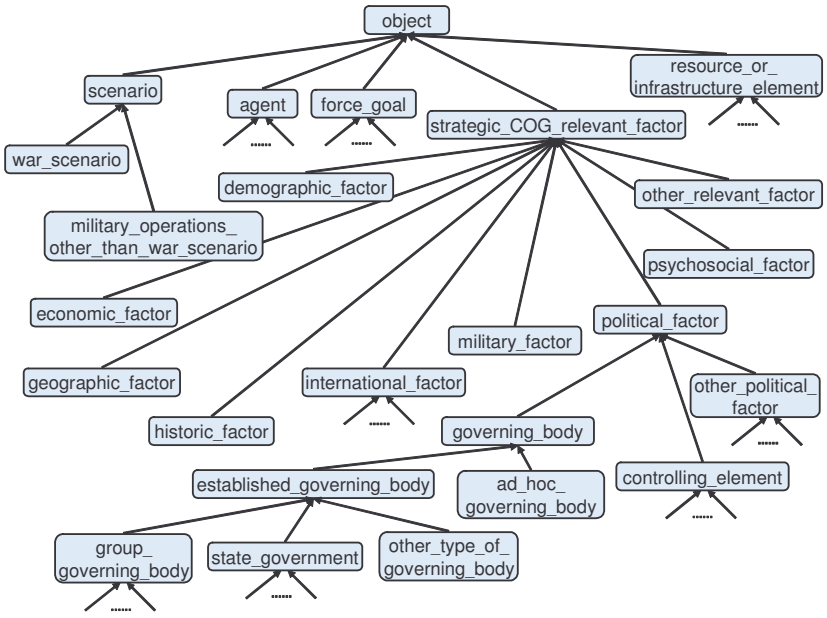


Fig. 9.4. A fragment of the COG object ontology

force_goal, and strategic_COG_relevant_factor. Each of them is the top node of a different hierarchy. Notice, for instance, that immediately under strategic_COG_relevant_factor are various such factors, including political_factor. Under political_factor are various types of political factors, one of them being governing_body. Under governing_body are the various types of governments, and so on.

Disciple-RKF includes several types of ontology browsers and editors that facilitate the ontology development process. The careful design and development of the object ontology is of utmost importance because it is used by Disciple as its generalization hierarchy for learning, as will be illustrated latter.

The result of the first development stage is a customized Disciple agent. This agent is trained to solve problems by a subject matter expert, with very limited assistance from a knowledge engineer, in the second major stage of agent development. Figure 9.5 shows the main phases of the agent training process, which starts with a knowledge base that contains only a general object ontology (but no instances, no problem solving tasks, and no task reduction rules), and ends with a knowledge base that incorporates expert problem solving knowledge. In the following we will illustrate these agent training phases.

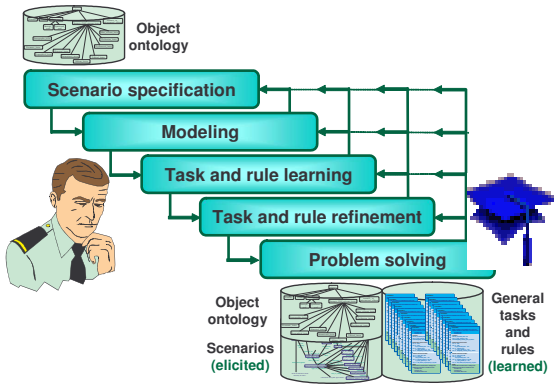


Fig. 9.5. The main phases of the agent training process

During the ***Scenario specification*** phase, the Scenario Specification module of Disciple guides the expert in describing the objects that define a specific strategic scenario (e.g. the US invasion of the island of Okinawa in 1945). The expert does not work directly with the object ontology in order to specify the scenario. Instead, the expert-agent interaction takes place as presented in section 9.2.2 and illustrated in Figure 9.1. The left hand side of the screen in Figure 9.1 contains the elements that need to be described by the expert. Notice that for each of the strategic_COG_relevant_factor in Figure 9.4 (such as political_factor or military_factor) there is a corresponding element in the left hand side of Figure 9.1 (i.e. Political factor, Military factor). These factors appear both under Japan_1945 and under US_1945. When the expert clicks on one of them in the interface shown

in Figure 9.1, Disciple asks various questions about that factor. For instance, when the expert clicks on the "Political factors" under "US_1945", Disciple asks the expert to indicate what is the type of the government of US_1945. When the expert answers that it is a representative democracy, Disciple asks additional questions about this type of government, such as, who is the head of the government, which is the name of the legislative body, and so on. Based on the answers provided by the expert, Disciple creates formal representations of specific objects that define the Okinawa_1945 scenario. These specific objects, such as government_of_US_1945, President_Truman, and Congress_of_US, are included by Disciple into the object ontology, as instances of their corresponding concepts. A fragment of the extended ontology is shown in Figure 9.6. For example, government_of_US_1945 is represented as an instance (or a member) of the concept "representative_democracy". This means that government_of_US_1945 is a representative democracy. Notice also that "representative_democracy" is a subconcept of "democratic_government." This means that any representative democracy is a democratic government. Therefore, because the government_of_US_1945 is a representative democracy, Disciple can conclude that the government_of_US_1945 is a democratic government. Based on the answers provide by the expert, Disciple defines also the features of these instances. For example, as one can see in Figure 9.6, the government_of_US_1945 has as head of government President Truman, and as legislative body Congress_of_US. Experimental results show that the experts can easily interact with the Scenario Specification module to answer Disciple's questions.

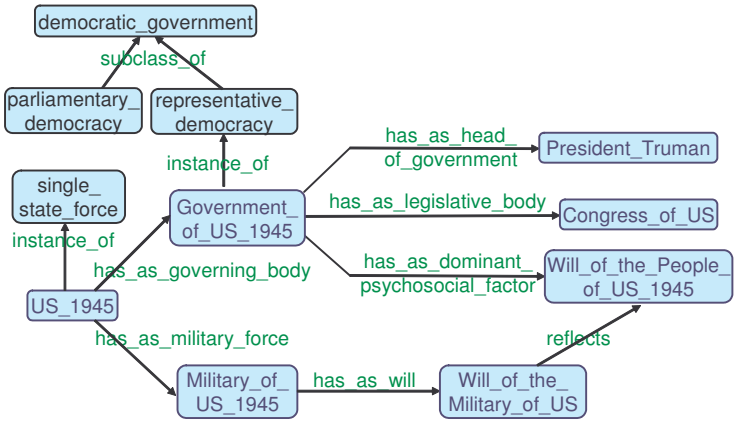


Fig.9.6. Fragment of the object ontology extended with descriptions of objects

After the expert has specified the Okinawa_1945 scenario, he can start to teach Disciple how to identify and test the strategic COG candidates for this particular scenario. The expert needs to show Disciple how he solves this problem by using the task reduction paradigm. The use of the task reduction paradigm is necessary

because this the general problem solving strategy used by the problem solving engine of Disciple-RKF. According to this paradigm, which is illustrated in Figure 9.7, a complex problem solving task is performed by successively reducing it to simpler tasks, finding the solutions of the simplest tasks, and successively composing these solutions until the solution of the initial task is obtained. In the illustration from Figure 9.7, the initial problem solving task T_0 is first reduced to the simpler task T_1. Then T_1 is reduced to the simpler tasks T_{11}, … , T_{1n}. These tasks are simple enough to find their solutions S_{11}, … , S_{1n}. These solutions are first composed into the solution S_1 of the task T_1. Then the solution S_0 of the initial task T_0 is obtained from S_1.

Task reduction is used in many domains under different names, such as "divide and conquer," or "problem decomposition." In the Disciple approach we have refined this general problem solving strategy by introducing questions and answers that guide the task reduction process, as illustrated in the right hand side of Figure 9.7. T_0 is the initial problem solving task to be performed. Finding a solution of this task is an iterative process where, at each step, we consider some relevant information that leads us to reduce the current task to a simpler task or to several simpler tasks. The question Q associated with the current task identifies the type of information to be considered. The answer A identifies that piece of information and leads us to the reduction of the current task.

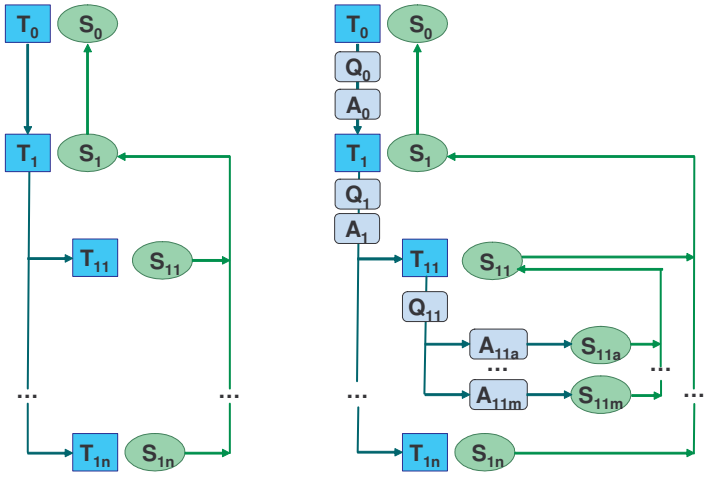


Fig. 9.7. The task reduction paradigm of problem solving.

The expert uses the **Modeling** module of Disciple-RKF to make explicit the way he identifies and tests COG candidates, within the task reduction paradigm. He expresses his reasoning in English, as a sequence of task reduction steps like the ones illustrated in Figure 9.8. First he formulates the top level problem solving task: "Identify and test a strategic COG candidate for the Okinawa_1945 scenario." To perform this task, the expert asks himself a series of questions.

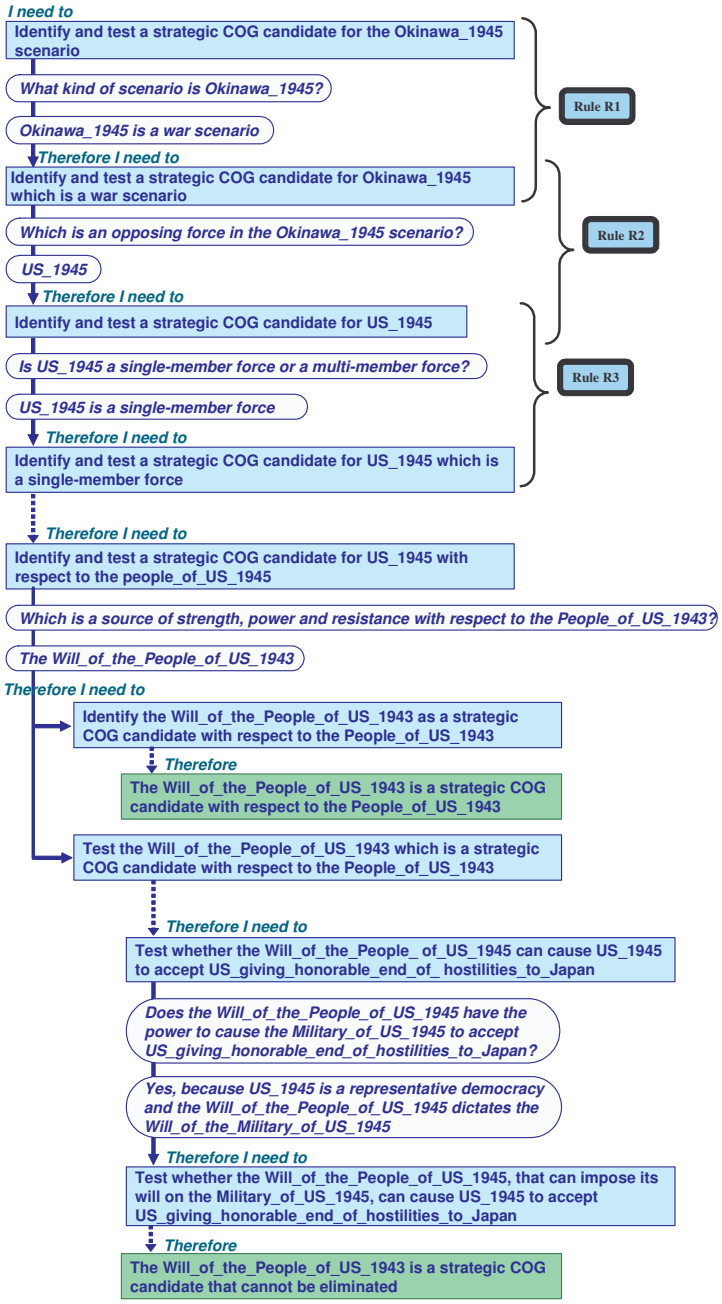


Fig. 9.8. Sample modeling of the expert's problem solving process

The answer of each question allows the expert to reduce the current task to a simpler one. This process continues until the expert has enough information to first identify a strategic COG candidate, and then to determine whether it should be eliminated or not. Let us follow the task reduction steps from Figure 9.8. Through a series of questions and answers the top level task is successively reduced to the task "Identify and test a strategic COG candidate for US_1945 with respect to the People_of_US_1945" This task is then reduced to two simpler tasks:

- "Identify the Will_of_the_People_of_US_1943 as a strategic COG candidate with respect to the People_of_US_1943"
- "Test the Will_of_the_People_of_US_1943 which is a strategic COG candidate with respect to the People_of_US_1943"

Then, each of these tasks is reduced to find their respective solutions:

- "The Will_of_the_People_of_US_1943 is a strategic COG candidate with respect to the People_of_US_1943"
- "The Will_of_the_People_of_US_1943 is a strategic COG candidate that cannot be eliminated"

Notice that this is one of the solutions of the initial task: "Identify and test a strategic COG candidate for the Okinawa_1945 scenario."

The right hand side of Figure 9.8 shows that Disciple learns a general task reduction rule from each specific problem solving step indicated by the expert. This learning takes place in the ***Task and rule learning*** phase. The learned rules allow the agent to identify and test COG candidates for new scenarios.

Let us consider, for instance, the task reduction step from the bottom part of Figure 9.8, which is also shown in the left hand side of Figure 9.9. The top task is the current task that needs to be reduced. The expert has to define a question that is relevant to the reduction of this task, then answer the question, and then reduce the top task to a simpler one that incorporates the information from the answer. Because all these expressions are in natural language and are not understood by the agent, the expert and the agent have to collaborate to translate them into the formal logical expressions on the right hand side of Figure 9.9.

First the natural language expression of each task is structured into an abstract phrase, called the task name, and several specific phrases, called the task's features. The task name should not contain any instance (such as "US_1945"), but only general concepts (such as "state"). The task's features are specific phrases (such as "The state is US_1945"). Together, the task name and the task features should have an equivalent meaning with the natural language expression of the task. The formalizations of the two tasks from the left had side of Figure 9.9 are shown in the right hand side of Figure 9.9. The task formalizations are proposed by the agent and may be modified by the expert.

Next the expert and the agent collaborate to also formalize the question and the answer from the left hand side of Figure 9.9 into the explanation from the right hand side of Figure 9.9. This explanation represents the best approximation of the meaning of the question-answer pair that can be formed with the elements of the object ontology. The explanation consists of various relations between certain elements from the agent's ontology (some of which are shown in Figure 9.6):

US_1945 has_as_governing_body Government_of_US_1945

Government_of_US_1945 is representative_democracy

US_1945 has_as_military_force Military_of_US_1945

Military_of_US_1945 has_as_will Will_of_the_Military_of_US_1945

Will_of_the_Military_of_US_1945 reflects Will_of_the_People_of_US_1945

These explanation pieces state, in Disciple's language, that US_1943 has a government which is a representative democracy and a military the will of which reflects the will of the people.

An expert can understand these formal expressions because they actually correspond to his question-answer pair. However, he cannot be expected to be able to define them because he is not a knowledge engineer. For one thing, he would need to use the formal language of the agent. But this would not be enough. He would also need to know the names of the potentially many thousands of concepts and features from the agent's ontology (such as "has_as_governing_body").

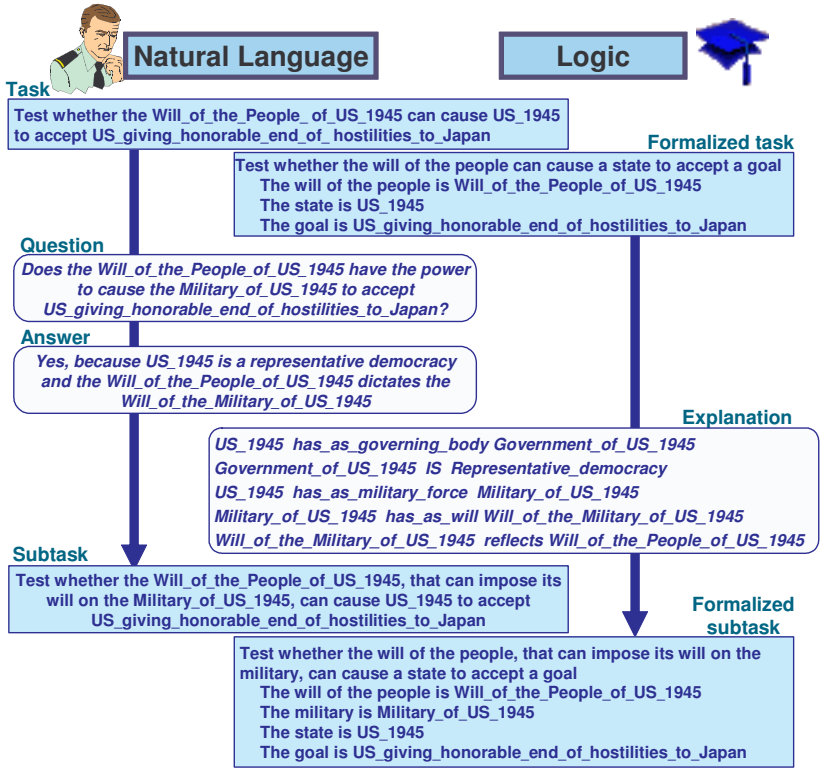


Fig. 9.9. Mixed-initiative language to logic translation

While defining the formal explanation of this task reduction step is beyond the individual capabilities of the expert and the agent, it is not beyond their joint capabilities. Finding these explanation pieces is a mixed-initiative process involving the expert and the agent. In essence, the agent will use analogical reasoning and help from the expert to identify and propose a set of plausible explanation pieces from which the expert will have to select the correct ones [15]. Based on the formalizations from Figure 9.9 and the object ontology illustrated in Figures 9.4 and 9.6, the Disciple agent learns the general IF-THEN task reduction rule shown in Figure 9.10. This rule has an informal structure, shown in the upper part of Figure 9.10, and a formal structure, shown in the lower part of Figure 9.10. The informal structure of the rule is obtained by simply replacing the five instances from the expert's example from the left hand side of Figure 9.9, with five different variables. This form of the rule preserves the natural language of the expert and is used in agent-user communication.

The formal structure of the rule is obtained by using a knowledge-based generalization of the formalized tasks and explanation from the right hand side of Figure 9.9 [1, 5]. This is the form that is used in the internal formal reasoning of the agent. This is an IF-THEN structure that indicates the condition under which the task from the IF part can be reduced to the task from the THEN part. However, instead of a single applicability condition, the rule learned by Disciple contains two plausible conditions that bound the exact condition to be learned during the ***Task and rule refinement*** phase. Initially, when the agent has no rules and no tasks, the expert teaches Disciple how to solve problems and Disciple generates partially learned tasks and rules, as indicated above. As Disciple learns from the expert, the interaction between the expert 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 expert, but also from its own successful or unsuccessful problem solving attempts. Indeed, the plausible upper bound condition of the rule in Figure 9.10 allows the rule to be applicable in many analogous situations, but the result may not be correct. The agent will apply this rule to solve new problems and the feedback received from the expert will be used to further refine the rule. Usually, the plausible upper bound condition is specialized to no longer apply in situations that produce incorrect solutions. Similarly, the plausible lower bound condition is generalized to apply in situations that produce correct solutions. In this way the two conditions will converge toward one another, both approaching the exact applicability condition of the rule. Rule refinement could lead to a complex task reduction rule, with additional Except-When conditions which should not be satisfied in order for the rule to be applicable [5].

It is important to stress that the expert does not deal directly with the learned rules, but only with their examples used in problem solving. Therefore, the complex knowledge engineering operations of defining and debugging problem solving rules are replaced in the Disciple approach with the much simpler operations of defining and critiquing specific examples.

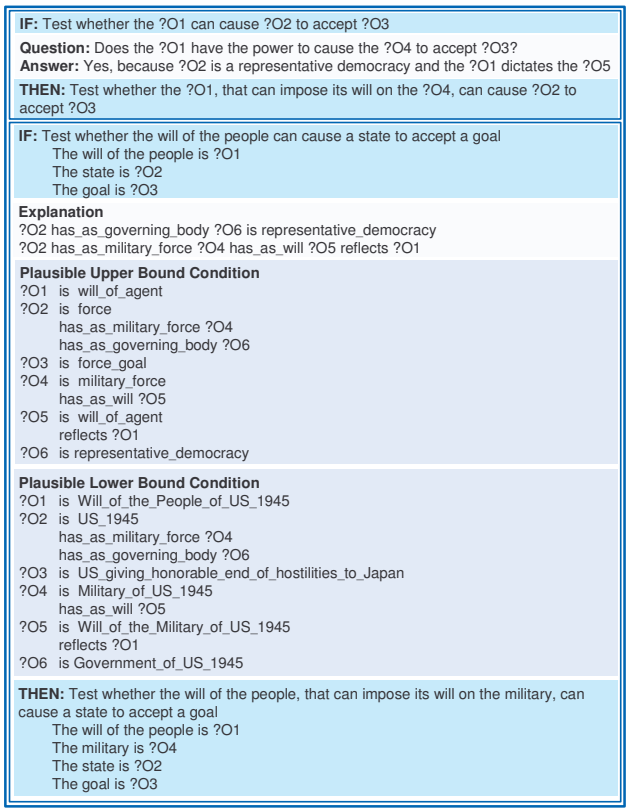


Fig. 9.10. Rule learned from the example in Fig. 9.9

After the Disciple agent has been trained, it can be used in the autonomous problem solving mode, to identify and test the strategic COG candidates for a new scenario, as was illustrated in the previous section.

In the last two 3-hour sessions of the Spring-2001 Military Applications of Artificial Intelligence course, the students have participated in a controlled agent development experiment that was videotaped in its entirety. Each of the five 2-student teams was provided with a copy of Disciple-RKF/COG that contained a generic object ontology, but no specific instances and no rules. They received a 7-page report describing the Okinawa scenario (that they have not seen before), and were asked to train their Disciple agent to identify center of gravity candidates, based on that scenario. After each significant phase of agent training and knowledge base development (i.e. scenario specification, modeling, rule learning, and rule refinement) a knowledge engineer reviewed their work, and the team then made any necessary corrections under the supervision of the knowledge engineer. Each team used the Scenario Specification tool to populate Disciple's ontology

with different instances and features. On average they defined 85.40 instances and 93.80 feature values in 1 hour and 6 minutes. After that, each team taught its Disciple agent to identify COG candidates in the Okinawa scenario. The average number of rules per team was 18.80, and the average time interval was 4 hours and 7 minutes. Although obviously incomplete (both because of the use of a single training scenario, and because of incomplete training for that scenario), the knowledge bases were good enough for identifying correct COG candidates not only for the Okinawa (evaluation) scenario, but also for new scenarios whose inputs were taken from the class projects.

At the end of this final experiment, the students completed a detailed questionnaire, containing questions about the main components of Disciple. One of the most significant results was that 7 out of the 10 experts agreed, 1 expert strongly agreed and 2 experts were neutral with respect to the statement: "I think that a subject matter expert can use Disciple to build an agent, with limited assistance from a knowledge engineer." We consider this experiment to be a very significant success. Indeed, to our knowledge, this is the first time that subject matter experts have trained an agent their own problem solving expertise, with very limited assistance from a knowledge engineer.

9.3 Course of Action Critiquing

A military Course of Action (COA) is a preliminary outline of a plan for how a military unit might attempt to accomplish a mission. A COA is not a complete plan in that it leaves out many details of the operation such as exact initial locations of friendly and enemy forces.

After receiving orders to plan for a mission, a commander and his staff analyze the mission to conceive alternative COAs, evaluate them, and select the best one. Then they prepare a detailed plans to accomplish the mission based on the selected COA. The general practice is for the staff to generate several COAs for a mission, and then to make a comparison of those COAs based on many factors including the situation, the commander's guidance, the principles of war, and the tenets of army operations. The commander makes the final decision on which COA will be used to generate his or her plan based on the recommendations of the staff and his or her own experience with the same factors considered by the staff [16]. DARPA has chosen COA critiquing as one of the challenge problems for the High Performance Knowledge Bases (HPKB) program. The goal of the HPKB program was to produce the technology needed to rapidly construct large knowledge-bases 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 participating organizations were given the challenge of rapidly developing knowledge-based systems for that problem domain, and then modifying their systems quickly to solve further problems in the same domain. The aim of the exercise was to test the claim that, with the latest

Artificial Intelligence technology, large knowledge bases can be built quickly and efficiently. The COA challenge problem is described in the next section.

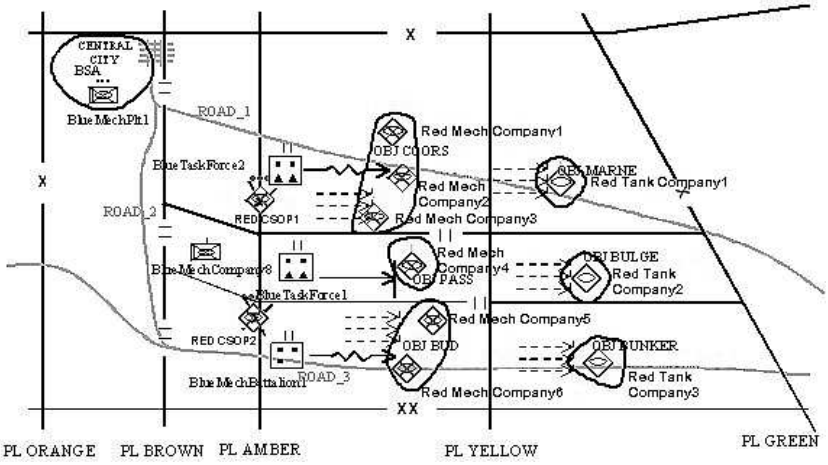
9.3.1 The Course of Action Critiquing Problem

The COA challenge problem consists of rapidly developing a knowledge-based critiquing agent that can automatically critique COAs for ground force operations, can systematically assess selected aspects of a COA, and can suggest repairs to it. The role of this agent is to act as an assistant to the military commander, helping the commander to choose between several COAs under consideration for a certain mission. The agent could also help students to learn to develop courses of action.

The input to the COA critiquing agent consists of the description of a COA that includes the following three aspects:

1. The COA sketch, such as the one in the top part of Figure 9.11, is a graphical depiction of the preliminary plan being considered. It includes enough of the high level structure and maneuver aspects of the plan to show how the actions of each unit fit together to accomplish the overall purpose, while omitting much of the execution detail that will be included in the eventual operational plan. The three primary elements included in a COA sketch are: control measures which limit and control interactions between units; unit graphics that depict known, initial locations and make up of friendly and enemy units; and mission graphics that depict actions and tasks assigned to friendly units. The COA sketch is drawn using a palette-based sketching utility.
2. The COA statement, such as the partial one shown in the bottom part of Figure 9.11, clearly explains what the units in a course of action will do to accomplish the assigned mission. This text includes a description of the mission and the desired end state, as well as standard elements that describe purposes, operations, tasks, forms of maneuver, units, and resources to be used in the COA. The COA statement is expressed in a restricted but expressive subset of English.
3. Selected products of mission analysis, such as the areas of operations of the units, avenues of approach, key terrain, unit combat power, and enemy COAs.

Based on this input, the critiquing agent has to assess various aspects of the COA, such as its viability (suitability, feasibility, acceptability and completeness), its correctness (array of forces, scheme of maneuver, command and control), and its strengths and weaknesses with respect to the principles of war and the tenets of army operations, to justify the assessments made and to propose improvements to the COA.



Mission: BLUE-BRIGADE2 attacks to penetrate RED-MECH-REGIMENT2 at 130600 Aug in order to enable the completion of seize OBJ-SLAM by BLUE-ARMOR-BRIGADE1.

Close: BLUE-TASK-FORCE1, a balanced task force (MAIN-EFFORT) attacks to penetrate RED-MECH-COMPANY4, then clears RED-TANK-COMPANY2 in order to enable the completion of seize OBJ-SLAM by BLUE-ARMOR-BRIGADE1.
BLUE-TASK-FORCE2, a balanced task force (SUPPORTING-EFFORT1) attacks to fix RED-MECH-COMPANY1 and RED-MECH-COMPANY2 and RED-MECH-COMPANY3 in order to prevent RED-MECH-COMPANY1 and RED-MECH-COMPANY2 and RED-MECH-COMPANY3 from interfering with conducts of the MAIN-EFFORT1, then clears RED-MECH-COMPANY1 and RED-MECH-COMPANY2 and RED-MECH-COMPANY3 and RED-TANK-COMPANY1. ...

Reserve: The reserve, BLUE-MECH-COMPANY8, a mechanized infantry company, follows Main Effort (MAIN-EFFORT1), and is prepared to reinforce MAIN-EFFORT1.

...

Fig. 9.11. A sample of a COA sketch and a fragment of a COA statement

The COA challenge problem was solved by developing an integrated system composed of several critiquers, each built by a different team, to solve a part of the overall problem. The teams were Teknowledge-Cycorp, the Expect team from the Information Science Institute of the University of Southern California (ISI/Expect), the Loom group from the Information Science Institute (ISI/Loom), and Disciple group from George Mason University (GMU/Disciple). All these teams shared an input ontology and used the same internal representation of the input generated by Teknowledge, Artificial Intelligence Applications Institute of the University of Edinburgh, and Northwestern University, from COA descriptions provided by Alphatech.

9.3.2 The Disciple-COA Critiquer

GMU has developed a COA critiquer, called Disciple-COA, that identifies the strengths and the weaknesses of a course of action with respect to the principles of war and the tenets of army operations [17].

There are nine principles of war: 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 army operations describe the characteristics of successful operations. They are: initiative, agility, depth, synchronization and versatility. Figure 9.12, for instance, shows some of the strengths of the COA from Figure 9.11 with respect to the Principle of Mass, identified by Disciple-COA.

In addition to generating answers in natural language, Disciple also provides the reference material based on which the answers are generated, as shown in the bottom of Figure 9.12. Also, the Disciple-COA agent can provide justifications for the generated answers at three levels of detail, from a very abstract one that shows the general line of reasoning followed, to a very detailed one that indicates each of the knowledge pieces used in generating the answer.

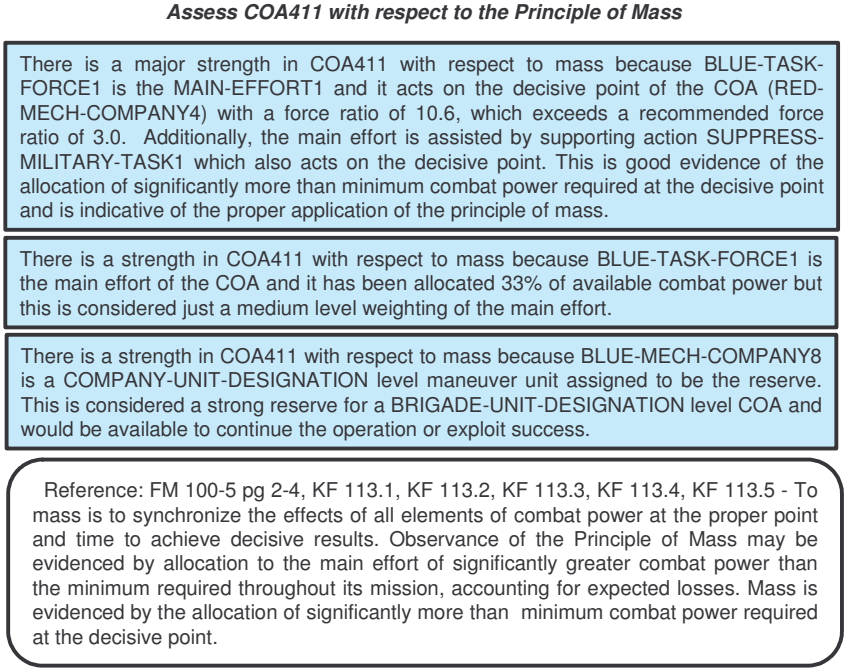


Fig. 9.12. Solutions generated by Disciple-COA

The goal of Disciple is to learn to reason in a similar way with the military expert, combining by-the-book doctrine and tactics with the lessons learned from him. The Disciple agent would not replace the commander or staff, but with no loss of capacity due to stress or other environmental factors, it would be a useful knowledge-based assistant, providing concise, relevant and explainable considerations that the commander can take into account when making decisions.

For Disciple-COA, an initial ontology was defined by importing the input ontology built by Teknowledge and Cycorp for the COA challenge problem. The input ontology contains the terms needed to represent the COAs to be critiqued, and was shared by all the developed critiquers. The top level of this ontology includes concepts for representing geographical information, military organizations and equipment, descriptions of specific COAs, military tasks, operations and purposes. The ontology was further developed by using the ontology tools of Disciple.

After the COA ontology has been developed, Disciple-COA was taught to critique COAs with respect to the principles of war and the tenets of army operations. First, the expert formulates the critiquing task to be performed, for instance, *Assess COA411 with respect to the Principle of Mass.* To perform this assessment one needs a certain amount of information that is obtained by asking a series of questions. The answer to each question allows one to reduce the current assessment task to a more specific one, as illustrated in Figure 9.13. From each task reduction step in Figure 9.13, Disciple has learned a task reduction rule, in a way that is similar to the process explained in section 9.2.3.

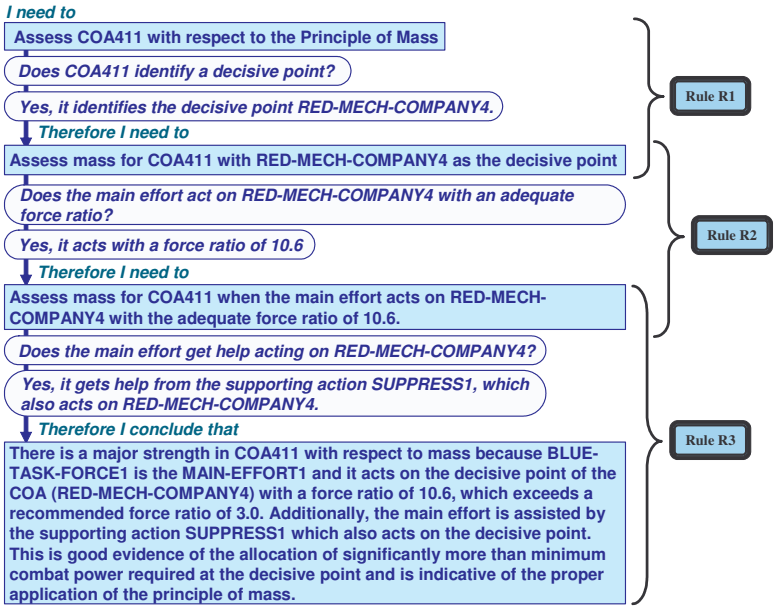


Fig. 9.13. COA critiquing through task reduction

While the critiquer developed by GMU was based on the Disciple approach, the other three critiquers were based on different approaches. Teknowledge and CYC have developed a critiquer based on the CYC system [18]. The ISI/Expect group developed a critiquer based on the Expect system [19], and ISI/Loom group developed a critiquer based on the Loom and PowerLoom systems [20]. All the critiquers were evaluated as part of the HPKB's annual evaluation that took place during the period July 6-16, 1999, and included five evaluation items of increasing difficulty. Each item consisted of descriptions of various COAs and a set of questions to be answered about each of them. Item1 consisted of COAs and questions that were previously provided by DARPA to guide the development of the COA critiquing agents. Item2 included new test questions about the same COAs. Items 3, 4, and 5 consisted of new COAs that were increasingly more complex and required further development of the COA agents in order to properly answer the asked questions. Each of the Items 3, 4 and 5 consisted of two phases. In the first phase each team had to provide initial system responses. Then the evaluator issued the model answers and each team had a limited amount of time to repair its system, to perform further knowledge acquisition, and to generate revised system responses.

Figure 9.14 compares the recall and the coverage of the developed critiquers for the last three most complex items of the evaluation. For each item, the beginning of each arrow shows the coverage and recall for the initial testing phase, and the end of the arrow shows the same data for the modification phase. The best results are those from the upper-right corner of the graph. This graph shows that all the systems increased their coverage during the evaluation. In particular, the KB of Disciple increased by 46% (from the equivalent of 6229 simple axioms to 9092 simple axioms), which represents a very high rate of knowledge acquisition of 286 simple axioms/day. The final knowledge base contained 801 concepts, 444 object and task features, 360 tasks and 342 task reduction rules. Also, each COA was represented with around 1,500 facts.

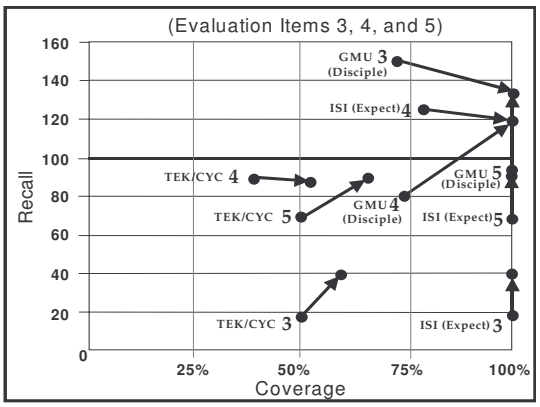


Fig. 9.14. Coverage vs. recall pre- and post-repair

9.4 Workaround Reasoning

Workaround reasoning is another domain used in the DARPA's High Performance Knowledge Bases Program to focus the research and development efforts and measure the effectiveness of alternative technical approaches to knowledge base and agent development. Workaround reasoning consists of estimating enemy's best way of working around damage to an infrastructure, such as a damaged bridge or a cratered road. Workaround reasoning supports air campaign planning in two ways. First it facilitates the evaluation of targeting strategies aimed at disabling enemy infrastructure systems. Second, it provides information about enemy assets that should be preemptively targeted to impede its efforts to work around battle damage. The specific workaround challenge problem addressed is presented in the next section.

9.4.1 The Workaround Reasoning Challenge Problem

The Workaround Reasoning problem consists of rapidly developing a special knowledge-based planning agent that can estimate which is the best way for an enemy unit to workaround some damage in its path [21].

The input to the agent includes two elements:

1. a description of the damage (e.g. a bridge is destroyed – see Figure 9.15), and of the terrain (e.g. the soil type, the slopes of the river's banks, the river's speed, depth and width);
2. a detailed description of the resources in the area that could be used to repair the damage (e.g. a description of the engineering assets of the military unit that has to workaround the damage, as well as the descriptions of other military units in the area that could provide additional resources).

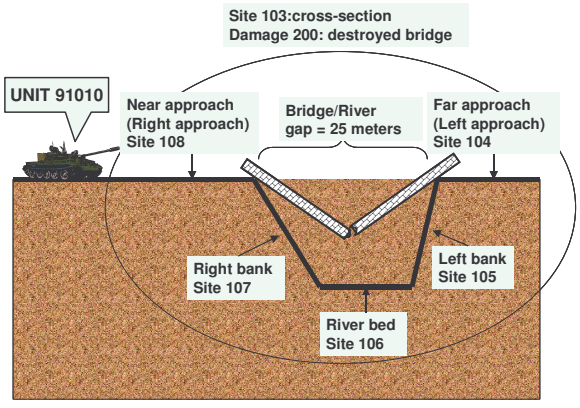


Fig. 9.15. An example of a workaround problem

The output of the agent consists of the most likely repair strategies, each described in terms of three elements:

1. a reconstitution schedule, giving the transportation capacity of the damaged link (bridge, road or tunnel), as a function of time, including both a minimum time and an expected time;
2. a partially ordered plan of engineering actions to perform the repair, and the minimum as well as the expected time that each of these actions require;
3. a set of required resources for the entire plan and for each action.

To solve this problem we have developed the Disciple-Workaround learning agent [4], as discussed in the next section.

9.4.2 The Disciple-Workaround Planner

The Disciple-Workaround planning agent was developed very rapidly. First, an initial object ontology was built by representing knowledge from Military Engineering manuals. After that, Disciple was taught based on sample solutions provided by the Alphatech's domain expert, by using an approach similar to that presented in the previous sections. For instance, the teacher defines a specific workaround task (such as, "Workaround obstacle by unit10" in Figure 9.16). Then he asks a question related to this task, the answer of which leads to the reduction of this task to a simpler one (or, in other cases, to several simpler tasks). This process continues until the top level task is reduced to a partially ordered sequence of elementary tasks that represent the plan to workaround the obstacle. From each task reduction step Disciple learns a general task reduction rule, as discussed in the previous sections.

Disciple-workaround was evaluated together with three other planners, one developed by the Information Science Institute, one developed by Teknowledge, and one developed the Artificial Intelligence Applications Institute. The planning agents were tested on several sets of problems. Then the model solutions of the problems were provided, and the agents needed to be updated in a limited amount of time, to improve their performance [22]. The problem illustrated in Figure 9.15 was one of the testing problems. The damage is a destroyed bridge and UNIT91010 has to work around it, to cross the river. The best solution generated by Disciple is shown in Figure 9.17, and consists of installing an AVLB bridge over the river gap. It is estimated that this will take a minimum of 11h:4m:58s, the expected duration being 14h:25m:56s. UNIT91010 will need the help of UNIT202, which has AVLB equipment, and of UNIT201, which has a bulldozer. After the bridge is installed, it will allow a traffic rate of 135.13 vehicles/h. The plan consists of 12 elementary actions. UNIT91010 has to obtain operational control of UNIT202 which has the AVLB. Then this unit has to come to the site of the destroyed bridge. Also, UNIT91010 has to obtain operational control of UNIT201 which has a bulldozer. This unit will have to move to the site of the destroyed bridge and then to narrow the river gap from 25m to 17m. These actions can take place in parallel with the actions of bringing UNIT202 to the bridge site. Then the AVLB bridge is emplaced, the bulldozer moves over the bridge and

clears the other side of the river to restore the flow of traffic. This plan was generated by successively reducing the WORKAROUND-OBSTACLE task to simpler subtasks, until this task was reduced to the 12 tasks in Figure 9.17.

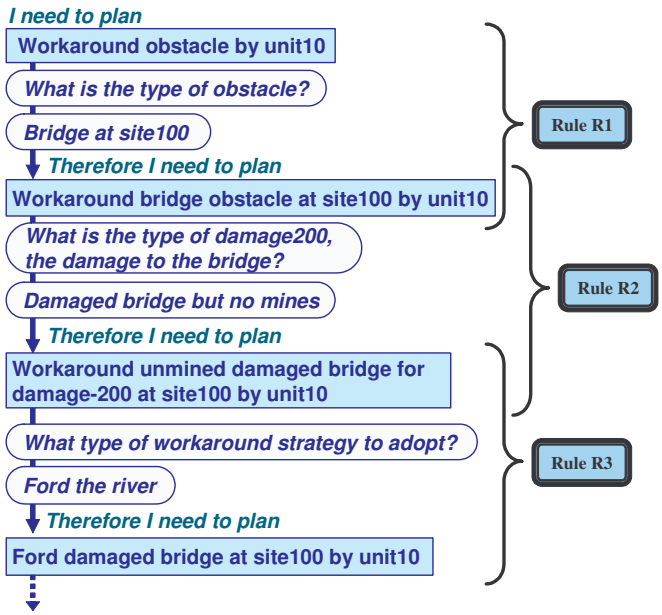
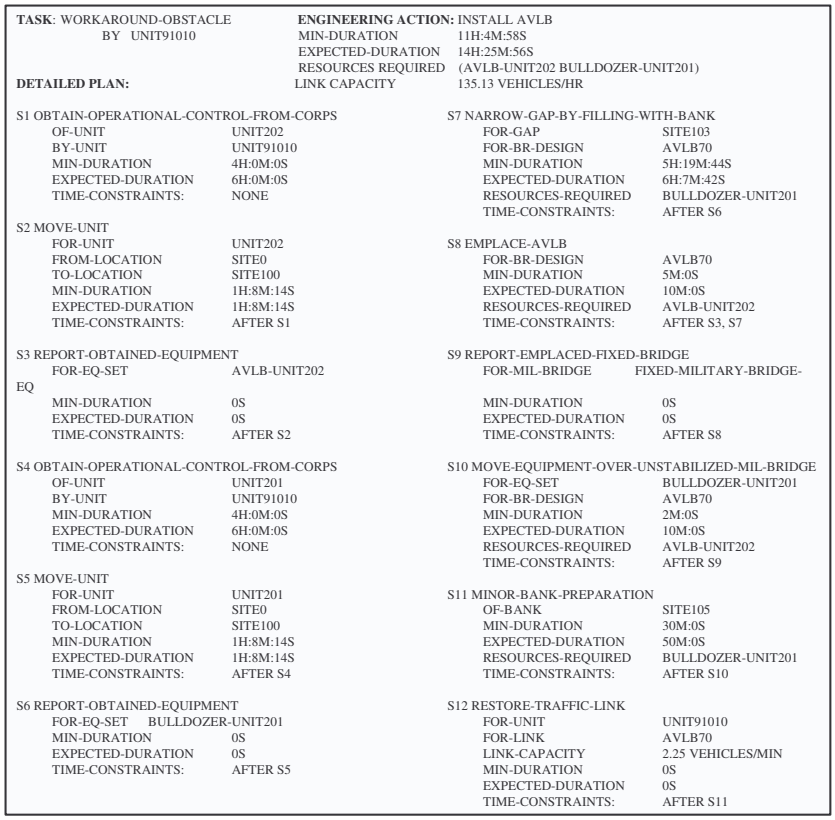


Fig. 9.16. Sample task reduction tree

During the 17 days of DARPA's evaluation, the knowledge base of Disciple was increased by 72%, from the equivalent of 5,920 simple axioms to 10,162 simple axioms. It was extended with 147 concepts, 104 tasks and 206 task reduction rules, more than any other system. The Disciple-Workaround agent has also achieved the best correctness among all the teams that participated in the workaround challenge problem, and was selected to represent the HPKB program at EFX'98, the Air Force's show case of the most promising technologies.

9.5 Long-term research vision

The Learning Agents Laboratory of George Mason University is developing the Disciple theory, methodology and tools, for building instructable knowledge-based agents. This effort directly addresses the knowledge acquisition bottleneck which we consider to be one of the major barriers in the development and maintenance of Artificial Intelligence applications.



TASK: WORKAROUND-OBSTACLE BY UNIT91010

ENGINEERING ACTION: INSTALL AVLB
MIN-DURATION 11H:4M:58S
EXPECTED-DURATION 14H:25M:56S
RESOURCES REQUIRED (AVLB-UNIT202 BULLDOZER-UNIT201)
LINK CAPACITY 135.13 VEHICLES/HR

DETAILED PLAN:

S1 OBTAIN-OPERATIONAL-CONTROL-FROM-CORPS
- OF-UNIT: UNIT202
- BY-UNIT: UNIT91010
- MIN-DURATION: 4H:0M:0S
- EXPECTED-DURATION: 6H:0M:0S
- TIME-CONSTRAINTS: NONE

S2 MOVE-UNIT
- FOR-UNIT: UNIT202
- FROM-LOCATION: SITE0
- TO-LOCATION: SITE100
- MIN-DURATION: 1H:8M:14S
- EXPECTED-DURATION: 1H:8M:14S
- TIME-CONSTRAINTS: AFTER S1

S3 REPORT-OBTAINED-EQUIPMENT
- FOR-EQ-SET: AVLB-UNIT202
- MIN-DURATION: 0S
- EXPECTED-DURATION: 0S
- TIME-CONSTRAINTS: AFTER S2

S4 OBTAIN-OPERATIONAL-CONTROL-FROM-CORPS
- OF-UNIT: UNIT201
- BY-UNIT: UNIT91010
- MIN-DURATION: 4H:0M:0S
- EXPECTED-DURATION: 6H:0M:0S
- TIME-CONSTRAINTS: NONE

S5 MOVE-UNIT
- FOR-UNIT: UNIT201
- FROM-LOCATION: SITE0
- TO-LOCATION: SITE100
- MIN-DURATION: 1H:8M:14S
- EXPECTED-DURATION: 1H:8M:14S
- TIME-CONSTRAINTS: AFTER S4

S6 REPORT-OBTAINED-EQUIPMENT
- FOR-EQ-SET: BULLDOZER-UNIT201
- MIN-DURATION: 0S
- EXPECTED-DURATION: 0S
- TIME-CONSTRAINTS: AFTER S5

S7 NARROW-GAP-BY-FILLING-WITH-BANK
- FOR-GAP: SITE103
- FOR-BR-DESIGN: AVLB70
- MIN-DURATION: 5H:19M:44S
- EXPECTED-DURATION: 6H:7M:42S
- RESOURCES-REQUIRED: BULLDOZER-UNIT201
- TIME-CONSTRAINTS: AFTER S6

S8 EMPLACE-AVLB
- FOR-BR-DESIGN: AVLB70
- MIN-DURATION: 5M:0S
- EXPECTED-DURATION: 10M:0S
- RESOURCES-REQUIRED: AVLB-UNIT202
- TIME-CONSTRAINTS: AFTER S3, S7

S9 REPORT-EMPLACED-FIXED-BRIDGE
- FOR-MIL-BRIDGE: FIXED-MILITARY-BRIDGE-EQ
- MIN-DURATION: 0S
- EXPECTED-DURATION: 0S
- TIME-CONSTRAINTS: AFTER S8

S10 MOVE-EQUIPMENT-OVER-UNSTABILIZED-MIL-BRIDGE
- FOR-EQ-SET: BULLDOZER-UNIT201
- FOR-BR-DESIGN: AVLB70
- MIN-DURATION: 2M:0S
- EXPECTED-DURATION: 10M:0S
- RESOURCES-REQUIRED: AVLB-UNIT202
- TIME-CONSTRAINTS: AFTER S9

S11 MINOR-BANK-PREPARATION
- OF-BANK: SITE105
- MIN-DURATION: 30M:0S
- EXPECTED-DURATION: 50M:0S
- RESOURCES-REQUIRED: BULLDOZER-UNIT201
- TIME-CONSTRAINTS: AFTER S10

S12 RESTORE-TRAFFIC-LINK
- FOR-UNIT: UNIT91010
- FOR-LINK: AVLB70
- LINK-CAPACITY: 2.25 VEHICLES/MIN
- MIN-DURATION: 0S
- EXPECTED-DURATION: 0S
- TIME-CONSTRAINTS: AFTER S11

Fig. 9.17. Sample workaround plan generated by Disciple-Workaround.

The military applications presented in this chapter have shown that Disciple has reached a significant level of maturity, being usable to rapidly develop complex knowledge based agents. However, these are only initial results of a long-term research the objective of which is to develop the Artificial Intelligence theory and practice that will change the way intelligent agents are built, from "being programmed" by a knowledge engineer to "being taught" by a user that does not have prior knowledge engineering experience. Achieving this objective will allow a normal computer user, who is not a trained knowledge engineer, to build by himself an intelligent assistant as easily as he now uses a word processor to write a paper. It is expected that this research will contribute to a new revolution in the use of computers, probably even more important than the creation of personal computers. Indeed, it will allow every person to no longer be only a user of programs developed by others, but also an agent developer himself.

Acknowledgements

The Disciple approach was started with the support of Mihai Draganescu, at the Research Institute for Informatics and the Romanian Academy. The research reported in this chapter was done in the GMU Learning Agents Laboratory (LALAB) with support from the Defense Advanced Research Projects Agency, Air Force Research Laboratory, Air Force Material Command, USAF, under agreement number F30602-00-2-0546, from the Air Force Office of Scientific Research grant number F49620-97-1-0188 and grant number F49620-00-1-0072, and from the US Army War College, and under the direction of David Gunning, Murray Burke, Robert Herklotz, Alex Kilpatrick, William Rzepka, Douglas Campbell and David Cammons. Several members of the LALAB have contributed to the development of the versions of Disciple presented in this chapter, especially Michael Bowman, Dorin Marcu, Kathryn Wright, Bogdan Stanescu, and Cristina Boicu (Cascaval). Jerry Comello was the primary subject matter expert for the center of gravity application of Disciple.

References

1. Tecuci, G. Building Intelligent Agents: An Apprenticeship Multistrategy Learning Theory, Methodology, Tool and Case Studies. Academic Press, London, England, 1998.
2. Gunning, D., and Burke, M., High Performance Knowledge Bases (HPKB) Program Description, http://dtsn.darpa.mil/iso/index2.asp?mode=9
3. Burke, M., Rapid Knowledge Formation Program Description. At http://www.darpa.mil/iso2/rkf/RKF_PIP.htm
4. Tecuci, G., Boicu, M., Wright, K., Lee, S.W., Marcu, D., and Bowman, M. An integrated shell and methodology for rapid development of knowledge-based agents. Proc. AAAI/IAAI, pp. 250-257, AAAI Press, Menlo Park CA, 1999.
5. Boicu M., Tecuci G., Marcu D., Bowman M., Shyr P., Ciucu F., and Levcovici C. Disciple-COA: From Agent Programming to Agent Teaching, Proceedings of the Seventeenth International Conference on Machine Learning, Stanford, CA, Morgan Kaufmann, 2000.
6. 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, vol. 53, pp. 583-610, 2000.
7. Boicu M., Tecuci G., Stanescu B., Marcu D. and Cascaval C., Automatic Knowledge Acquisition from Subject Matter Experts. Proceedings of the International Conference of Tools with Artificial Intelligence, Dallas, Texas, November 2001.
8. Tecuci, G., Boicu, M., Marcu, D., Stanescu, B., Boicu C., Comello, J., Lopez, A., Donlon, J. and Cleckner, W., Development and Deployment of a Disciple Agent for Center of Gravity Analysis, Fourteenth Annual Conference on Innovative Applications of Artificial Intelligence, IAAI-2002, Edmonton, Alberta, Canada, 2002, Deployed Application Award.

9. Department of the Army. Field Manual 3-0, Operations. Washington, DC: U.S. Government Printing Office, 2001.
10. Department of Defense. Joint Publication 3-0, Doctrine for joint operations. Washington, DC: U.S. Government Printing Office, 1993.
11. Clausewitz, C.V. 1832. On War, translated and edited by M. Howard and P. Paret, Princeton, NJ: Princeton University Press, 1976.
12. Giles, P.K., and Galvin, T.P., Center of Gravity: Determination, Analysis and Application. CSL, U.S. Army War College, PA: Carlisle Barracks, 1996.
13. Strange, J., Centers of Gravity & Critical Vulnerabilities: Building on the Clausewitzian Foundation So That We Can All Speak the Same Language, Quantico, VA: Marine Corps University Foundation, 1996.
14. Chaudhri, V. K., Farquhar, A., Fikes, R., Park, P. D., and Rice, J. P., OKBC: A Programmatic Foundation for Knowledge Base Interoperability. In Proceedings of the Fifteenth National Conference on Artificial Intelligence, 600–607, Menlo Park, California: AAAI Press, 1998.
15. 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, Vol. 22, No. 2, pp. 43-61, 2001.
16. Jones, E., 1999. HPKB Course of Action Challenge Problem Specification, Alphatech, Inc., Burlington, MA.
17. FM 100-5, US Army Field Manual 100-5, Operations, Headquarters, Department of the Army, 1993.
18. Lenat, D. B., CYC: A Large-scale Investment in Knowledge Infrastructure. Communications of the ACM Vol. 38, No. 11, pp 33-38, 1995.
19. Kim, J., and Gil, Y., Deriving Expectations to Guide Knowledge Base Creation. In Proceedings of the Sixteenth National Conference on Artificial Intelligence, 235-241, Menlo Park, California: AAAI Press, 1999.
20. MacGregor, R., Retrospective on LOOM. Available online at: http://www.isi.edu/isd/LOOM/papers/macgregor/Loom_Retrospective.html, 1999.
21. Jones, E., HPKB Year 1 End-to-End Battlespace Challenge Problem Specification, Burlington, MA, 1998.
22. Cohen P., Schrag R., Jones E., Pease A., Lin A., Starr B., Gunning D., and Burke M., The DARPA High-Performance Knowledge Bases Project, AI Magazine, Vol. 19, No. 4, pp. 25-49, 1998.