Ontologies and the Knowledge Acquisition Bottleneck

Mihai Boicu, Gheorghe Tecuci, Bogdan Stanescu, Gabriel C. Balan and Elena Popovici
Learning Agents Laboratory, Department of Computer Science, MS 4A5
George Mason University, 4400 University Drive, Fairfax, VA 22030-4444
{mboicu, tecuci, bstanesc, gbalan, epopovic}@gmu.edu, http://lalab.gmu.edu

Abstract

Ontologies and information sharing have a major role to play in the development of knowledge-based agents and the overcome of the knowledge acquisition bottleneck. This paper supports this claim by presenting an approach to ontology specification, import, and development that is part of Disciple-RKF. Disciple-RKF is a theory, methodology, and learning agent shell for the rapid development of knowledge-based agents by subject matter experts, with limited assistance from knowledge engineers. The Disciple approach has been subject of intensive evaluations, as part of DARPA's "High Performance Knowledge Bases" and "Rapid Knowledge Formation" programs, demonstrating very good results.

1 Introduction

Ontologies and information sharing have a major role to play in the development of knowledge-based agents and the overcome of the knowledge acquisition bottleneck [Buchanan and Wilkins, 1993]. Indeed, building a knowledge base is too difficult a task to always start from scratch when a new knowledge-based system needs to be created. It makes more sense to reuse knowledge from related knowledge bases than to recreate such knowledge because this process should, in principle, be easier. Moreover, this reuse should also facilitate the communication between the systems because of their shared knowledge.

However, knowledge sharing and reuse are in themselves very complicated processes, especially if the systems involved have not been specifically designed for this purpose. How to design a knowledge-based system to facilitate knowledge sharing or reuse is an open research question.

In this paper we present an approach to rapid development of knowledge-based agents that illustrates several general methods and ideas related to ontology reuse and development. This approach is implemented in the Disciple-RKF learning agent shell.

Disciple-RKF is a tool for the development of a knowledge-based agent directly by a subject matter expert, with limited assistance from a knowledge engineer. Disciple-RKF contains a general problem solving engine, a learning engine and an initially empty knowledge base. The process of developing a Disciple agent for a specific application relies on importing ontologies from existing repositories of knowledge, and on teaching Disciple how to perform various tasks, in a way that resembles how an expert would teach a human apprentice when solving problems in cooperation. While teaching Disciple how to solve problems is a major feature of this system, in this paper we concentrate on its ontology-related aspects.

The next section describes the architecture of the Disciple-RKF shell. An important feature of this architecture is the structuring of the knowledge base into a general object ontology that can be imported and a set of problem solving methods or rules that can be learned from a subject matter expert.

Section 3 presents the general domain modeling methodology used with the Disciple approach. A characteristic feature of this methodology is that it produces an initial specification of the object ontology needed for the application knowledge base being developed. This ontology specification is the input to the ontology import module that is described in section 4. This module implements a general approach to ontology import.

Section 5 discusses several intelligent assistants that help in the complex process of extending and improving the object ontology. Then section 6 presents a practical approach for eliciting instances from subject matter experts, to populate the object ontology.

Section 7 discusses briefly the process of agent teaching and rule learning. This is continued in section 8 with a discussion of the ontology learning issue.

The knowledge base developed through the processes mentioned above can also be exported into existing knowledge servers, for further reuse. The knowledge export method of the Disciple approach is presented in section 9.

The work reported here has been done as part of the DARPA's High Performance Knowledge Bases program [Cohen *et al.*, 1998], and continues as part of the Rapid Knowledge Formation program [Burke, 1999]. These programs included intensive experimentation periods that tested the claim that with the latest AI technology knowledge bases can be built quickly and efficiently. The tests required the development of knowledge-based systems for solving several challenge problems, including the following ones: 1) the workaround challenge problem:

planning the repair of damaged bridges and roads [Jones, 1998; Tecuci *et al.*, 2000a]; 2) the COA challenge problem: critiquing military courses of action [Jones, 1999; Tecuci *et al.*, 2000b], and 3) the COG challenge problem: identifying strategic center of gravity candidates in military conflicts [Gilles *et al.*, 1996]. In section 10 we present experimental results from these evaluations that support the claims made in this paper.

We conclude the paper with a discussion of our future research related to ontology and information sharing.

2 The Disciple-RKF Learning Agent

Disciple-RKF contains a domain modeling and problem-solving engine that is based on the general problem (or task) reduction paradigm of problem solving, and is therefore applicable to a wide range of domains. In this paradigm, a problem to be solved (or a task to be performed) is successively reduced to simpler problems until the problems are simple enough to be immediately solved. Their solutions are then successively combined to produce the solution to the initial problem.

An important feature of Disciple-RKF is the structuring of the knowledge base into two distinct components: an object ontology and a set of reduction and composition rules. The object ontology is a hierarchical representation of the objects and types of objects from a particular domain, such as military or medicine. That is, it represents the different kinds of objects, the properties of each object, and the relationships existing between objects. The object ontology provides a representation vocabulary that is used in the description of the reduction and composition rules. Each reduction rule is an IF-THEN structure that expresses the conditions under which a problem (or task) P_1 can be reduced to the simpler problems (tasks) P_{11}, … , P_{1n}. Similarly, a composition rule is an IF-THEN structure that expresses the conditions under which the solutions S_{11}, … , S_{1n} of the problems (tasks) P_{11}, … , P_{1n} can be combined into a solution S_1 of P_1.

Dividing the knowledge base into an object ontology and a set of rules is very important because it clearly separates the most general part of it (the object ontology), from its most specific part (the rules). Indeed, an object ontology is characteristic to an entire domain. In the military domain, for instance, the object ontology will include descriptions of military units and of military equipment. These descriptions are most likely needed in almost any specific military application. Because building the object ontology is a very complex task, it makes sense to reuse these descriptions when developing a knowledge base for another military application, rather than starting from scratch. In the case of Disciple-RKF the ontology reuse is further facilitated by the fact that the objects and the features are represented as frames, based on the knowledge model of the Open Knowledge Base Connectivity (OKBC) protocol. OKBC has been developed as a standard for accessing knowledge bases stored in different frame representation systems [Chaudhri *et al.*, 1998]. Therefore, importing an ontology from an OKBC compliant knowledge server, such as Loom [MacGregor, 1999], Ontolingua [Farquhar *et al.*, 1996], and Protégé [Fridman *et al.*, 2000] does not raise translation problems.

The rules from the knowledge base are much more specific than the object ontology. Consider, for instance, two agents in the military domain, one that critiques courses of action with respect to the principles of war, and another that plans the repair of damaged bridges or roads. While both agents need to reason with military units and military equipment, their reasoning rules are very different, being specific not only to their particular application (critiquing vs planning), but also to the subject matter experts whose expertise they encode.

3 Domain Modeling and Problem Solving

Domain modeling is the first and the most difficult activity when developing a knowledge base. First, the subject matter expert and the knowledge engineer have to develop a model of the application domain that will make explicit, at a qualitative and informal level, the way the subject matter expert performs tasks. In the case of Disciple-RKF this means modeling the process of performing a specific task as a sequence of qualitative and informal task reduction and composition steps. The knowledge engineer and the subject matter expert will consider a set of specific tasks that are representative of the set of tasks that the final agent should be able to perform. Then, for each of these tasks, they will represent the problem solving process as a sequence of task reductions (and, possibly, task composition) steps.

The left hand side of Figure 1, for instance, represents an example of task reduction modeling from the Course of Action critiquing domain. The task to perform is "Assess COA411 with respect to the Principle of Surprise". To perform this assessment, the expert needs a certain amount of information about COA411. This information is obtained through a series of questions and answers that help reduce the initial assessment task to simpler and better-defined ones, until the expert has enough information to perform the assessment: "Report strength in surprise for COA411 because of countering enemy recon."

A main result of this modeling process is that it identifies the concepts and the features that need to be part of the object ontology in order for the agent to perform the type of reasoning illustrated in Figure 1. Indeed, the reasoning steps from the left hand side of Figure 1 reveal the need for the concepts and the features from the right hand side of Figure 1. The collection of all these concepts and feature represent a specification of the ontology that will have to be developed. In our approach, this specification guides the import of relevant ontological knowledge from external repositories such as CYC [Lenat, 1995], Loom [MacGregor, 1999], or Ontolingua [Farquhar *et al.*, 1996], as will be presented in the next section.

A second result of the modeling process are the task reduction steps themselves. They represent problem solving examples from which the Disciple agent will learn general

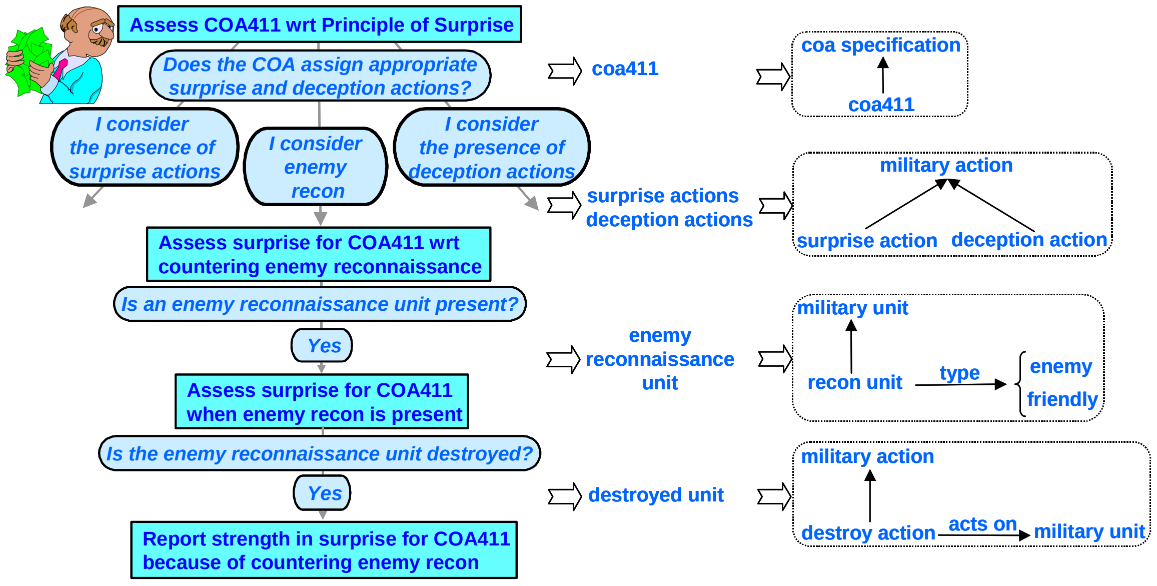

Assess COA411 wrt Principle of Surprise
Does the COA assign appropriate surprise and deception actions?
I consider the presence of surprise actions
I consider enemy recon
I consider the presence of deception actions
Assess surprise for COA411 wrt countering enemy reconnaissance
Is an enemy reconnaissance unit present?
Yes
Assess surprise for COA411 when enemy recon is present
Is the enemy reconnaissance unit destroyed?
Yes
Report strength in surprise for COA411 because of countering enemy recon
coa411
coa specification
coa411
surprise actions
deception actions
military action
surprise action
deception action
enemy reconnaissance unit
military unit
recon unit
type
enemy
friendly
destroyed unit
military action
destroy action
acts on
military unit

Figure 1: An illustration of the Disciple modeling process in the COA domain.

rules through the application of a mixed-initiative multistrategy learning method [Boicu *et al.*, 2000].

4 Ontology Import

As presented in the previous section, when the knowledge engineer works with the subject matter expert to define an initial domain model, they also identify the type of objects and features that are needed in the knowledge base (see Figure 1). These objects and features will focus the process of importing relevant ontological knowledge from existing knowledge repositories. The architecture of the ontology import module of Disciple is represented in Figure 2. Basically there are three phases of the ontology import process: 1) mixed-initiative retrieval of potentially relevant ontological knowledge from an external knowledge repository; 2) automatic translation of the retrieved ontological knowledge into an intermediate Disciple ontology; and 3) mixed-initiative import from the intermediate Disciple ontology into the final Disciple ontology. Each of these phases is discussed bellow.

In general, one of the practical difficulties encountered in ontology import is the fact that the subject matter expert has to deal with the additional representation system and tools of the knowledge repository from where knowledge has to be imported. To alleviate this problem, for each knowledge repository from which we are importing knowledge in Disciple, a standard ontology retrieval interface is developed. This interface allows the subject matter expert to retrieve relevant knowledge from different representation systems without dealing with the tools or representation of that knowledge repository. In the current Disciple architecture there are three planned implementations of the standard interface, one for the CYC system, which already exists, another one for any OKBC-compliant knowledge repository, such as Loom [MacGregor, 1999], Ontolingua [Farquhar *et al.*, 1996], or Protégé [Fridman *et al.*, 2000], and another one for older Disciple repositories.

Figure 3 illustrates the process of mixed-initiative retrieval of relevant ontological knowledge from the CYC knowledge repository. The subject matter expert introduces one of the terms needed in the ontology to be developed. A specialized CYC-searching module retrieves CYC terms that are likely to correspond to the input term, together with their documentation and pretty-names. Then the subject matter expert selects from the retrieved terms those that actually correspond semantically to the input term. This process is repeated for all the terms identified as relevant during domain modeling and results into a set of relevant

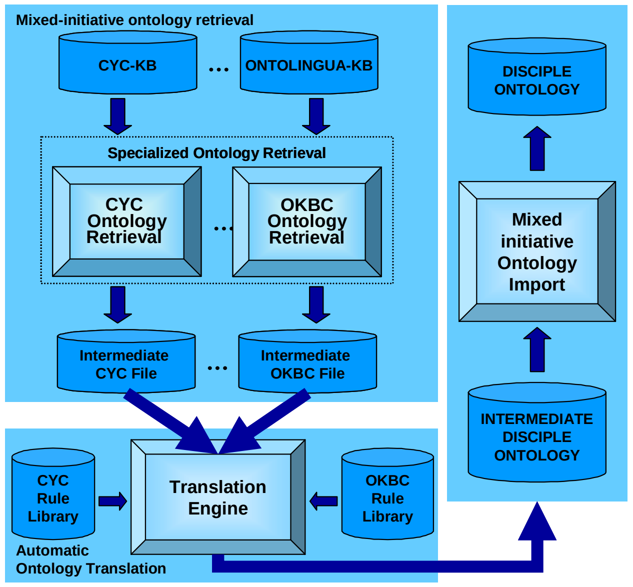


Figure 2: The Ontology import module.

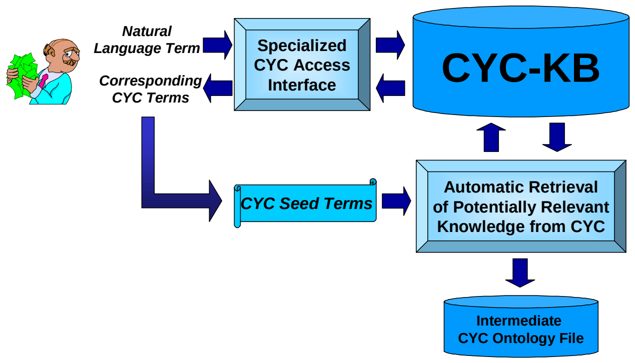

Natural Language Term
Corresponding CYC Terms
Specialized CYC Access Interface
CYC-KB
CYC Seed Terms
Automatic Retrieval of Potentially Relevant Knowledge from CYC
Intermediate CYC Ontology File

Figure 3: Mixed-initiative retrieval of relevant ontological knowledge

CYC terms, called seed. This seed represents the input to an automatic retrieval process that extracts from CYC all the terms that are related to those in the given seed. The automatic retrieval process is based on a breath-first search in a graph where the nodes are the terms and the edges are the CYC axioms that connect them. The result of this process is the transitive closure of the knowledge related to the seed, or a subset of it (the user has the possibility to specify a bound on the depth of the search or to stop the process at any time). The output of this process is a subset of the CYC ontology that is potentially relevant for the Disciple ontology to be developed.

In the second phase of the ontology import process, the retrieved CYC ontology is automatically translated into an intermediate Disciple ontology by a general rule-based translation engine that uses a CYC-Disciple rule translation library. Additional rule translation libraries need to be defined for each type of knowledge repository (e.g. for an OKBC-compliant knowledge server, for older Disciple repositories, etc.). Although we are currently using hand-written libraries of rules, we plan to use Disciple to learn general translation rules from the specific examples.

One important issue in ontology translation is the relative expressive power of the languages between which the translation takes place (see [Corcho and Gomez-Perez, 2000] for a comparison of the expressiveness of several ontology specification languages). As mentioned above, the representation of the Disciple object ontology is based on the OKBC knowledge model and is usually less powerful than the representations of the knowledge servers from which we need to import knowledge. On the other hand, the purpose of ontology import in Disciple is not to import the entire knowledge from the knowledge repository, but only the relevant knowledge that can be represented in the Disciple object ontology. This is because the primary purpose of the Disciple object ontology is to serve as a generalization hierarchy for learning of problem solving rules. Most of the representational and inferential power of Disciple does not come from the object ontology, but from the learned rules which we consider to be much more domain-specific and even expert-specific, and therefore less reusable and less likely to require importing.

The result of this translation process is an intermediate Disciple ontology which is the input for the third phase of the ontology import process. This intermediate ontology contains all the ontological knowledge retrieved from CYC or another knowledge repository. This is generally a very large ontology and only a relatively small part of it is likely to be useful for the final Disciple ontology to be built. The actual import is therefore taking place from this intermediate ontology. However, this is a Disciple ontology, and can be browsed using the Disciple tools. Therefore, the subject matter expert and the Disciple agent can collaborate to effectively import from it into the agent's ontology the object concepts that are considered useful.

An important feature of the Disciple approach is that most of the ontology import task can be done by the subject matter expert and the agent, with only limited assistance from the knowledge engineer. Also, the subject matter expert does not need to deal with the representation or tools of the external knowledge repository, but only with the representation of the system to be built (which, in this case, is Disciple). Finally, to be able to import knowledge from a new knowledge repository, the knowledge engineer would only need to implement a retrieval interface like the one in Figure 3, and to define rules to translate knowledge from the external repository to Disciple. These components are not very complicated. All the other components needed are independent of the external knowledge server.

5 Ontology Development

The imported ontology will generally need to be further extended and maintained. Disciple-RKF contains a set of browsers and viewers for easy navigation and visualization of the ontology. They include hierarchical browsers that allow the subject matter expert to navigate the ontology along the generalization relationships between the object concepts or the object features. There is also an association browser that allows the visualization of the object ontology as a network where the objects are the nodes and their relationships are the links. Navigating through this network is done by simply clicking on a object which becomes the center of the screen.

While visualizing and navigating the ontology are relatively simple tasks for a subject matter expert, modifying the ontology is a very complex task. For instance, let us consider the case where the user wishes to delete the subclass-of (is-a) relation between the concept B and the concept A (see Figure 4). This operation will not generate any inconsistency related to either A or B, but will generate an inconsistency for the sub-concept C of B. The concept C has the feature f, and this feature has the domain A (the domain of a feature represents the set of all objects that may have that feature). After removing B as sub-concept of A, the concept C will no longer be in the domain A of f, and therefore C may no longer have the feature f. As this example illustrates, a modification in one part of the ontology may generate subtle inconsistencies in other parts, and this makes ontology modification a very complex process.

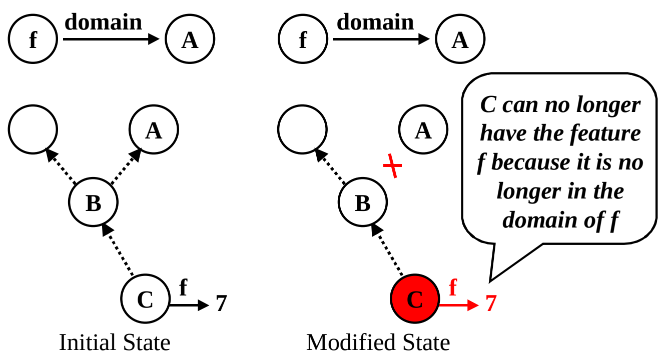

domain
f
A
B
C
f
7
Initial State
domain
C can no longer have the feature f because it is no longer in the domain of f
Modified State

Figure 4: Inconsistency generated by a modification in the object ontology

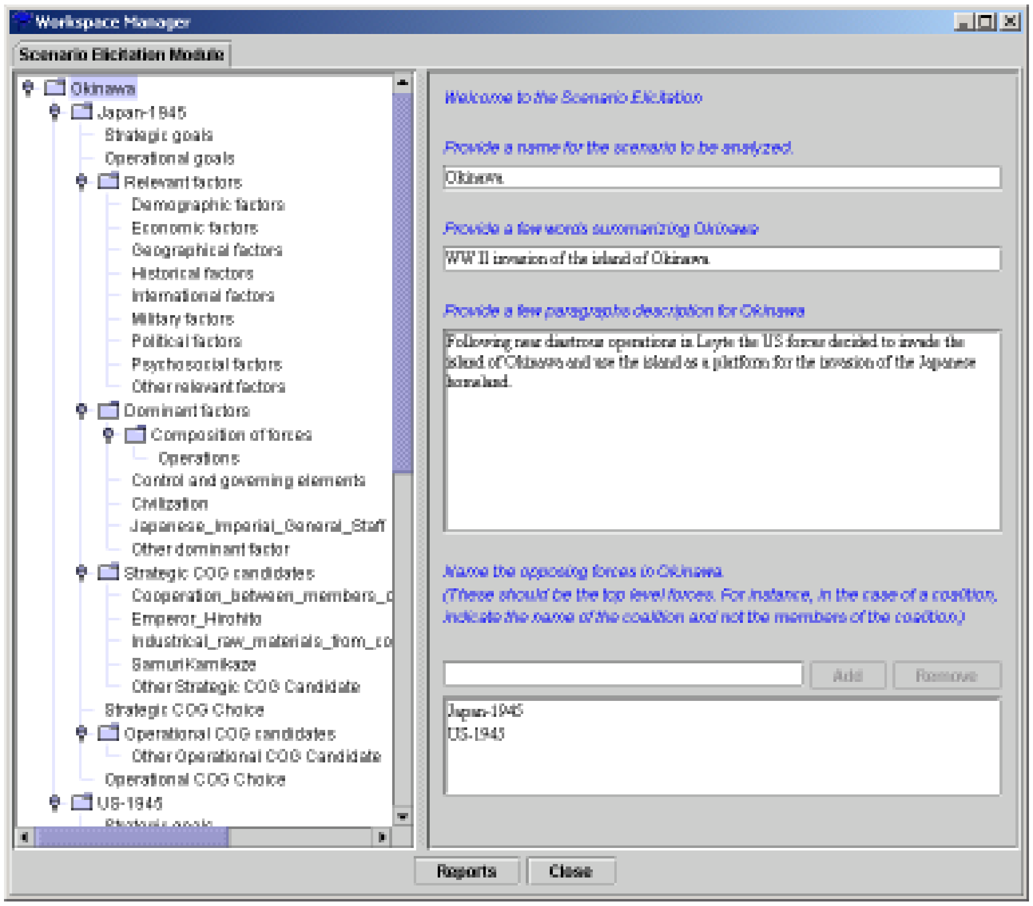
In principle, there are two different approaches to ontology modification. The first one is to allow the user to introduce inconsistencies in the knowledge base and then to correct them. This approach is used in the Chimaera system [McGuinness *et al.*, 2000]. In this approach the modification of the knowledge base becomes an easy process. However, removing the inconsistencies is a very difficult process, which we think to be well beyond the capabilities that can be expected from a subject matter expert. Therefore we did not adopt this approach in Disciple. Instead, we adopted an approach where specialized ontology management assistants (which implement knowledge engineering methods and operations) guide and support the user in modifying the knowledge base such that the ontology will always be in a consistent state. There are assistants to create object concepts and features, to change the superconcepts of an object concept, to specify the value of a feature, to delete objects and features, to rename or copy them, and others. To implement these assistants we are developing a hierarchy of errors and warnings, as well as corresponding error correction methods.

The assistants operate according to the following scenario. The user formulates a goal, for instance to delete a given object concept. Then the corresponding assistant, which in this case is the delete assistant, analyses the knowledge base to determine all the implications of the operation intended by the user. It then notifies the user on the consequences of his or her planned action. After that a mixed initiative process is started to achieve the user's goal without introducing inconsistencies in the knowledge base. The assistant will propose specific knowledge management operations and the user may select the operations and guide the assistant to perform them.

6 The Input Ontology

After the object ontology is created, the agent can be trained to solve problems, as will be briefly presented in section 7. For this, one has to represent a problem in the agent's knowledge base. The part of the object ontology that is used to describe an input problem represents the input ontology. Let us consider, for instance, the most recent application of Disciple: identification of strategic center of gravity candidates in military conflicts. In 1832 Clausewitz introduced the concept of a center of gravity of a force as "the hub of all power and movement, on which everything depends" [Gilles *et al.*, 1996]. In this domain, an input problem is a description of a conflict scenario, such as the World War II planned invasion of Okinawa by the Allied forces in 1945. This includes the specification of the goals of the opposing forces, of the relevant factors (such as economic and geographical factors), and of the dominant factors (such as the composition of forces, the controlling and governing elements, and the type of civilization). Only after all this information is provided can Disciple reason about the potential centers of gravity of the opposing forces. From an ontology point of view, specifying an input problem (a scenario) consists of defining instances of the concepts from the input ontology, together with their features. This is not a trivial task for a subject matter expert. Therefore specialized elicitation forms are used to facilitate it, such as those from the Protégé system. For Disciple, we have developed a Scenario Elicitation module that allows the subject matter expert to create and update a scenario using a simple interface, which is illustrated in figure 5. The left hand side of the interface is a tree of titles and subtitles, similar to a table of contents. Each title (or node) corresponds to a certain type of information. When the expert clicks on such a node, Disciple requests relevant information about that node in the right hand side of the screen. If the expert has previously provided this information he can review or update it. The subject matter expert can go to any entry in this table of contents, to provide or update the information corresponding to that entry. Some information provided by the expert may lead to the creation of additional nodes in the left hand side of the interface. For instance, when the expert defined Japan-1943 and US-1943 as opposing forces, several nodes have been introduced in the left hand side of the interface.

The main idea of the implementation of the scenario elicitation module is to associate elicitation scripts with the concepts from the input ontology. The script associated with a concept plays multiple roles: it specifies how an instance of that concept is created; what features of the instance need to be elicited; how the dialog with the user takes place, and what graphical components are used in this dialog. In the current version of Disciple these scripts have to be developed by a knowledge engineer after the input ontology has been created. A single concept from the ontology is also marked as the starting concept for the scenario elicitation. In the example from figure 5, the starting concept is "Scenario". The right part of figure 5 shows the dialog between the system and the user. Once the user introduced the name of the scenario ("Okinawa"), the system created an instance of the "Scenario" concept. Then the script to elicit the features of the Okinawa scenario was activated. To elicit a specific feature, Disciple also uses the information from the ontology about that feature, such as the possible values and its cardinality. When the subject matter expert specifies a value of a feature that is an instance of some other concept, the script of that concept is activated and a new


Workspace Manager

Scenario Elicitation Module

- Okinawa
  - Japan-1945
    - Strategic goals
    - Operational goals
    - Relevant factors
      - Demographic factors
      - Economic factors
      - Geographical factors
      - Historical factors
      - International factors
      - Military factors
      - Political factors
      - Psychosocial factors
      - Other relevant factors
    - Dominant factors
      - Composition of forces
        - Operations
      - Control and governing elements
      - Civilization
      - Japanese_Imperial_General_Staff
      - Other dominant factor
    - Strategic COG candidates
      - Cooperation_between_members_c
      - Emperor_Hirohito
      - Industrical_raw_materials_from_co
      - Samurai Kamikaze
      - Other Strategic COG Candidate
    - Strategic COG Choice
    - Operational COG candidates
      - Other Operational COG Candidate
    - Operational COG Choice
  - US-1945
    - Strategic goals

Welcome to the Scenario Elicitation

Provide a name for the scenario to be analyzed.

Okinawa

Provide a few words summarizing Okinawa

WW II invasion of the island of Okinawa.

Provide a few paragraphs description for Okinawa

Following near disastrous operations in Leyte the US forces decided to invade the island of Okinawa and use the island as a platform for the invasion of the Japanese homeland.

Name the opposing forces in Okinawa.
(These should be the top level forces. For instance, in the case of a coalition, indicate the name of the coalition and not the members of the coalition.)

Japan-1945
US-1945

Figure 5: An interface for scenario elicitation

entry is added to the table of contents for the features of that instance. Figure 6 shows a fragment of the elicitation script for the concept "Scenario." This script requires the user to specify the opposing forces as illustrated at the bottom right of figure 5. Figure 7 shows the corresponding instances and relationships that have been introduced in the ontology.

Elicitation Script for the instances of the concept: *Scenario*
. . .
Property: *detailed-name*
  Prompt: "Provide a few words summarizing " <*current-instance*>
  Control-type: ***single-line***
. . .
Property: *has_as_opposing_force*
  Prompt: "Name the opposing forces in "<*current-instance*>
  Control-type: ***multiple-names***
  Other ontology actions: <*property-value*> instance-of Opposing_force
. . .

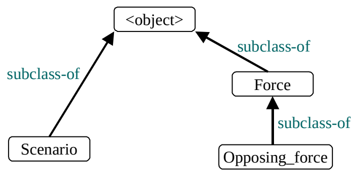


Figure 6: Fragment of the elicitation script for "Scenario"

7 Agent Teaching and Rule Learning

After an object ontology has been developed, the subject matter expert starts teaching the agent how to solve problems through successive reductions and compositions. This process is explained in [Tecuci *et al.*, 2000b]. Here we only briefly review it in order to have a complete description of the Disciple methodology. In essence, the subject matter expert starts from the domain models that

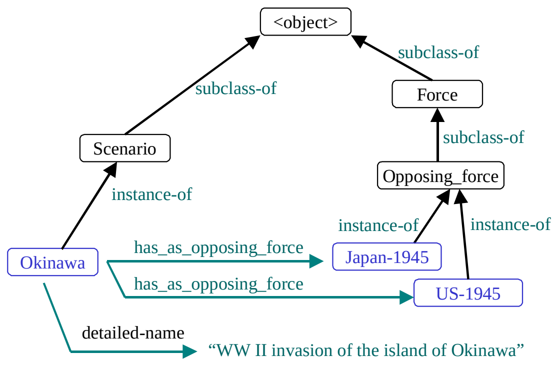


Figure 7: The result of the elicitation script from figure 6

have been previously prepared in collaboration with the knowledge engineer, as presented in section 3. The left hand side of Figure 1 shows an example of the task-reduction modeling of the problem solving process. Each abstract task reduction step (consisting of a task, a question, an answer, and a subtask) is expanded into a training example for the Disciple agent, as illustrated in Figure 8. Each task is now represented by a name phrase and a set of feature-value pairs. The answers are also made more specific. From each such task reduction example the agent learns a general task reduction rule that will allow it to apply a similar task reduction operation in future problem-solving situations. For instance, the rule learned from the second task reduction step in figure 8 is represented in figure 9. The process of learning such a general task reduction rule is a mixed-initiative one. First the subject matter expert and the agent collaborate in finding a formal justification of why the current task reduction is correct. Then, based on the found justification, the agent generalizes the example into a task reduction rule. As shown in figure 9, the learned rule is an IF-THEN structure with two applicability conditions, a plausible lower bound condition and a plausible upper bound condition. These two conditions represent a plausible version space for the exact applicability condition of the rule. Through further learning, the two conditions converge toward one another and toward this exact condition. An important thing to notice is that the rule's conditions are expressed in term of the concepts and the features from the object ontology. In general, the rule's conditions could be much more complex expressions than the ones illustrated in figure 9.

8 Ontology Learning

Ontology learning is becoming an important research issue [Staab *et al.* 2000]. It also plays an important role in the Disciple agent development methodology. During the process of defining or explaining specific task reductions or compositions, when training the agent, the subject matter expert may need to refer to objects or object features that are not yet part of the ontology. From these specific instances Disciple-RKF will learn general ontological elements. For example, the subject matter expert may point to a specific feature of an object, as being responsible for the failure of a certain task reduction step. In such a case the agent will learn a general object feature definition from that specific feature. Any object feature definition specifies a domain (a concept that represents the set of objects that could have that feature) and a range (another concept that represents the set of possible values of that feature).

Disciple-RKF generates a plausible version space for the domain concept, and another one for the range concept. These version spaces are similar to the plausible version space condition of the rule shown in figure 9. After the versions spaces are generated, Disciple initiates a feature refinement experimentation session with the goal of reducing the plausible version spaces of the feature's domain and range.

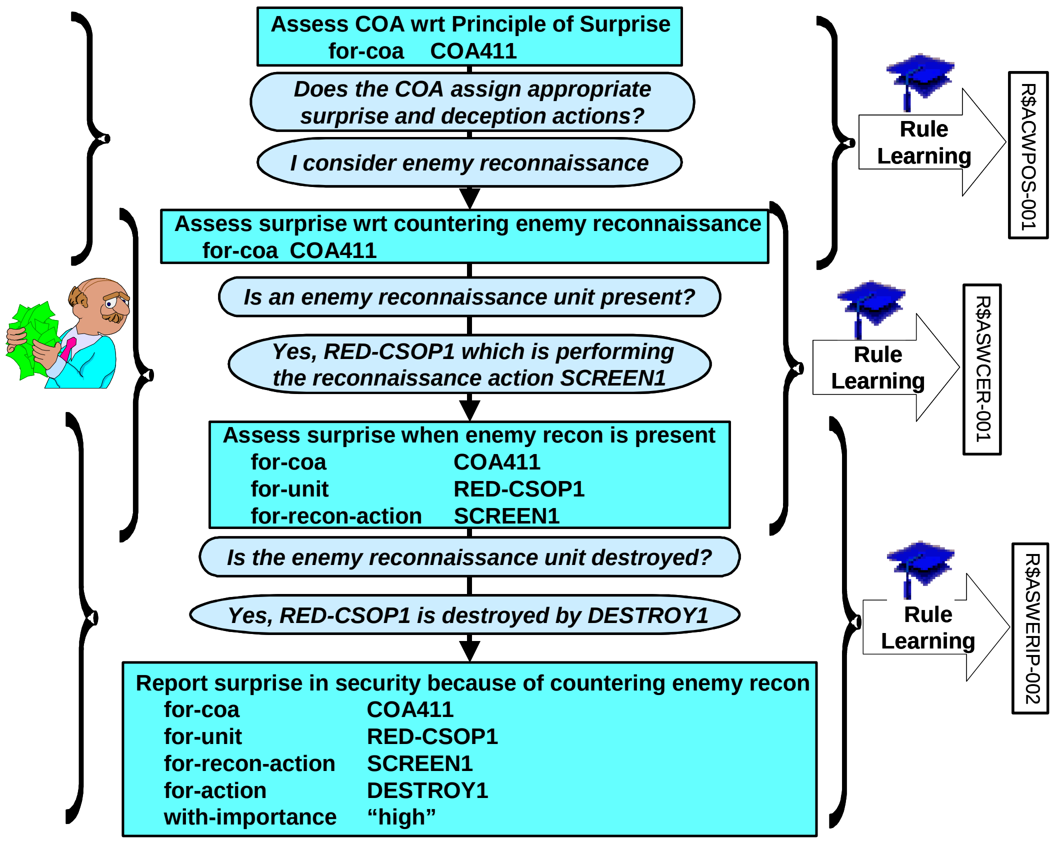
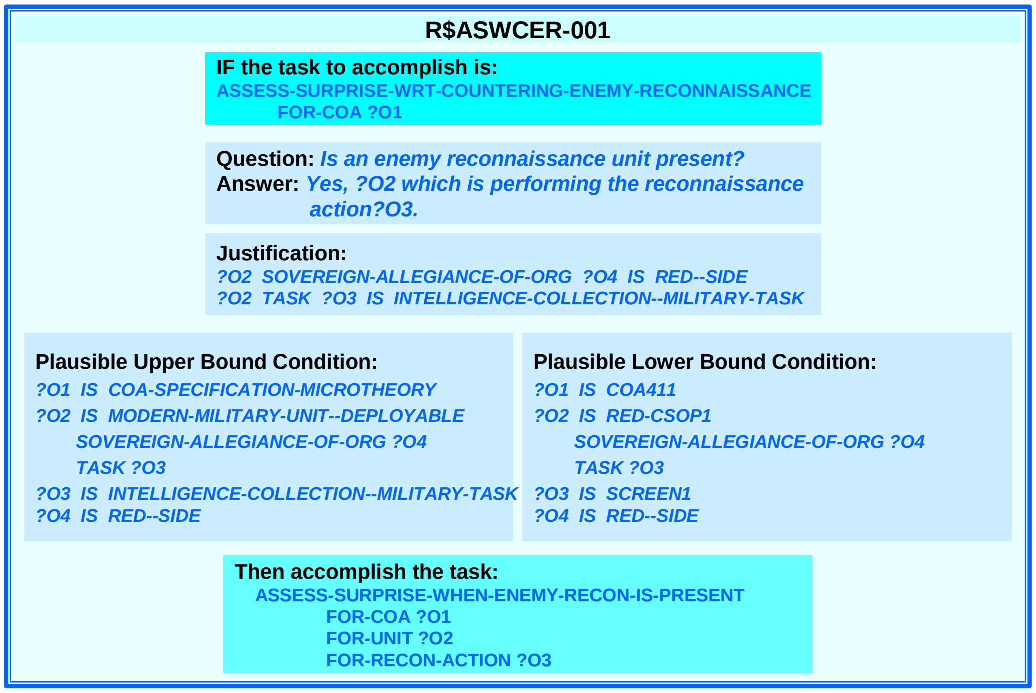

Assess COA wrt Principle of Surprise
for-coa COA411
Does the COA assign appropriate surprise and deception actions?
I consider enemy reconnaissance
Assess surprise wrt countering enemy reconnaissance
for-coa COA411
Is an enemy reconnaissance unit present?
Yes, RED-CSOP1 which is performing the reconnaissance action SCREEN1
Assess surprise when enemy recon is present
for-coa COA411
for-unit RED-CSOP1
for-recon-action SCREEN1
Is the enemy reconnaissance unit destroyed?
Yes, RED-CSOP1 is destroyed by DESTROY1
Report surprise in security because of countering enemy recon
for-coa COA411
for-unit RED-CSOP1
for-recon-action SCREEN1
for-action DESTROY1
with-importance "high"
Rule Learning
R$ACWPOS-001
Rule Learning
R$ASWCER-001
Rule Learning
R$ASWERIP-002

Figure 8: Sample teaching and learning scenario



R$ASWCER-001

IF the task to accomplish is:
ASSESS-SURPRISE-WRT-COUNTERING-ENEMY-RECONNAISSANCE
FOR-COA ?O1

Question: *Is an enemy reconnaissance unit present?*
Answer: *Yes, ?O2 which is performing the reconnaissance action?O3.*

Justification:
?O2 SOVEREIGN-ALLEGIANCE-OF-ORG ?O4 IS RED--SIDE
?O2 TASK ?O3 IS INTELLIGENCE-COLLECTION--MILITARY-TASK

Plausible Upper Bound Condition:
?O1 IS COA-SPECIFICATION-MICROTHEORY
?O2 IS MODERN-MILITARY-UNIT--DEPLOYABLE
SOVEREIGN-ALLEGIANCE-OF-ORG ?O4
TASK ?O3
?O3 IS INTELLIGENCE-COLLECTION--MILITARY-TASK
?O4 IS RED--SIDE

Plausible Lower Bound Condition:
?O1 IS COA411
?O2 IS RED-CSOP1
SOVEREIGN-ALLEGIANCE-OF-ORG ?O4
TASK ?O3
?O3 IS SCREEN1
?O4 IS RED--SIDE

Then accomplish the task:
ASSESS-SURPRISE-WHEN-ENEMY-RECON-IS-PRESENT
FOR-COA ?O1
FOR-UNIT ?O2
FOR-RECON-ACTION ?O3

Figure 9: Sample task reduction rule learned by Disciple

9 Knowledge Base Export

Figure 10 illustrates the synergistic relationship between the Disciple-RKF agent development tool and an external knowledge server, such as CYC. To develop a knowledge-based agent with Disciple-RKF one starts by importing an initial object ontology from the CYC knowledge server, as discussed in section 4. Then the subject matter expert interacts with Disciple, teaching it to solve problems, and thus developing the knowledge base of Disciple to incorporate the expertise of the subject matter expert. After the Disciple knowledge base has been developed, it is exported back into CYC, as a separate CYC microtheory. This is an automatic translation process that does not raise any problems because CYC's knowledge representation is more powerful than that of Disciple-RKF. Then this CYC microtheory can be semantically integrated with the rest of the CYC knowledge repository, by the developers of CYC. This semantic integration is a difficult task, but it is facilitated in this case by the fact that the initial Disciple ontology has been imported from CYC, to begin with.

We have performed a preliminary experiment during which the knowledge base of Disciple corresponding to the Course of Action challenge problem has been automatically translated into a CYC microtheory. Then, using its inference engine, CYC generated the same critiques of a course of action as Disciple. The integration model described above can be adapted for any other knowledge server, with only minor modifications. For instance, in the case of an OKBC knowledge server, only the object ontology of Disciple will be exported because the rules cannot be represented using the OKBC frame-based knowledge model.

10 Experimental Results

Successive versions of the Disciple approach and other competing knowledge base development approaches have been evaluated in several intensive studies requiring the rapid development and maintenance of knowledge bases for solving the workaround challenge problem (consisting of planning the repair of damaged bridges and roads [Jones, 1998]), and the COA challenge problem (consisting of generating critiques of military courses of action [Jones, 1999]). These evaluations were performed by Alphatech, as part of the DARPA's HPKB program, and involved, in addition to Disciple, the following teams and approaches: 1) Teknowledge and Cycorp that used the CYC system [Lenat, 1995].

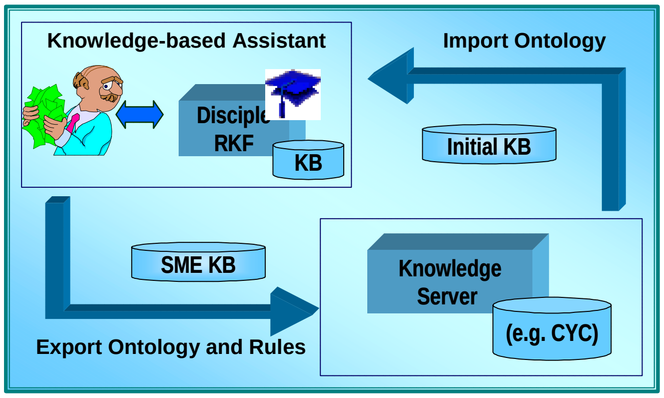


Figure 10: The synergy between Disciple-RKF and the knowledge servers

They achieved rapid knowledge base development through extensive re-use of CYC's carefully developed ontology, wide-ranging common-sense knowledge, and general inferential capabilities. 2) The EXPECT group from USC-ISI. This group developed knowledge bases with wide problem coverage and expert-level performance, using the knowledge acquisition tools of EXPECT that assist a knowledge engineer in debugging and refining a knowledge base [Kim and Gil, 1999]. 3) The Loom/PowerLoom group from USC-ISI that used novel case-based reasoning techniques for the COA challenge problem, in conjunction with the PowerLoom [MacGregor, 1999] representation system and an imported ontology. 4) The AIAI group from the University of Edinburgh that developed a high performance knowledge base for the workaround challenge problem by designing a planning ontology in CYC.

In these experiments all the approaches demonstrated very good results and relative technology strengths. However, the Disciple approach has achieved the highest rates of knowledge acquisition and the best problem solving performance, while the generated solutions and justifications where judged as being very intelligible.

The first evaluation concerned the workaround challenge problem and lasted for 17 days. At the beginning of the evaluation Disciple had an incomplete knowledge base consisting of 723 object concepts, 100 tasks, and 121 task reduction rules. Out of the 723 concepts 126 were imported from LOOM (an OKBC compliant knowledge server). They included elements of the military unit ontology, as well as various characteristics of military equipment (such as their tracked and wheeled military load classes). The extent of knowledge import was more limited than it could have been because the LOOM's ontology was developed at the same time as that of Disciple, and we had to define concepts that have later been also defined in LOOM and could have been imported. In any case, importing those concepts proved to be very helpful, and has demonstrated the ability to reuse previously developed knowledge.During the 17 days of the evaluation, the knowledge base of Disciple was increased with 147 object concepts, 104 tasks, and 87 complex task reduction rules. The performance of the developed knowledge-based agent was judged by the evaluators as being at the level of a human expert.

The second evaluation concerned the COA challenge problem and lasted 8 days. In this case the initial ontology was imported from CYC. During the evaluation period the knowledge base of Disciple was increased by 46%, which represents an even higher daily rate of knowledge acquisition than in the first experiment. Also, in addition to generating most of the critiques expected by the evaluators, Disciple generated many new critiques. The final knowledge base contained 801 concepts, 444 object and task features, 360 tasks and 342 rules. Also, each input problem (the description of a course of action) was represented with around 1500 facts. Currently Disciple is further developed and evaluated at the US Army War College, being used by subject matter experts to develop knowledge bases for the identification of strategic center of gravity candidates.

11 Conclusions and Future Research

We have presented an approach to rapid development of knowledge-based agents by subject matter experts that is based on ontology reuse and development.

In addition to the further development of the methods presented in this paper, future research on ontologies and information sharing will consist in extending the Disciple approach (Tecuci, 1998) and the supporting tools to allow several experts to collaborate in building different parts of a larger knowledge base. In particular, we plan to develop a distributed architecture for collaborative knowledge base development, as shown in Figure 11.

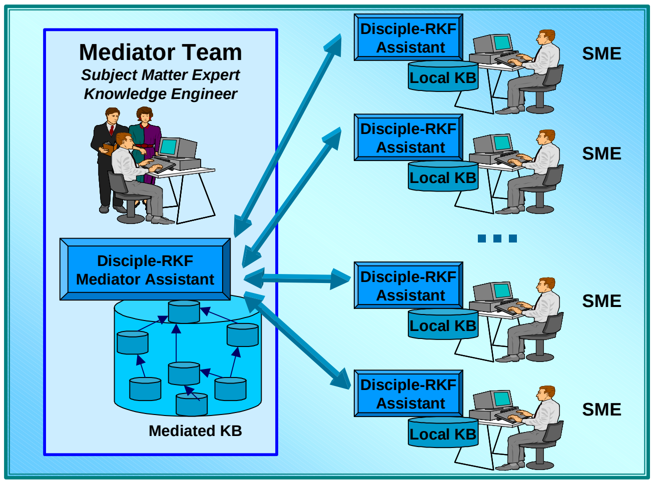


Figure 11: Collaborative knowledge base development

The right hand side of Figure 11 represents a team of subject matter experts that collaborate to rapidly build an integrated knowledge base. Each individual subject matter expert works with a personal Disciple-RKF agent to build a part of the integrated knowledge base. These separately developed knowledge bases are periodically integrated into a single knowledge base by the mediator team that includes a knowledge engineer, a subject matter expert, and a Disciple agent specialized in knowledge integration. The mediator team not only integrates the knowledge bases, but also mediates the collaboration between all the subject matter experts.

Several features of the proposed approach facilitate collaborative knowledge base development. First, the knowledge base is structured into an object ontology that defines the terms of the representation language, and set of task reduction rules that are expressed using these terms. As a consequence, the subject matter experts have to agree on the shared object ontology but they can develop the rules independently.

Second, the knowledge base to be built is divided into parts that are as independent as possible, with each subject matter expert responsible for the development of a different part. The mediator team coordinates the partitioning and the integration of the knowledge base, and facilitates a consensus among the subject matter experts concerning the developed knowledge that is to be shared.

Third, the integrated knowledge base consists of a hierarchy of component knowledge bases that are each internally consistent, but may contain portions that supersede or contradict portions from other knowledge bases. This corresponds to the fact that the knowledge model of a subject matter expert is internally consistent but it may contain knowledge that contradicts aspects of the knowledge model of another subject matter expert. This knowledge base organization not only facilitates knowledge acquisition from multiple subject matter experts, but also leads to a knowledge base that can provide solutions to problems from different points of view.

Acknowledgements

This research has been performed in the GMU Learning Agents Laboratory and was sponsored by the Defense Advanced Research Projects Agency (DARPA), Air Force Research Laboratory, Air Force Material Command, USAF, under agreement number F30602-00-2-0546, by the Air Force Office of Scientific Research (AFOSR) under grant no. F49620-00-1-0072, and by the US Army. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Dorin Marcu, Michael Bowman, Cristina Cascaval, and other members of the LALAB have contributed to successive versions of Disciple. In addition to Michael Bowman, Tony Lopez and Jim Donlon, from the Center for Strategic Leadership of the US Army War College, have contributed to the application of Disciple to the center of gravity challenge problem. The anonymous reviewers of this paper provided insightful comments that helped us to improve it.

References

[Boicu *et al.*, 2000] Mihai Boicu, Gheorghe Tecuci, Dorin Marcu, Michael Bowman, Ping Shyr, Florin Ciucu, and Cristian Levcovici. Disciple-COA: From Agent Programming to Agent Teaching. In *Proceedings of the Seventeenth International Conference on Machine Learning*, Stanford, CA, Morgan Kaufmann, 2000.

[Buchanan and Wilkins, 1993] Bruce G. Buchanan and David C. Wilkins (editors). *Readings in Knowledge Acquisition and Learning: Automating the Construction and Improvement of Expert Systems.* Morgan Kaufmann, San Mateo, CA., 1993

[Burke, 1999] Murray Burke. *Rapid Knowledge Formation (RKF) Program Description,* http://dtsn.darpa.mil/iso/index2.asp?mode=9

[Chaudhri *et al.* 1998] Vinay K. Chaudhri, Adam Farquhar, Richard Fikes, Daniel P. Park, and James P. Rice. OKBC: A Programmatic Foundation for Knowledge Base Interoperability. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, Menlo Park, CA: AAAI Press, pages 600 – 607, 1998.

[Cohen *et al.*, 1998] Paul Cohen, Robert Schrag, Eric Jones, Adam Pease, Albert Lin, Barbara Starr, David Gunning and Murray Burke. The DARPA High-Performance Knowledge Bases Project, *AI Magazine*, 19(4), 25-49, 1998.

[Corcho and Gomez-Perez, 200] Oscar Corcho and Asuncion Gomez-Perez. Evaluating Knowledge Representation and Reasoning Capabilities of Ontology Specification Languages. In *Proceedings of the ECAI 2000 Workshop on Application of Ontologies and Problem-Solving Methods, Berlin, 2000*.

[Farquhar *et al.,* 1996] Adam Farquhar, Richard Fikes, and James Rice. The Ontolingua Server: a Tool for Collaborative Ontology Construction. In *Proceedings of the Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, Alberta, Canada, 1996.

[Fridman *et al.,* 2000] Natalya Fridman Noy, Ray W. Fergerson, Mark A. Musen. The Knowledge Model of Protégé-2000: Combining Interoperability and Flexibility. In *Proceedings of the European Knowledge Acquisition Workshop*, pages 17-32, 2000.

[Gilles *et al.*, 1996] MAJ Phillip Kevin Giles, CPT Thomas P. Galvin. *Center of Gravity: Determination, Analysis, and Application*, U.S. Army War College, Carlisle Barracks, PA, 1996.

[Jones, 1998]. Eric Jones. *HPKB Year1 End-to-End Battle-space Challenge Problem Specification*. Burlington, 1998.

[Jones, 1999]. Eric Jones. *HPKB Course of Action Challenge Problem Specification*. Alphatech, Burlington, 1998.

[Kim and Gil, 1999]. Jihie Kim and Yolanda Gil. Deriving Expectations to Guide Knowledge Base Creation. In *Proc. of the Sixteenth National Conference on Artificial Intelligence*,235-241, Menlo Park, CA: AAAI Press, 1999.

[Lenat, 1995] Douglas B. Lenat. CYC: A Large-scale investment in knowledge infrastructure. In *Communications of the ACM* 38(11): 33-38, 1995.

[MacGregor, 1999] Robert MacGregor. *Retrospective on LOOM.* Available online: http://www.isi.edu/isd/LOOM/papers/macgregor/Loom_Retrospective.html, 1999.

[McGuinness *et al.*, 2000] Deborah L. McGuinness, Richard Fikes, James Rice, Steve Wilder. The Chimaera Ontology Environment. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, Menlo Park, CA, AAAI Press, pages 1123-1124, 2000.

[Staab *et al.* 2000] Steffen Staab, Alexander Maedche, Claire Nedellec, and Peter Wiemer-Hastings (eds.) *Proceedings of the First Workshop on Ontology Learning OL'2000*, Berlin, Germany, August 25, 2000.

[Tecuci, 1998] Gheorghe Tecuci. *Building Intelligent Agents: An Apprenticeship Multistrategy Learning Theory, Methodology, Tool and Case Studies*. London, England: Academic Press, 1998.

[Tecuci et al. 2000a] Gheorghe Tecuci, Mihai Boicu, Kathryn Wright, Seok-Won Lee, Dorin Marcu, and Michael Bowman. A Tutoring Based Approach to the Development of Intelligent Agents. In Teodorescu, H.N., Mlynek, D., Kandel, A. and Zimmermann, H.J. (editors). *Intelligent Systems and Interfaces,* Kluwer Academic Press. 2000.

[Tecuci et al. 2000b] Gheorghe Tecuci, Mihai Boicu, Michael Bowman, Dorin Marcu, Ping Shyr, and Cristina Cascaval. 2000 "An Experiment in Agent Teaching by Subject Matter Experts," *International Journal of Human-Computer Studies* 53: 583-610.