

# **A Methodology for Modeling Expert Knowledge for Development of Agent-Based Systems**

**Michael Bowman, Ph.D.**

Murray State University

Department of Computer Science and Information Systems

Murray State University

Murray, Kentucky 42071 USA

Voice: 1 270 809 6218

Fax 1 270 809 3199

[michael.bowman@murraystate.edu](mailto:michael.bowman@murraystate.edu)

# **A Methodology for Modeling Expert Knowledge for Development of Agent-Based Systems**

**Abstract:** For intelligent agents to become truly useful in real-world applications it is necessary to identify, document, and integrate into them the human knowledge used to solve real-world problems. This article describes a methodology for modeling expert problem-solving knowledge that supports ontology import and development, teaching-based agent development, and agent-based problem solving. It provides practical guidance to subject matter experts on expressing how they solve problems using the task reduction paradigm. It identifies the concepts and features to be represented in an ontology; identifies tasks to be represented in a knowledge base; guides rule learning/refinement; supports natural language generation; and is easy to use. The methodology is applicable to a wide variety of domains and has been successfully used in the military domain. This research is part of a larger effort to develop an advanced approach to expert knowledge acquisition based on apprenticeship multi-strategy learning in a mixed-initiative framework.

**Keywords:** knowledge acquisition; knowledge based systems; knowledge discovery; knowledge modeling; knowledge utilization; ontology development; domain modeling; machine learning; mixed-initiative multi-strategy learning

## **INTRODUCTION**

In order for multi-agent systems (MAS) to be truly useful in real-world applications and environments it is necessary to identify, document, and integrate into the MAS the human knowledge people use to solve their real-world problems. This process has been found to be difficult, and is a key part of the knowledge acquisition bottleneck.

This chapter presents a general methodology for collecting, modeling, and representing expert problem-solving knowledge for a given domain that supports ontology development and import, agent design, teaching-based intelligent agent development, and agent-based problem solving that is applicable to MAS. The methodology was developed as part of the George Mason University (GMU) Learning Agent Laboratory (LALAB) research funded by the Defense Advanced Research Projects Agency (DARPA) in the DARPA High Performance Knowledge Base (HPKB) and Rapid Knowledge Formation (RKF) programs (Bowman, 2002).

A methodology is a collection of methods, and methods are plans or systems of action, consisting of procedures and guidelines for accomplishing tasks. The methodology described here was designed to address the necessity to identify and document expert problem-solving knowledge and facilitate its integration into a knowledge base, to support the design and creation of agents that solve real-world problems. It is applicable to a wide variety of domains and is natural and easy to use. It provides organized, effective, and repeatable methods with detailed procedures and guidelines for accomplishing these goals:

- identification of the tasks to be represented in a knowledge base and a means of providing problem-solving examples
- identification of necessary concepts and relationships to be represented in an ontology
- guidance for rule learning and refinement
- support for natural language generation for solutions and justifications

This methodology represents extensions of existing methods of knowledge modeling based on the task reduction paradigm. These extensions include a method that can be used to model how experts solve problems, and are applicable to any domain in which task reduction is a suitable approach to problem solving. Further, a method is provided for creating an ontology specification that is a key early step in ontology development.

Other works (Mustafa, 1994; Tecuci, 1998; Schreiber, et al., 2000) have included elements that describe the importance of knowledge acquisition, explain how knowledge engineers support the process, suggest ways knowledge can be represented, and provide examples. MAS specific works (Maes, et al., 2001; Hernandez, et al., 2003) emphasize the importance of and describe methods for organizing or modeling MAS architecture. Neither of these groups of work however, has concentrated on modeling expert problem-solving knowledge or providing specific methodologies for accomplishing it. This chapter documents a methodology for modeling expert problem-solving knowledge in support of agent development that can be used directly by an expert, knowledge engineers, or system designers for the design and implementation of MAS or individual agents.

## **BACKGROUND**

A typical first step in the development of agent-based systems is modeling expert problem-solving processes and reasoning. Reasoning is the formulation of conclusions, judgments, or inferences from facts or premises. With respect to agent development, knowledge modeling is the conceptualization and representation of problem-solving knowledge in a knowledge base. It is potentially the most difficult aspect of developing knowledge-based systems. Knowledge conceptualization and representation are particularly difficult because the form in which experts express their knowledge is significantly different from how it should be represented in the knowledge base. Moreover, experts typically fail to specify the knowledge that is common sense or implicit in human communication, but which needs to be explicitly represented in the knowledge base. After knowledge is conceptualized and represented, the representation must be verified for correctness and usefulness. This modeling and transfer of knowledge between the domain expert and the knowledge base is often a long, inefficient process. The methodology described here addresses these issues by allowing subject matter experts to directly model their problem solving technique.

This methodology is based on the task reduction paradigm of problem solving. A task reduction solution diagram represents a logical set of steps that reduces a complex problem to a series of increasingly less complex problems until the problem can be solved with the facts at hand. In this methodology, modeling is the process of creating an explicit, yet initially informal

representation of processes and reasoning an expert uses to solve a problem. That is, specifying a detailed sequence of problem-solving steps in the form of a task reduction solution diagram, and expressing the problem-solving logic that justifies the reduction of each task to a subsequent sub-task or solution, in the form of a natural language question and answer.

Task reduction as an element of system design has been a key concept in artificial intelligence and agent-based systems since the earliest days of the field. Allen Newell and Herbert A. Simon's seminal book on human intelligence and research directions for artificial intelligence, *Human Problem Solving*, (1972) emphasizes the importance of decomposition of complex problems into sub-problems. In his book, *Society of Mind* (1985), Marvin Minsky emphasizes the importance of problem decomposition in the design of agents. Task reduction remains a key element of modern agent development environments like Disciple (Tecuci, 1998), Protégé (Grosso et al., 1999), CommonKADS (Schreiber et al., 2000), and MIKE (Angele et al., 1998). Task reduction is also an essential element in the development of MAS as it serves as the basis for work allocation and cooperation among the agents comprising a system.

Despite its importance, the general concept of task-reduction is not sufficient for agent-based system design. Newell, Minsky, and others have recognized the importance of dialog with domain experts to capture the essence of their problem solving methods. This methodology takes advantage of a design concept generally alluded to by Newell, Simon, Minsky and others and which can be called: Task Reduction and Dialog as Metaphors for Knowledge-Based System Design. In this methodology, the domain experts conducts a dialog through interaction with a computer or knowledge engineer, and by breaking down problem solving methods into a series of tasks to be accomplished, questions to be addressed in the task, and the answers to those questions, the requisite problems solving steps and logic, including ontological elements and state information can be captured as a necessary first step in system design. The chapter includes both the "how to" steps and procedures of the methodology, and a structured, mathematical representation of the methodology.

## **A New Approach to Intelligent Agent Development - Direct Knowledge Modeling**

The Learning Agents Laboratory (LALAB) at George Mason University (GMU) has developed an apprenticeship, multi-strategy, learning approach for building intelligent agents called Disciple. A primary goal of Disciple is the creation of an environment in which subject matter experts (SME) develop and maintain their own problem-solving intelligent agents without the direct intervention of knowledge engineers (KE). In the Disciple approach a SME teaches a learning agent how to perform domain-specific tasks in a manner that resembles the way the SME would teach a human apprentice, by giving the agent examples and explanations as well as by supervising and correcting the agent's behavior (Tecuci, 1998). The LALAB has developed a series of increasingly more capable learning agents from the Disciple family which address very complex problems in the military domain.

While the principles, procedures, and formalization of the modeling methodology described here are very generally applicable to MAS development, it has been developed, tested, and externally

evaluated over many years as an integrated element of the Disciple approach to agent development.

### **Task Reduction - A General Problem-Solving Paradigm**

A fundamental and powerful concept for problem solving is that a complex problem can be successively reduced to simpler sub-problems until the sub-problems are simple enough to be solved immediately. The solutions to the sub-problems can then be successively combined to produce the solution to the initial problem.

A more formal representation of the task reduction concept is:

*A problem P can be reduced to the simpler problems P1, ... Pn. The solutions S1, ... Sn of the problems P1, ... Pn can be combined into a solution S of P.*

A simple example of this process from mathematics is:

P: Solve for x:  $|x + 5| > 2$   
Reduces to:      P1: Solve for x:       $x + 5 < -2$       S1:  $x < -7$   
                    And              P2: Solve for x:       $x + 5 > 2$       S2:  $x > -3$   
S:  $S1 \cup S2 = (x < -7) \text{ or } (x > -3)$

This general concept has been given many names including problem or task decomposition, factorization, divide and conquer, and task reduction. In many works on problem solving, decision making, systems engineering and various aspects of artificial intelligence research, the process is described but not named. The term task reduction will be used for this concept throughout the remainder of this chapter except where directly quoting other related works.

In an essay on human problem solving in *Society of Mind*, Minsky described human intelligence as a combination of simpler things, or agency through task reduction (Minsky, 1985). As an example of this concept, Minsky describes an agent for stacking a child's blocks to make a tower, called "builder," presenting the task and sub-tasks of builder with a simple task hierarchy diagram, similar to Figure 1. Task hierarchy diagrams are useful for most representations of problem solving with task reduction. This methodology uses the same principle to depict solutions graphically, but calls the diagrams 'solution trees.'

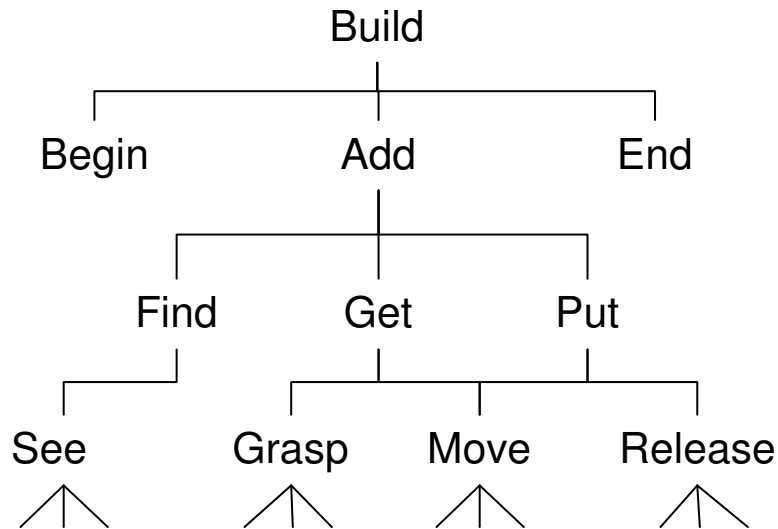


Figure 1 - Marvin Minsky's Builder Agent (Minsky, 1985, pp.21)

## Task Reduction in Multi-agent Systems (MAS)

Minsky described an agent as a process or entity that could accomplish some simple task (Minsky, 1985). Tecuci provides a general characterization of an intelligent agent as:

*“... a knowledge-based system that perceives its environment; reasons to interpret perceptions, draw inferences, solve problems and determine actions; and acts upon that environment to realize a set of goals or tasks for which it was designed.”* (Tecuci, 1998, pp 1)

Multi-agent systems are environments in which multiple, independently operating agents function, coexist, and cooperate to achieve goals. Although Minsky described theoretical multi-agent systems in detail as early as 1985, it has only been recently that research and computing advances have made systems of this type practical. Agent coexistence and cooperation are currently key research topics in artificial intelligence.

Task reduction is an essential element in the development of multi-agent systems as it serves as the basis for work allocation and cooperation among the agents comprising the system. Four distinct phases have been identified for problem solving with multi-agent systems (Uma et al., 1993):

- problem decomposition
- sub-problem allocation
- sub-problem solution
- sub-problem integration or result synthesis

Alvares et al., (1998) identified two types of task reduction (using the term problem decomposition) necessary for multi-agent systems:

- extrinsic decomposition, where each agent accomplishes the same type of tasks in parallel to speed up problem solving
- intrinsic decomposition, where agents are specialized to solve particular sub-problems in parallel or serial manners

Basic investigation, development, and evaluation of MAS are major areas of emphasis in current artificial intelligence research. Luck et al., (1998), described fundamental theory for MAS. Brazier et al., (1998), described principles for MAS component reuse. Chavez et al., (1997), developed and described Challenger, a MAS for resource allocation. All of these works include task reduction as an element of MAS development and operation.

## **Task Reduction and Dialog as Metaphors for Knowledge-Based System Design**

As stated, a necessary first step in the development of knowledge-based systems is modeling expert problem-solving reasoning. Here, reasoning is the process of forming conclusions, judgments, or inferences from facts or premises. Task reduction is a natural and effective method for doing this. A key thesis of this methodology is that done effectively, task reduction based modeling of expert problem solving can contribute significantly to overcoming the knowledge acquisition bottleneck. This is also a general premise of the related agent development systems such as Protégé, CommonKADS, and MIKE.

The general concept of task reduction must be extended significantly to constitute a worthwhile component of a MAS development tool. A more comprehensive and repeatable methodology is needed for identifying and representing not only the problem-solving steps, but also the background knowledge, state information, and problem specific logic that explains the rationale of the problem-solving steps. Additionally, experts attempting to define, describe and justify problem-solving steps for use in a knowledge-based system require guidance on how to organize supporting dialog and for what constitutes adequate definitions, descriptions and justifications.

The importance of human-computer interaction and dialog for problem solving, and the use of dialog as a metaphor for system design is a key component of mixed-initiative methods in artificial intelligence research (Cohen, R., et al., 1998). The dialog metaphor is seen as a means of establishing context, goals and current state information in knowledge-based problem solving. What is missing from the general concepts of task reduction and dialog, are mechanisms for documenting and translating the human dialog or narrative that explains the problem-solving steps into a usable knowledge base. The methodology described here combines the identification of problem-solving steps, background knowledge, state information, and problem-solving logic, as well as a language-to-logic translation mechanism in a single, multi-step process that can be used by both SMEs and KEs.

# A GENERAL AND FORMAL PRESENTATION OF A METHODOLOGY FOR MODELING EXPERT KNOWLEDGE

Restating the concept of task reduction, a complex problem can be solved by successively reducing it to simpler problems, finding the solutions of the simplest problems, and successively composing these solutions until the solution to the initial problem is developed. Within the context of the task reduction paradigm, and the developed methodology, the phrase problem-solving task is used to convey the idea of a step or incremental procedure required to solve a given problem. To simplify the notation of this chapter, where appropriate, the term “task” will generally be used in place of problem, problem-solving task, sub-problem, or sub-task, which can be used interchangeably in most cases within the context of problem solving through task reduction. Figure 2 represents the task reduction concept.

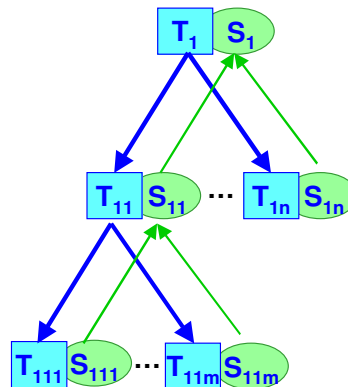


Figure 2 – Problem/Task Reduction Diagram or Tree

To make task reduction useful for knowledge-based system development it is necessary to do more than identify problem-solving steps. For the reduction from any task to any sub-task, and for each subsequent reduction between sub-tasks it is necessary to identify the elements of knowledge that are relevant to solving each task.

One such element is background knowledge in the form of ontological structures (concepts, instances and relationships) that are relevant to the current task and its reduction. Additionally, it is necessary to identify relevant, current, state information associated with the reduction between tasks and providing linkages to eventual solutions. For our purpose, state information is the collective relevant current characteristic conditions, or variable values within the system being modeled, at a particular point in time. Finally, the background knowledge and state information relevant to a given task must be combined with the appropriate problem-solving logic (reasoning) that justifies the reduction from a task to subsequent sub-tasks or solutions.

## Modeling Expert Problem Solving as Tasks, Questions and Answers

In this methodology required problem-solving steps, background knowledge, state information and logic is initially identified and represented in content rich natural language expressions consisting of:

- a task name that corresponding to a problem or sub-problem;
- a question relevant to the solution of the named task;
- one or more answers to that question;
- sub-tasks or solutions identified by the preceding task, question and answer.

The process begins with the expert identifying the problem to be solved and generating a corresponding top-level task. Most problems are likely to have many potential solutions and it is useful at this point to organize potential solutions into groups or categories. The expert may continue the modeling process by specifying a question that elicits categories of potential solutions, and as an answer, identify a category that corresponds to a particular example solution within the category to be modeled.

The modeling examples in this chapter come from the military domain and are examples of a process know as '*center of gravity (COG) analysis.*' Carl von Clausewitz introduced the COG concept as "the hub of all power and movement, on which every thing depends" (Clausewitz, 1832, pp 595). Modern military doctrine generally acknowledges that if a combatant can eliminate or influence an opponent's COG, the opponent will lose control of its power and resources and will eventually be defeated. Similarly, if a combatant fails to adequately protect its own strategic COG, it invites disaster (Giles and Galvin, 1996). This methodology has been used extensively to create COG analysis agents as part of the DARPA RKF project since 2002 (Bowman et. al., 2004).

At the top of Figure 3 is an example of a top-level task (Identify a Strategic COG candidate for the Sicily\_1943 scenario). This top-level task is followed by a question relevant to solving that task (What is one of the opposing forces?). The question is followed by an appropriate answer (Anglo\_allies\_1943 is an opposing force), which leads to a new sub-task.

The process continues in the bottom half of Figure 3 as the sub-task (Identify a Strategic COG candidate in the Sicily\_1943 scenario for Anglo\_allies\_1943) is followed by a new question (single member force or multi-member force?). This question is followed by the answer (multi-member force) which leads to a new sub-task.

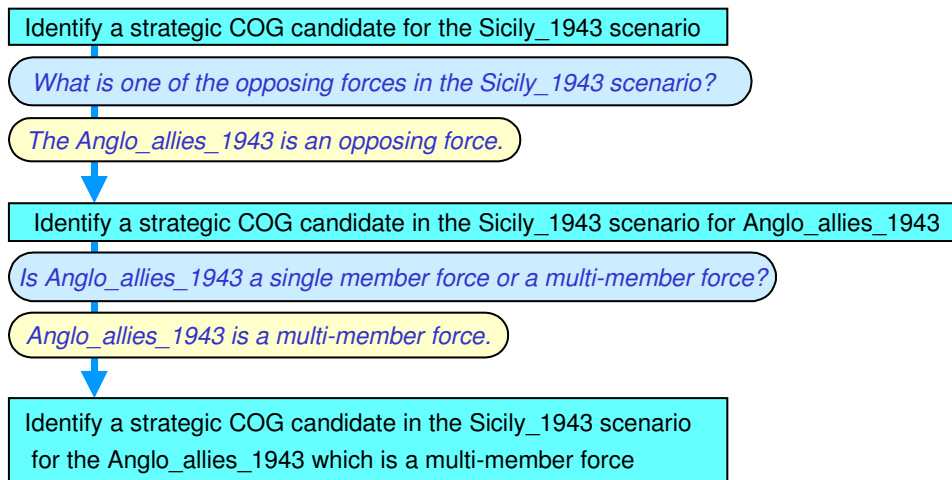


Figure 3 - Example Top Level Modeling

The question following a task can have either a single answer leading to a single sub-task or solution, or multiple answers leading to multiple sub-tasks or solutions. Attempting to simultaneously model multiple branches of a solution tree down to multiple solutions can be extremely difficult. For any question resulting in more than one answer, the modeler selects the answer that corresponds to the example solution being modeled, and continues down that branch of the solution tree, returning to the other branch(s) after the first branch is completed.

When an answer to a question appears to be complex, or includes an “and” condition, attempt to rephrase the question into a simpler more incremental question that leads to a simpler answer. Figure 4 is a subtle variation on the top half of Figure 3, where the question is worded in a way that results in an answer containing an “and” condition (The Anglo\_allies\_1943 and European\_axis\_1943 are the opposing forces). Rephrasing of the question as in Figure 3 to ask “What is one of the opposing forces ...” results in two less complex answers, (where the second answer is European\_axis\_1943 which was omitted from Figure 3 for simplicity).



Figure 4 – Mis-phrasing a Question Causing an “And” Condition

If the question can not be rephrased to result in a simpler answer, create branches as necessary, select a branch represented by a sub-task, and continue modeling down that branch. Return to the other branch(s) after the first branch is completed.

A different, but common situation exists when two or more different subtasks must be completed to accomplish a parent task. For example, in the COG domain, if the parent task was to both identify a candidate and evaluate it in some fashion “Identify and evaluate a strategic COG

candidate for ...” the two required sub-tasks might be “Identify a strategic COG candidate ...” and “Evaluate a strategic COG candidate ...”. When this occurs, the required sub-tasks are identified, and are circled, boxed, or linked in some other graphical manner to indicate an “and” condition, meaning all of the grouped tasks must be completed to accomplish the parent task. A branch represented by a sub-task is selected and modeling is continued down that branch. Return to the other branch(s) after the first branch is completed.

An effective manner of representing this situation in a question and answer is to phrase the question in a manner similar to “How does one X?” or “What actions must I complete to accomplish X?” where X is the action portion of the current task. This question would be followed by an answer with several sub-tasks such as “To do X, do A, B, and C” or “To accomplish X, I must complete A, B, and C.” For example if the current task was to emplace a bridge which involved multiple steps that should be done concurrently, the question might be “How do I emplace this bridge?” The corresponding answer might then be “To emplace this bridge, I must prepare the banks, and relocate and construct a mobile military bridge.” The steps, or actions identified in the answer then form the basis of the new sub-tasks.

The task names, questions and answers should be detailed and expressive because they serve five primary purposes:

- they identify the sequence of problem-solving steps;
- they identify the concepts, instances and relationships necessary to represent the knowledge relevant to solving the problem;
- they identify the state information that is relevant at that particular point in the problem-solving process;
- they express the reasoning justifying the reduction from one task to the next sub-task, or to a solution;
- in their final form they serve as a script that directly supports agent training.

With these purposes in mind, it is important to express answers in complete sentences, restating the key elements of the question in the answer. In Figure 3 for example, the answer to the first question is recorded as “The Anglo\_allies\_1943 is an opposing force,” rather than just “Anglo\_allies\_1943.” The expressiveness and thoroughness of the answers are very important to the language to logic translation process in MAS development.

The effects of this information gathering are cumulative. Subtasks generally contain the key information represented in preceding tasks, questions and answers. The information necessary to solve a problem is the total of the information collected in each task reduction to a subsequent sub-task or solution. This holds true regardless of the problem type (planning, identification, analysis, etc.) or nature of the eventual solution (single-first adequate solution found, collection of partial solutions, or re-composition of partial solutions). This is truer in the larger sense that the goal of the methodology is to identify and represent the problem-solving knowledge necessary to teach an agent general problem-solving methods, rather than just solving any single given problem. For this broader goal all information gathered is useful. Partial, or even incorrect solutions and the information that leads to them, contribute directly to producing more competent MAS.

To help the modeler maintain the problem-solving chain of thought while modeling complex solutions, and to portray the cumulative nature of the knowledge gathering, it is useful to express elements of preceding tasks in each sub-task. For example, the final sub-task shown in Figure 3 begins with the words of the preceding task and adds the new relevant factor from the question and answer, “which is a multi-member force.”

A final key guideline for the initial modeling phase of the developed methodology is that it should only be necessary to completely model solutions that are unique in their entirety. Entirely unique solutions are likely to be rare. Solutions can generally be organized within categories of solutions, and solutions within a category are often variations of, or branches off of other solutions within the category. After completing a first solution tree, subsequent variations or branches off of that solution will reuse portions of the original solution tree. Figure 3 provides a very simple example of this. After identifying the top-level task and question, and generating the first answer in Figure 3 (Anglo\_allies\_1943), it is obvious that this top-level task and question can be reused when modeling solutions for the other opposing force in this scenario (European\_axis\_1943).

## **Creating the Ontology Specification**

If complete and correct, the tasks, questions, and answers expressed during initial modeling should reveal the concepts, instances and relationships that are necessary to represent the knowledge relevant to solving the problem. These objects represent a specification of the object ontology that must be developed. The role of tacit knowledge in expert problem solving makes this challenging. Experimental results collected indicate however, that the cyclic nature of agent development and training compensates for this challenge, and the experience gained by experts in training an agent eventually overcomes the challenge (Bowman, 2002).

In most cases, initial ontology development with this methodology is a cyclic team effort between the SME using the methodology and a supporting KE. The expert models their problem-solving process as previously described, and then sketches semantic nets of objects, instances and relationships as described below. The basic idea is to draw the object ontology in the form of a graph in which the nodes represent objects and the arcs represent the relations between them. The supporting KE refines the sketches and then uses them as a basis for searching for useful, pre-existing object ontologies to be imported. If all of the necessary ontology elements can not be found in knowledge repositories, which is likely to be the case, the SME or supporting KE can create them with ontology development tools. Once an initial object ontology is created it is reviewed by the expert, who then updates the modeling to reflect the terminology used in the ontology. This in turn may generate additions to the ontology specification, which can cause another cycle of ontology import and extension.

Ontology specification is generally done after thoroughly modeling several example solutions in several solution categories. After completing solution trees for available examples in one or more categories, the expert selects a solution tree and begins ontology specification from the top of that tree. Examining each of the tasks, questions and answers in the tree should reveal objects that must be represented in the ontology. By hand or with any available automated drawing tool, the expert draws simple, fragmentary semantic nets representing the objects that appear in the

tasks, questions, and answers of the solution tree. More general objects are placed above their less general subclasses and instances. Directed arcs are used to represent relationships between objects, with less general objects pointing to their more general parent objects. Initial modeling may include objects whose relationship to other objects is not initially clear. These objects are included in the semantic net with no arcs to other objects. These relationships may be worked out during ontology import and extension or by further modeling. Objects that are known by the SME to have been included in previous ontology specifications, or in the ontology itself, may be omitted from the semantic net for a newly modeled solution tree.

Nouns in a task, question, or answer are likely to be objects needed in the ontology specification. Questions are likely to contain objects that are more general than related objects that appear in the corresponding answer.

Figure 5 is an example of some of the ontology structure (Force, Opposing\_force, Multi\_member\_force) and state information (Anglo\_allies\_1943 is a Multi\_member\_force and an Opposing\_force) that could be derived from the domain modeling from Figure 3.

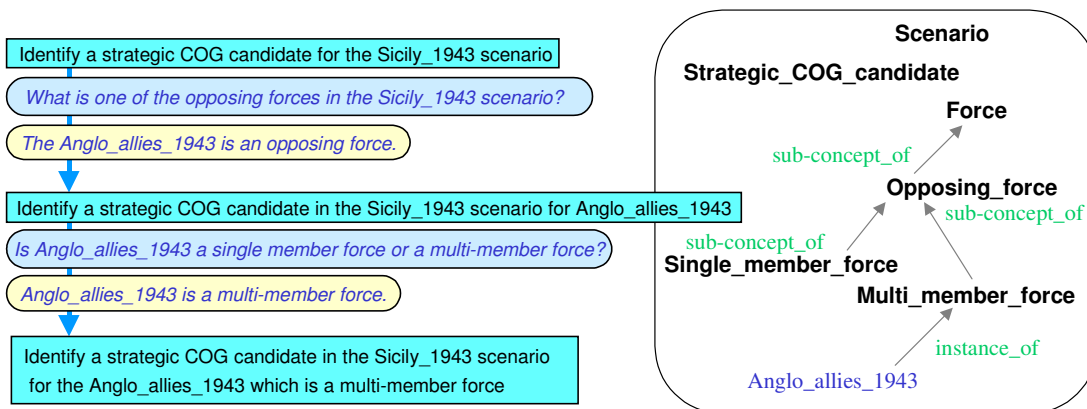


Figure 5 - Ontology Specification Derived from Initial Domain Modeling

The top-level task in Figure 5 clearly identifies at least two objects, ‘strategic COG candidate’ and ‘scenario.’ These are included in the semantic net with no arcs since their relationship to other objects is not obvious at this time. The first question in Figure 5 includes something called an ‘opposing force.’ The answer to this question indicates that ‘Anglo\_allies\_1943’ is an instance of an opposing force so these objects are added to the semantic net with an arc from *Anglo\_allies\_1943*, which is less general, to *Opposing\_force*, which is more general. The second question and answer indicate that an *opposing force* can be subcategorized as either a ‘single member force’ or ‘multi-member force,’ and that *Anglo\_allies\_1943* is an example of a *multi-member force*.

Given these new objects and relationships, *Single\_member\_force* and *Multi\_member\_force* are added to the semantic net with arcs to *Opposing\_force*, and the arc from *Anglo\_allies\_1943* is

redrawn to point to *Multi\_member\_force*. Finally, even without the object appearing anywhere in the modeling, the expert could reach the conclusion that a ‘*Force*’ may be a useful generalization of *Opposing\_force* as it corresponds to a well-established concept in the application domain. This and other objects the expert feels will be useful can be added to the semantic net with the corresponding arcs.

Modeling and ontology specification, as described above, can be complex non-linear processes. Figure 6 is a graphic representation of the first two steps of an agent development approach. Step 1, *Initial Domain Modeling*, and Step 2, *Initial Ontology Development*, are done in a nearly concurrent, cyclic manner. Domain modeling produces informal solution trees. Initial Ontology Development produces an initial ontology specification. Both steps, 1 and 2, support and feed the next step, Formalization, where the initial ontology is used to transform the informal solution trees into formalized solution trees as described next. Steps 1 and 2 are nearly concurrent in that once even a small collection of solutions trees is developed, an initial and partial semantic net of objects and relationships can be created.

The initial, partial semantic net can be used by a KE as a guide for object ontology import and extension while additional modeling is done. Steps 1 and 2 are cyclic in that modeling identifies the need for ontology development and ontology development can identify the need for additional modeling. While ontology development can begin after a single solution tree and corresponding semantic net of objects and relationships is created, it is likely to be more effective to wait until a significant body of related trees and nets are created in order to limit the rework caused by later modeling.

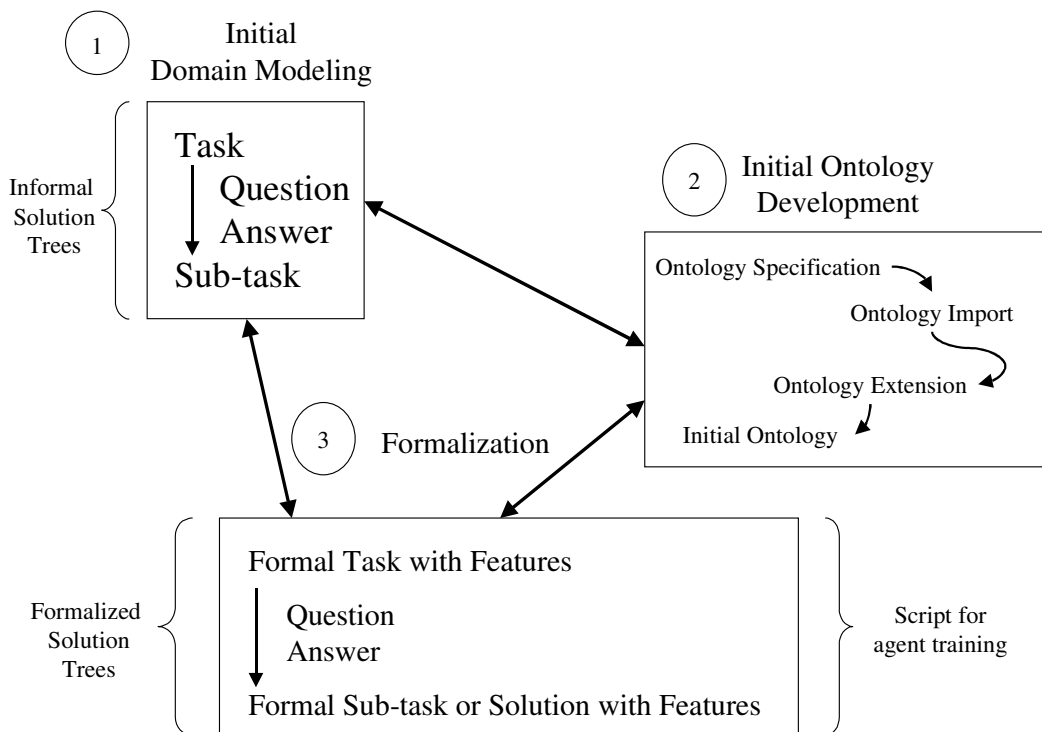


Figure 6 – A Segment of an Agent Development Process

When ontology elements are successfully identified and imported from pre-existing knowledge repositories, the names of imported objects and relationships are adopted into existing and future modeling where practical. Whether an imported name is adopted or an original name is preserved, a final decision is made as to which name to use and a correspondence between the original and imported names is recorded and maintained.

## **Formalization of Modeled Tasks and Solutions**

During Task Formalization the expert rephrases the natural language description of each task into an equivalent formal description. A formal task description consists of an abstract phrase that represents its name, and a sequence of other phrases that represent its features. A general rule for creating the task name is to replace each specific object from the natural language description of the task with a general category. In the second task of the domain modeling in Figure 3 one could replace “*the Sicily\_1943 scenario*” with “*a scenario*,” and replace “*Anglo\_allies\_1943*” with “*an opposing force*” as in Figure 7.

In this manner, the original task “*Identify a strategic COG candidate in the Sicily\_1943 scenario for Anglo\_allies\_1943*” becomes this new task with two feature phrases:

Identify a strategic COG candidate in a scenario for an opposing force

The scenario is Sicily 1943

The opposing force is Anglo allies 1943

The underlined words in the new task are general categories for the underlined words in the feature phrases, which are the objects that were removed from the original task. Feature phrases generally take the form: “The *general-category* is – *object-removed-from-the-original-task-name*.”

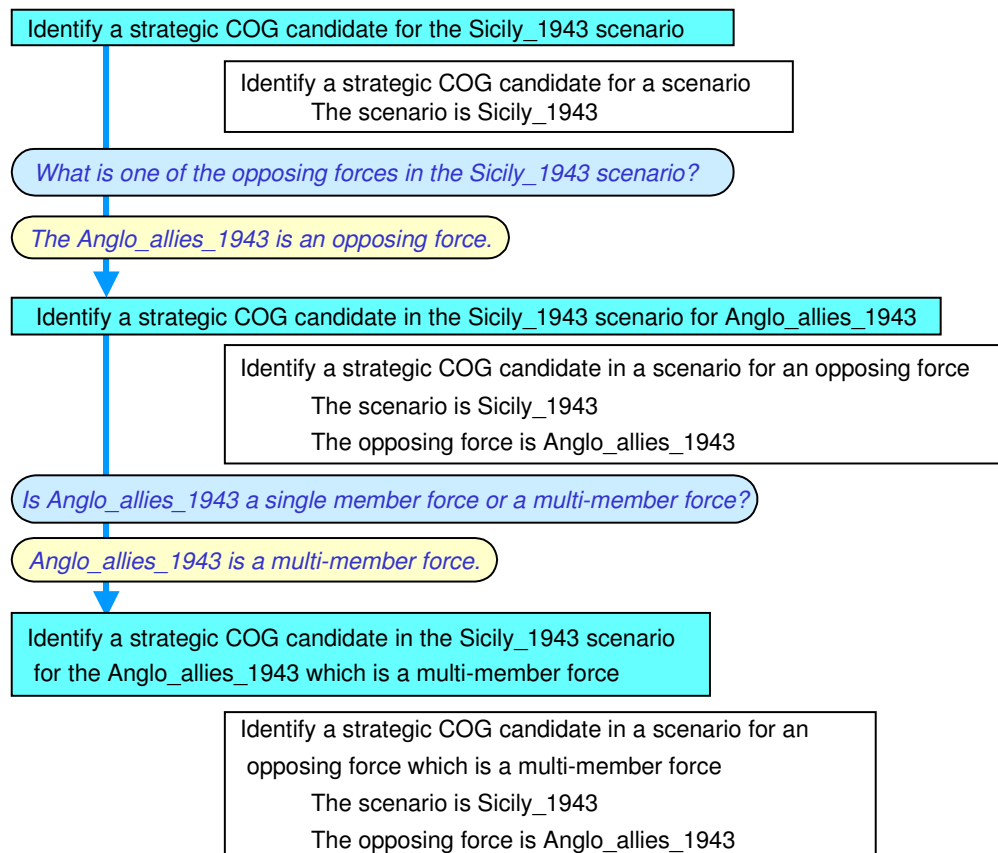


Figure 7 - Task Formalization

When correctly formalized, a task name should not contain any specific object or constant. Each task feature is a sentence that describes one of the abstract categories introduced in the task name (e.g. “a scenario”) such that the task name and the task features together are equivalent to the natural language description of the task.

Each feature sentence is independent of another. It is recommended that previously defined feature sentences should be re-used when it makes sense, rather than defining many variations that represent the same information. This limits the proliferation of task names and task features. Solutions must also be formalized with the same process.

There may be more than one way to correctly formalize a task or solution. Some formalizations are more versatile and useful than others. Figure 8 shows two different ways of correctly formalizing a solution. The first form follows the pattern specified above where each object is replaced with a general category. The second form is a more abstract representation of the task. Both forms are acceptable. However, the greater abstractness of the second form means it could be reused to formalize a larger number of solutions. It is up to the expert to determine the final form of the abstract names and features, selecting forms that he or she feels are most correct and natural.

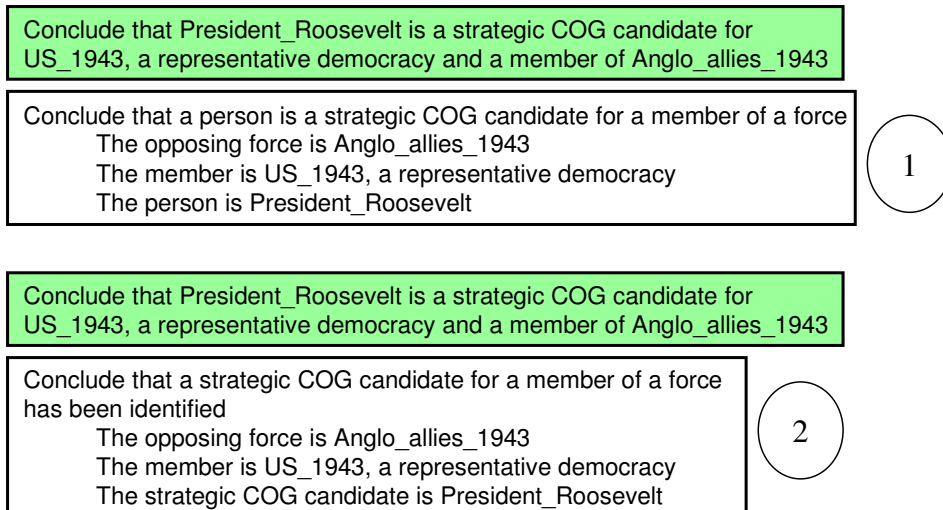


Figure 8 - Variations of a Formalized Solution

## Language to Logic Translation Example from the Disciple Environment

The following section describes an example of how this modeling methodology supports the completion of the language to logic translation process. While the final formal representation of the logic is specific to the Disciple environment, the description of the general process should be useful in any MAS development environment.

After the expert formalizes a task or solution, Disciple forms a matching internal view that it uses in follow-on steps of the language to logic translation process, as depicted in Figure 9. The expert was responsible for first creating the natural language version of a specific task (A) and then formalizing it to create (B). The Disciple agent learns a general task from this specific task by replacing the instances, *Anglo\_allies\_1943* and *US\_1943*, with the variables *?O1* and *?O2*, respectively. From the natural language expression of the specific task (A) the agent learns the informal structure of the general task (C). From the formalized expression of the specific task (B), the agent learns the formal structure of the general task (D).

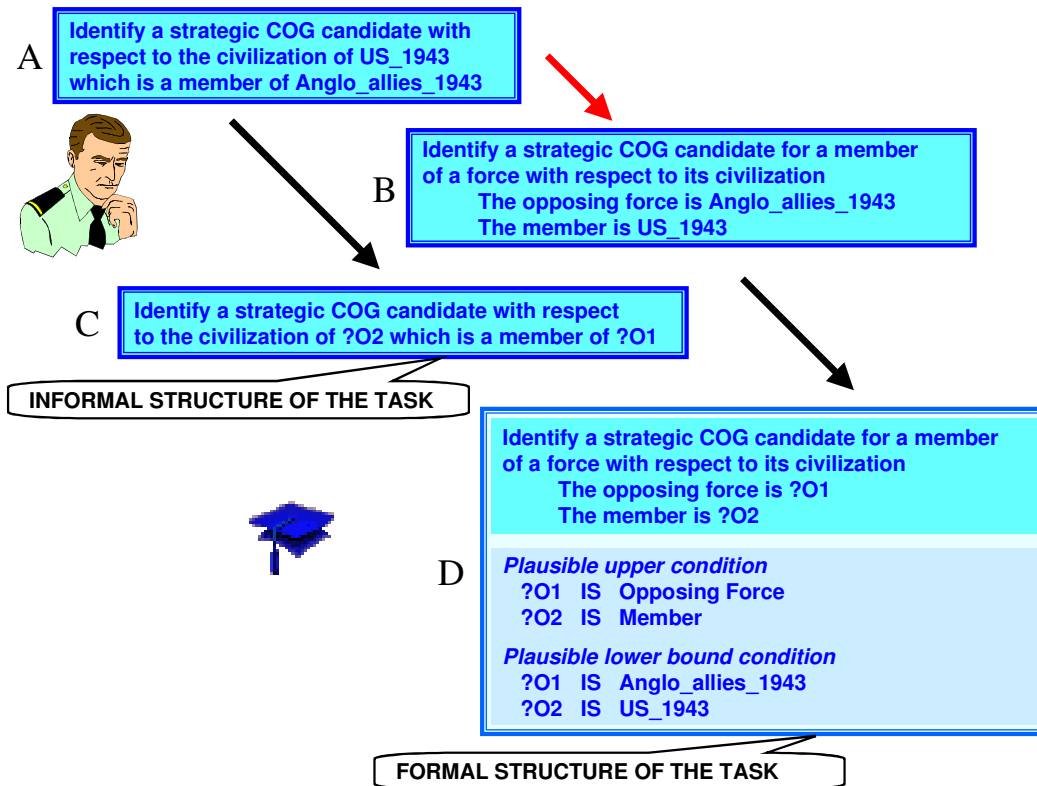


Figure 9 – Disciple’s Internal View of a Formalized Task

## Completing the Natural Language to Logic Translation through Agent Training

The questions and answers in Figure 7 provide the justification or explanation for the task reductions in the figure. While these explanations are very natural to a human expert, Disciple cannot understand them because they are written with unrestricted natural language. Figure 10 shows another task reduction, with both the initial and formalized tasks, which lead to a conclusion. The question and answer in this figure again provide a justification for this reduction that a human expert would understand. A translation of this justification that could be understood by Disciple is expressed in the three larger, un-boxed expressions, which consist of various relationships between relevant elements from the agent's ontology. When the agent's ontology is developed it must include the objects, features and relationships that will be part of these explanations.

In the agent-training process, the expert and the agent collaborate to identify the correct sequence of problem-solving steps in the form of the formalized tasks/sub-tasks/solutions. The expert and agent then establish the links between the steps by identifying the justifications for the reduction from one step to the next. This is done by performing the language to logic translation where the expert's expressive, natural language question and answer are converted to logic that can be understood by the agent, in the form of a sequence of object relationships that correspond to a

fragment of the object ontology. The training process completes the natural language to logic translation and results in problem-solving rules that are understood and useable by the agent.

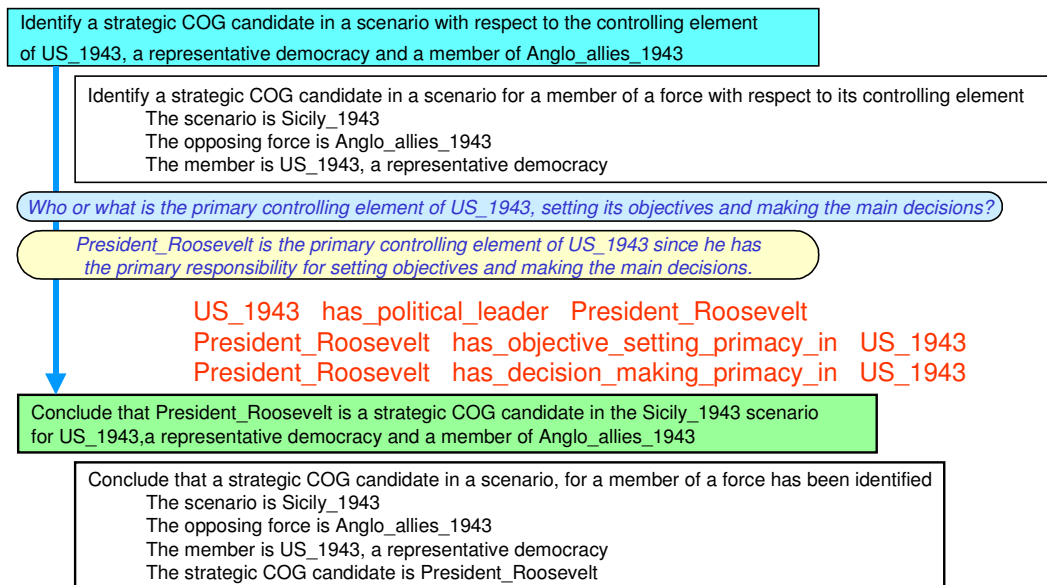


Figure 10 - Disciple Useable Reduction Justifications

In agent training with Disciple, the expert uses a point and click interface to identify key concepts, instances and types of relationships present in the question and answer. The agent uses analogical reasoning heuristics and general heuristics to generate a list of plausible explanations from these hints. The expert reviews the proposed explanations and selects those that match the logic in the natural language question and answer. If the desired explanations are not initially found, the expert can provide additional hints and prompt the agent to generate additional potential explanations until the necessary explanations are generated and selected by the expert.

Figure 11 is an example of a language-to-logic translation. The question and its answer on the left side of the figure represent the reasoning process of the expert. This natural language explanation must be translated to the ontology-based explanation on the right side of the figure, consisting of the following relations between relevant elements from the agent's ontology:

US\_1943 → has\_as\_industrial\_factor → Industrial\_capacity\_of\_US\_1943  
 Industrial\_capacity\_of\_US\_1943 → is\_a\_major\_generator\_of →  
 War\_materiel\_and\_transports\_of\_US\_1943  
 War\_materiel\_and\_transports\_of\_US\_1943 → is →  
 Strategically\_essential\_goods\_or\_materiel.

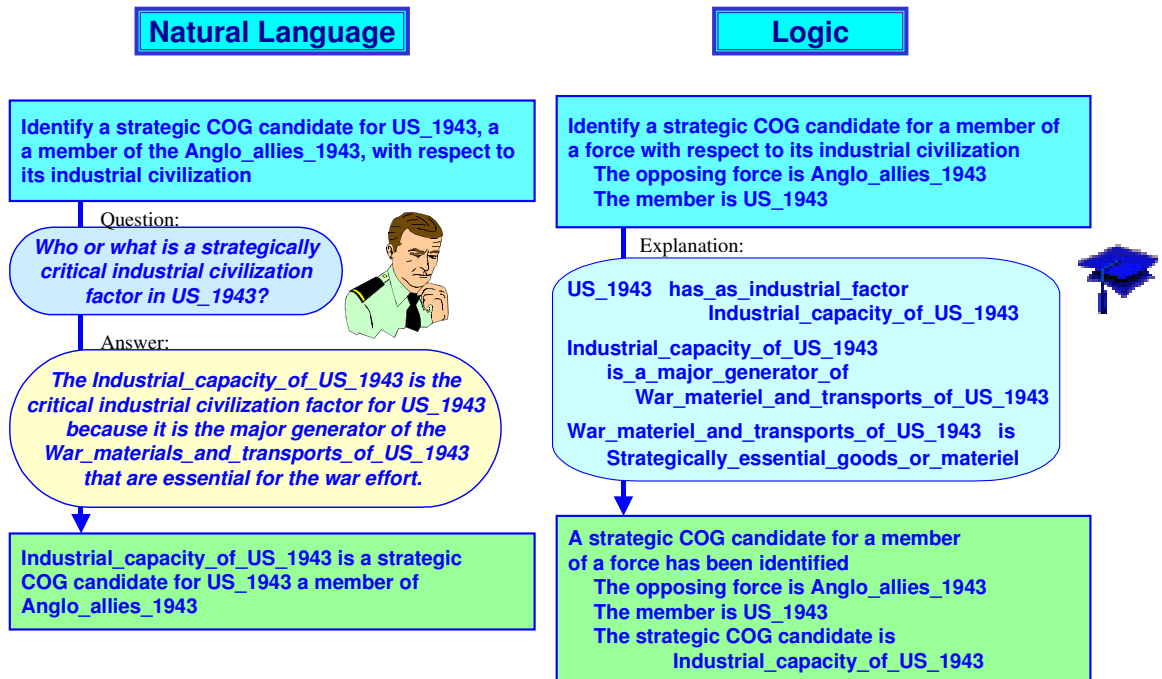


Figure 11 - Agent Training

These lines state, in Disciple's language, that *US\_1943* has as an industrial factor its industrial capacity, which is a major generator of war materiel and transports, which are strategically essential to the war effort. Thus, matching the logic from the expert's question and answer that the industrial capacity of the US in 1943 is a strategically critical element of its industrial civilization.

An expert is likely to be able to understand these formal expressions because they directly correspond to the natural language explanations generated by the expert during modeling. Not being a KE however, the expert is not likely to be able to create them. For one thing, the expert would need to use the formal language of the agent. But this would not be enough. The expert would also need to know the names of the potentially many thousands of concepts and features from the agent's ontology (such as "*is\_a\_major\_generator\_of*").

While defining the formal explanation of this task reduction step is beyond the individual capability of the expert or the agent, it is not beyond their joint capabilities. Finding these explanation pieces is a mixed-initiative process involving the expert and the agent. Once the expert is satisfied with the identified explanation pieces, the agent generates the rule in Figure 12.

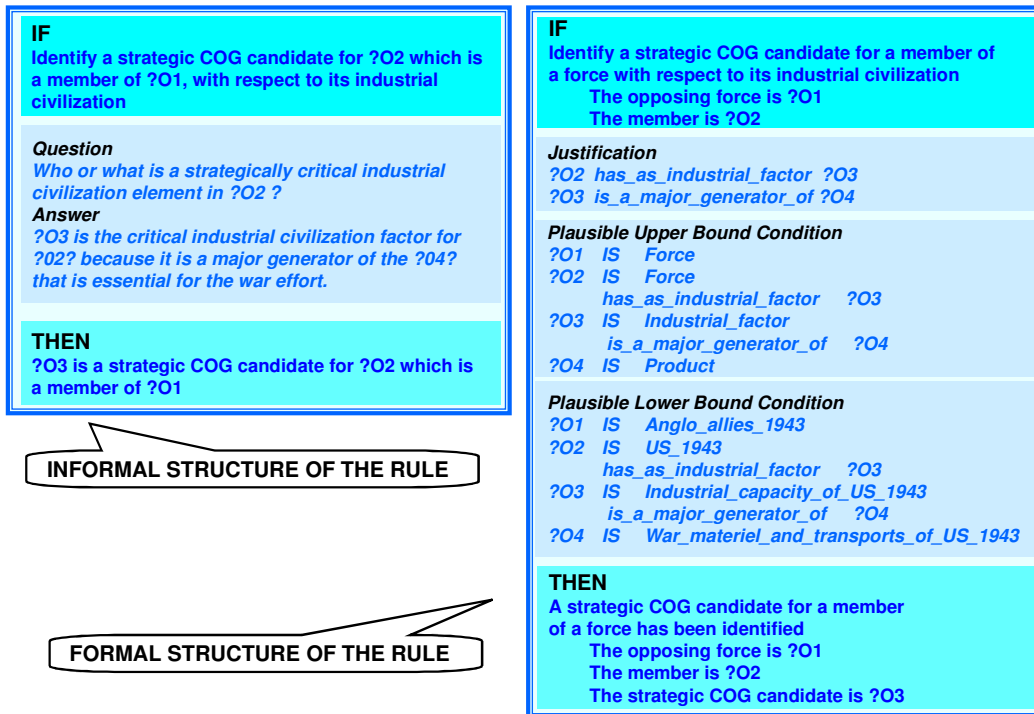


Figure 12 – Informal and Formal Structure of a Learned Rule

The learned rule in Figure 12 has both an informal structure and a formal structure. The formal structure contains the formal structures of the tasks, a general justification, and two applicability conditions. The two applicability conditions approximate the exact applicability condition to be learned by the agent through rule refinement. During rule refinement, the agent will generalize the plausible lower bound condition and specialize the plausible upper bound condition until they essentially meet, allowing the variables to take on only values that produce correct reductions.

### Agent Feedback to Domain Modeling

Figure 13 shows an entire agent development process, adding a fourth and fifth step and feedback loops to Figure 6. As stated earlier, domain modeling in support of agent development is a complex, non-linear process, involving many feedback loops. Domain modeling and ontology development go on in a nearly concurrent manner throughout the process. A significant amount of domain modeling and ontology development must be completed before formalization is practical, and before agent training is possible.

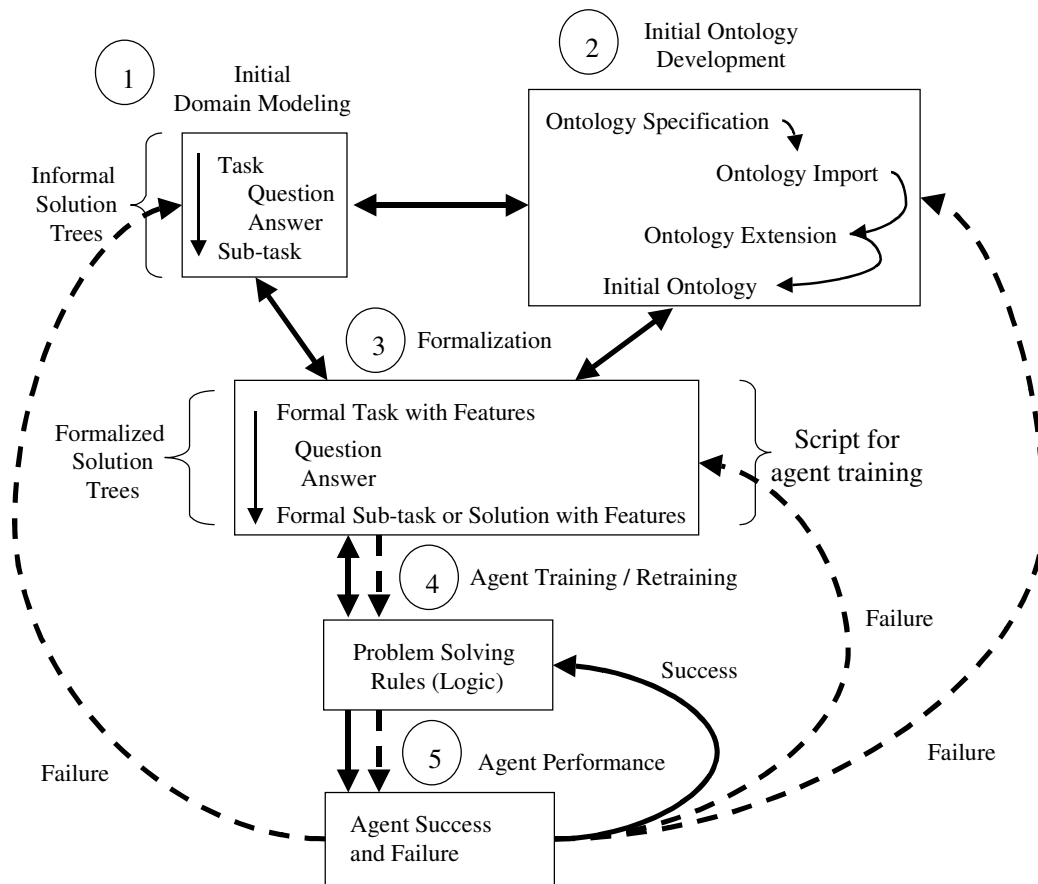


Figure 13 - Agent Development Process with Feedback Loops

Initial domain modeling and initial ontology development will drive and feed one another. Since even minor changes in the domain modeling or ontology could require major changes in formalized modeling, formalization and agent training should not be done until the domain model and ontology are relatively stable with regard to examples to be used for agent training.

As suggested by the feedback loops in Figure 13, changes and rework in the overall process cannot be avoided. It is unrealistic to expect that either the domain modeling or the ontology will ever be comprehensive for any real-world application domain.

### A General Representation of the Developed Methodology

Modeling expert problem solving in the manner described above is a complex, cyclic, and iterative process, but it retains basic task reduction properties. Figure 14 places the basic task reduction diagram from Figure 2 next to a new version where new constructs for questions Q, and answers A, are added.

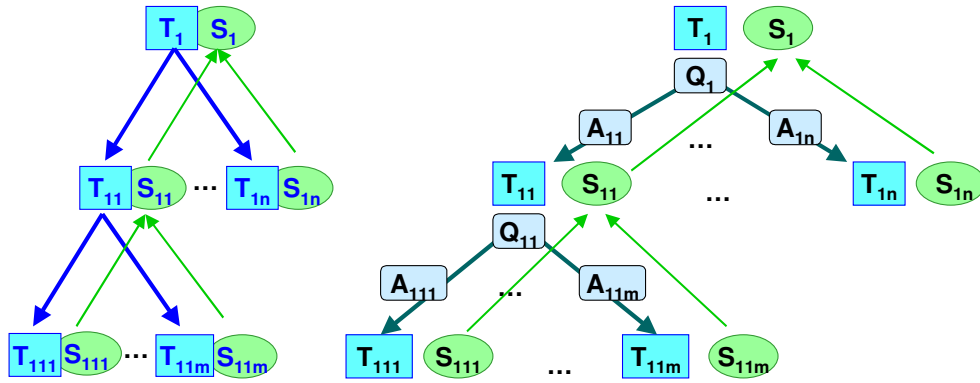


Figure 14 – Revised Task Reduction Diagram with Tasks, Questions, and Answers

Both diagrams in Figure 14 express the idea that problem-solving steps are represented in their structures. The new diagram on the right side of Figure 14 is expanded to express the idea that in addition to problem-solving steps, new information is represented in the form of the questions and answers portrayed. This new diagram however, still falls short of representing the methodology. Still missing are representations for the ontology specification, state information, and expert reasoning that emerge from the content rich natural language task names, questions and answers the methodology generates. Figure 15 builds on Figure 14 to more fully represent the product the methodology generates during the language to logic translation of problem-solving knowledge. The expression following Figure 15 summarizes the top half of the figure and is followed by an explanation of the terms.

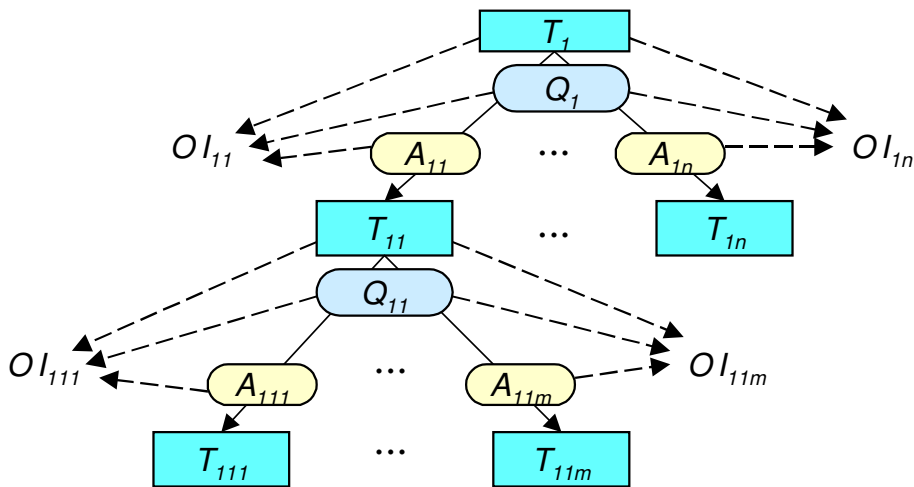


Figure 15 - A General Example of a Partial Solution Tree

$$\{T_1 Q_1 \{(A_{11}, OI_{11}, T_{11}), \dots (A_{1n}, OI_{1n}, T_{1n})\}$$

Where  $T_1$  is the top-level problem-solving task, and  $Q_1$  is the first question asked. The answers to  $Q_1$  are  $A_{11}$  to  $A_{1n}$ . Additionally,  $OI_{11}$  is the problem-solving knowledge generated through the language to logic translation of  $T_1$ ,  $Q_1$ , and  $A_{11}$ , where:

$O$  - is the set of ontological elements (objects and their relationships) revealed by the corresponding  $T$ ,  $Q$  and  $A$  and expressed in the ontology specification;

$I$  - is the set of state information (relevant current characteristic conditions, or variable values) and expert reasoning conceptually contained in the corresponding  $T$ ,  $Q$  and  $A$ ;

and  $T_{11}$  is the sub-problem of  $T_1$  reduced by  $Q_1$  and  $A_{11}$ , and accumulating  $OI_{11}$ .

In each case the reduced problem-solving task  $T_{1i}$  generally accumulates the identified relevant facts from the preceding  $T$ ,  $Q$  and  $A$ . All of these expressions are free form natural language sentences that include objects and relationships whose meaning remain consistent for a given problem-solving example.

The arrows from  $T_1$  through  $Q_1$  and  $A_{11}$  to  $T_{11}$  in Figure 15 are solid to denote a direct relationship between them. The arrows from the  $T_1 Q_1 A_{11}$  set to  $OI_{11}$  are dotted because the identification and representation of the ontology structures and state information are closely related but separate processes that are begun during initial domain modeling, but not completed until the entire language to logic translation process is completed. The  $O$  of  $OI_{11}$  is captured initially in the construction of the ontology specification process described above, and completed in the ontology development process. The  $I$  of  $OI_{11}$  is conceptually contained in the logic of the natural language  $T_1$ ,  $Q_1$  and  $A_{11}$  but is not formally captured and expressed until formal problem-solving rules are generated by the agent-training process.

Continuing the reduction in Figure 15 on  $T_{11}$  leads to a second level representation of:

$$T_{11} Q_{11} \{(A_{111}, OI_{111}, T_{111}), \dots (A_{11m}, OI_{11m}, T_{11m})\}$$

A chain of reductions is continued until a solution is obtained:

$$T_{1k} Q_{1k} \{(A_{1k}, OI_{1k}, S_{1k})\}$$

Whenever a reduction creates a result, that solution is back propagated and combined with other solutions through the solution tree substituting for the 3-tuple of answer, ontological element set/state information, and sub-task.

Thus a level of reasoning is concluded when:

$$T_{\delta} Q_{\delta} \{(A_{\alpha}, OI_{\alpha}, S_{\alpha}), (A_{\beta}, OI_{\beta}, S_{\beta}) \dots (A_{\eta}, OI_{\eta}, S_{\eta})\}$$

The entire problem is solved when:

$$\{TQ\{(S_1), (S_2), \dots (S_n)\}\}$$

This leads to a final expression for the methodology where the knowledge ( $K$ ) necessary to solve a given problem within a class of problems is the total ( $T$ ) of all the knowledge accumulated in task reduction and solution composition:

$$K_T = \{T_T Q_T (A_T, OI_T, S_T)\}$$

That is, the knowledge  $K_T$  required to solve a given task  $T$  is the union of the knowledge contained in all of the problem-solving tasks  $T_T$ , questions  $Q_T$ , answers  $A_T$ , and the solutions  $S_T$ . This knowledge is represented in all of the ontology elements generated in ontology development, and conceptually contained in all of the state information and expert problem-solving logic contained in  $OI_T$ .

## **Initial Modeling Discussion and General Lessons Learned**

### **Getting Started**

When using this methodology to model expert problem solving for the first time with a new problem domain, or with a new class of problems within a domain, three general options are available to the modeler:

Depth first. The expert begins with a depth first analysis, modeling with great detail, a complete solution to a specific example problem.

Breadth first. The expert begins with a breadth first analysis of the problem domain, brainstorming a wide range of potential solutions to many types of problems, before examining any of those solutions in-depth.

Combination. The third approach is a logical combination of the first two. In this Combination approach the expert may begin with a fairly clear understanding of what the problem is, and have in mind one or more potential solutions to the problem. After categorizing and prioritizing the potential solutions the expert begins modeling the highest priority potential solutions, but at each question considers branches and alternative solutions. The expert continues modeling the selected solution to completion, but as each new branch or alternative is considered the expert makes notes to facilitate future modeling of that branch or alternative.

In practice, the Combination approach is likely to occur. This is because the complete set of all possible solutions to a given problem is not likely to be fully identified during a purely breadth first examination, and a depth first analysis of a single possible solution naturally identifies likely alternative solutions or branches to the solution being modeled.

Experimental results (Tecuci et al., 2000, Tecuci et al., 2001, Tecuci et al., 2005) show that a natural approach to the domain modeling process is to:

1. Brainstorm possible categories of solutions to the broad problem domain.
2. Identify a specific, example problem to be solved.
3. Brainstorm potential solutions for that problem within a solution category.
4. Complete in-depth modeling of one or more promising solutions within a category.
5. Repeat steps as necessary.
6. Reevaluate/revise modeled solutions, adding levels of detail, based on lessons learned during subsequent modeling.

## **Task Reduction**

A second major procedural issue is the manner in which a complete in-depth modeling of a solution is completed. Again, three approaches are possible:

Steps First The expert begins by identifying all the steps leading to a solution or conclusion without a detailed analysis of the logic that justifies the step-to-step progression. One or more solutions are modeled in this manner before each step is analyzed and modeled in detail.

Details First The expert begins by identifying one logical step along the solution tree, and then thoroughly identifies and documents all of the necessary knowledge elements and conditions necessary to justify the progression to the next step, before additional steps are identified.

Combination The third approach is a logical combination of the two pure approaches.

Experimental results strongly suggest that the two pure approaches to in-depth solution modeling are not practical. The first approach seems natural and can be productive but it has serious drawbacks. In identifying all of the necessary steps before specifying any of the justifications for step-to-step progress, it is possible for the expert to lose track of the purpose or the reasoning behind a particular step by the time the end of the solution tree is reached. This approach also routinely produced steps that were too complex to adequately describe and justify later, which results in the steps being subsequently broken down into finer, simpler steps. Finally, this approach does not lend itself to incremental or team work. In long, complex solutions, it is difficult for the expert to stop work and efficiently pick it up again later, and extremely difficult for a development team member to understand the incomplete model when there is no documented justification for proceeding from one step to the next.

The second pure approach to in-depth modeling of a solution is also impractical. In this case the modeling proceeds much too slowly and the task descriptions and justifications are very rarely actually complete until a solution has been modeled to a logical conclusion and has been looked at several times. Additionally, if analysis of a domain supports the import and reuse of an existing ontology much of the work done early in this approach will have to be revised to use the terminology from the imported ontology.

Finally, the level of detail actually necessary for productive work increases steadily during the stages of MAS development. Early, a minimal level of detail is necessary to support ontology

development and import, and levels of detail beyond this are unnecessary and are likely to require major revision later. As work progresses to the later stages of knowledge base and agent development, additional levels of detail are necessary and are generated naturally in the cyclic feedback between modeling, agent training, and performance phases.

### **Modeling in Big versus Small Steps**

Another procedural issue for the methodology is the level of complexity or simplicity that is appropriate for each step in a solution tree. Experts often solve problems in what appears to be a “big bang” approach. They study a problem, collect relevant information, and announce a solution all in one, two or a few, complex steps. Asked to describe their process, they may write one very long, complex paragraph describing how they solved the problem. An expert may attempt to do domain modeling the same way. Using the proposed methodology, this may generate simple task names, but will require very complex questions and answers to adequately describe and justify the progression between these complex steps.

An alternate approach is to break down a problem solution into very simple, possibly tediously fine incremental steps. This approach results in more steps, possibly more complex task names, but generally produces much simpler questions and answers describing and justifying the progression between steps.

The first drawback to using complex steps is that as a problem-solving method, it does not scale up well to complex problems. When too much occurs in a single step, solutions and the justifications for moving from one step to the next easily become too complex to adequately describe or represent. This approach also disregards the primary goal of task reduction, which is breaking complex problems into simpler problems so that they may be more easily understood and solved. The primary drawback to this approach from the standpoint of knowledge acquisition and agent development is extremely important. The fundamental purpose for developing these representations of problem-solving knowledge is so they can be used to develop more general, re-useable problem-solving rules. Doing too much in a single step makes the step less general and much more difficult to turn into a re-useable problem-solving rule. The representation itself is also much more easily re-used for representing other problem-solving examples when it is broken down into finer steps. Most importantly this incremental approach generates a representation that is more suitable for generating general, re-usable problem-solving rules.

## **CONCLUSION**

The methodology described here provides a practical set of methods for modeling and representing an expert’s problem-solving knowledge for an application domain, and supporting ontology development and import, teaching-based intelligent agent development, and agent-based problem solving. It is suitable for use in domains where task reduction can be used for problem solving. Detailed experimental results for using the methodology have been reported as part of the DARPA HPKB and RKF efforts (Bowman 2002, Tecuci, et al., 2005). In general, those results show that the methodology is both effective for modeling complex domains in support of agent development, and easy to use with after minimal user training. While it was

developed for and has been used extensively with a specific agent environment, the GMU LALAB Disciple system, it has very general applicability for MAS development. The methodology is only a first step in the development of agent based systems. Research and experimentation in using the methodology to support the development of collaborative agents is ongoing. Use of the methodology to determine agent responsibilities in MAS is an excellent research opportunity.

## References:

- Alvares, L., Menezes, P., & Demazeau, Y., (1998). Problem Decomposition: an Essential Step for Multi-Agent Systems, *10th International Conference on Systems Research, Informatics and Cybernetics (ICSRIC'98)*. Baden-Baden.
- Angele, J., Fensel, D., and Studer, R., (1998). Developing Knowledge-Based Systems with MIKE, in *Journal of Automated Software Engineering*.
- Bowman, M., (2002). *A Methodology for Modeling and Representing Expert Knowledge that Supports Teaching-Based Intelligent Agent Development*. PhD Dissertation, George Mason University, Fairfax, VA.
- Bowman, M., Tecuci, G., Boicu, M., & Commello, J., (2004). Information Age Warfare – Intelligent Agents in the Classroom and Strategic Analysis Center. In *Proceedings of the 24<sup>th</sup> U.S. Army Science Conference*, Orlando FL.
- Brazier, F., Jonker, C., & Van Treur, J., (1998). Principles of Compositional Multi-agent System Development. In Cuenca, J. (Ed.). *15th IFIP World Computer Congress, WCC'98, Conference on Information Technology and Knowledge Systems, Evaluating PSMs for Adaptive Design 23 (IT&KNOWS'98)*, (347-360).
- Chavez, A., Moukas, A., & Maes, P., (1997). *Challenger: A multiagent system for distributed resource allocation*. In: *Proceedings of the First International Conference on Autonomous Agents*, Marina Del Ray, CA, ACM Press.
- Clausewitz, von C., (1832). *On War*, Howard, M., and Paret, P., (translators). Princeton, NJ: Princeton University Press (1976).
- Cohen, R., Allaby, C., Cumbaa, C., Fitzgerald, M., Ho, K., Hui, B., Latulipe, C., Lu, F., Moussa, N., Pooley, D., Qian, A., & Siddiqi, S., (1998). What is Initiative? *User Modeling and User-Adapted Interaction* Vol. 8, No. 3-4, 173.
- Giles, P., and Galvin, T., (1996). *Center of Gravity: Determination, Analysis, and Application*, Carlisle Barracks, PA: Center for Strategic Leadership.
- Grosso, W., Eriksson, H., Ferguson, R., Gennari, J., Tu, S., and Musen, M., (1999). Knowledge Modeling at the Millennium, The Design and Evolution of Protégé – 2000. In *Proceedings of the Twelfth Workshop on Knowledge Acquisition, Modeling and Management*, Banff, Alberta, Canada, October 16-21.
- Hernandez, F., Gray, J., & Reilly, K., (2003). A Multi-Level Technique for Modeling Agent-Based Systems. *2<sup>nd</sup> International Workshop on Agent-Oriented Methodologies*, (OOPSLA-2003), (pp. 33-42). Anaheim, CA.

Luck, M., d'Inverno, M., Fisher, M., and FoMAS'97 Contributors, (1998). Foundations of Multi-Agent Systems: Techniques, Tools and Theory. *Knowledge Engineering Review*, 13(3), 297-302.

Maes, S., Tuyls, K., & Manderick, B. (2001). Modeling a Multi-agent Environment Combining Influence Diagrams. Mohammadian, M. (Ed.), *Intelligent Agents, Web Technologies and Internet Commerce, IAWTIC2001*, Las Vegas, NV.

Minsky, M., (1985). *The Society of Mind*, New York: Simon and Schuster.

Mustafa, M., (1994). *Methodology of Inductive Learning: Structural Engineering Application*. PhD Dissertation, Wayne State University, Detroit, MI.

Newell, A., & Simon, H., (1972). *Human Problem Solving*, Englewood Cliffs, NJ: Prentice Hall.

Schreiber, A., Wielinga, B., and Breuker, J., (editors) (1993). KADS. A Principled Approach to Knowledge-Based System Development, *Knowledge-Based Systems*, Vol. 11, Academic Press, London.

Schreiber, G., Akkermans, H., Anjewierden, A, de Hoog, R., Shadbolt, N., Van de Velde, W., and Wielinga, B., (2000). *Knowledge Engineering and Management; the CommonKADS Methodology*, MIT Press, Cambridge.

Simon, H., (1977). *The New Science of Management Decision*, rev. ed. Englewood Cliffs: Prentice Hall; quoted in Van Gundy Jr., (1981). A., *Techniques of Structured Problem Solving* (pp. 5). (New York: Van Nostrand Reinhold Company.

Tecuci, G., (1998). *Building Intelligent Agents*, San Diego, CA: Academic Press.

Tecuci, G., Boicu, M., Bowman, M., Marcu, D., Syhr, P., & Cascaval, C., (2000). An Experiment in Agent Teaching by Subject Matter Experts. *International Journal of Human-Computer Studies*, Academic Press, 53, 583-610.

Tecuci, G., Boicu, M., Bowman, M., & Marcu, D., with a commentary by Murray Burke, (2001). An Innovative Application from the DARPA Knowledge Bases Program, Rapid Development of a Course of Action Critiquer. *AI Magazine*, Cambridge, AI Press, 22-2, 43-61.

Tecuci, G., Boicu, M., Boicu, C., Marcu, D., Stanescu, B., Barbulescu, M., (2005). The Disciple-RKF Learning and Reasoning Agent. *Computational Intelligence*. Vol. 21, No. 4, pp. 462-479.

Uma, G., Prasad, B., and Kumari, O., (1993). Distributed Intelligent Systems – Issues, Perspectives and Approaches, *Knowledge Based Systems*. (pp. 77-96) Vol. 6, No. 2.