In Proceedings of the Seventeenth International Conference on Machine Learning (ICML), Stanford, California, 2000
Morgan Kaufman, 2000.

Disciple-COA: From Agent Programming to Agent Teaching

Mihai Boicu MBOICU@GMU.EDU
Gheorghe Tecuci TECUCI@GMU.EDU
Dorin Marcu DMARCU@GMU.EDU
Michael Bowman MBOWMAN3@OSF1.GMU.EDU
Ping Shyr PING.SHYR@NASD.COM
Florin Ciucu FCIUCU@GMU.EDU
Cristian Levcovici CLEVCOVI@GMU.EDU
Learning Agents Laboratory, Department of Computer Science, George Mason University, Fairfax, VA 22030 USA

Abstract

This paper presents Disciple-COA, the most recent learning agent shell developed in the Disciple framework that aims at changing the way an intelligent agent is built: from "being programmed" by a knowledge engineer, to "being taught" by a domain expert. Disciple-COA can collaborate with the expert to develop its knowledge base consisting of a frame-based ontology that defines the terms from the application domain, and a set of plausible version space rules expressed with these terms. Its central component is a plausible reasoner that can distinguish between four types of problem solving situations: routine, innovative, inventive and creative. This ability guides the interactions with the expert during which the agent learns general rules from specific examples, by integrating a wide range of knowledge acquisition and machine learning strategies, including apprenticeship learning, empirical inductive learning from examples and explanations, and analogical learning. Disciple-COA was developed in the DARPA's High Performance Knowledge Bases program to solve the challenge problem of critiquing military courses of action that were developed as hasty candidate plans for ground combat operations. We present the course of action challenge problem, the process of teaching Disciple-COA to solve it, and the results of DARPA's evaluation in which Disciple-COA demonstrated the best knowledge acquisition rate and problem solving performance. We also present a separate knowledge acquisition experiment conducted at the Battle Command Battle Lab where experts with no prior knowledge engineering experience succeeded to rapidly teach Disciple-COA to correctly critique courses of action.

1. Introduction

The long term objective of the research done in the Learning Agents Laboratory of George Mason University is to develop apprenticeship multistrategy learning methods and tools that will allow users with little or no knowledge engineering experience to easily build, teach and maintain knowledge-based agents. The vision for this research is to change the way an intelligent agent is built, from "being programmed" by a knowledge engineer, to "being taught" by a subject matter expert.

The approach we are investigating, called Disciple, relies on developing a very capable learning and reasoning agent that can collaborate with a domain expert to develop its knowledge base consisting of an ontology that defines the terms from the application domain, and a set of general problem solving rules expressed with these terms. The process of developing the knowledge base of the agent relies on importing ontological knowledge from existing repositories of knowledge, and teaching the agent how to perform various tasks, in a way that resembles how the domain expert would teach an apprentice while solving problems in cooperation.

Over the years, the Disciple approach has been developed and scaled-up continuously, more recently as part of the 1997-1999 High Performance Knowledge Bases (HPKB) program supported by DARPA and AFOSR (Cohen et al., 1998). The organizations participating in HPKB were given the challenge of rapidly developing and updating knowledge-based systems for solving specially designed challenge problems. The aim was to test the claim that, with the latest AI technology, large knowledge bases can be built and updated quickly and efficiently.

One challenge problem for the first part of the HPKB program was to build a knowledge-based workaround agent that is able to plan how a convoy of military vehicles can "work around" (i.e. circumvent or overcome) obstacles in their path, such as damaged bridges or

minefields. To solve this challenge problem we developed the Disciple-Workaround learning agent, demonstrating that a knowledge engineer can rapidly teach Disciple using military engineering manuals and sample solutions provided by an expert. During the 17 days of DARPA's evaluation, the KB of Disciple was increased by 72% (from the equivalent of 5,920 simple axioms to 10,162 simple axioms) with almost no decrease in performance. The Disciple agent also achieved the best scores among all the teams that participated in the workaround challenge problem, and was selected to represent the HPKB program at EFX'98, the Air Force's show case of the most promising technologies.

One challenge problem for the second part of the HPKB program was to build a critiquing agent that can evaluate military Courses of Action (COA) that were developed as hasty candidate plans for ground combat operations. To solve this challenge problem we developed the Disciple-COA learning agent, and in the process we achieved two significant milestones with the Disciple approach:

- For the first time we developed the knowledge base around an ontology created by another group (Teknowledge and Cycorp), demonstrating both the feasibility of knowledge reuse with the Disciple approach, and the generality of the Disciple rule learning and refinement methods. Moreover, the Disciple-COA agent was taught even more rapidly than the Disciple-workaround agent, and has again demonstrated a significantly higher performance than the other developed critiquers.
- For the first time we conducted a knowledge acquisition experiment where four domain experts with no prior knowledge engineering experience received very limited training in the teaching of Disciple-COA and then each succeeded to significantly extend its knowledge base, receiving no or only very limited support from a knowledge engineer.

In this paper we present the current status of the Disciple approach which has achieved a significant level of maturity and holds the promise of realizing the vision stated above. We will first introduce the COA critiquing challenge problem. Then we will present the architecture, knowledge representation, problem solving and learning methods of the current implementation of the Disciple approach, illustrating them with some examples from the COA challenge problem. A significant part of the paper will be dedicated to the presentation of the evaluations of Disciple-COA, including the knowledge acquisition experiment mentioned above. We will then conclude the paper with a discussion of these results.

2. The COA Challenge Problem

The COA challenge problem consists of rapidly developing a knowledge-based system that receives as input the description of a military course of action and assesses various aspects of the COA, such as its viability, its correctness, and its strengths and weaknesses with respect to the principles of war and the tenets of army operations. The system should also be able to justify the assessments and to propose improvements to the COA, helping a military commander to choose the best COA for accomplishing a certain mission. The input to the COA system is a COA sketch and a COA statement. The COA sketch (like the one for COA421 shown in Figure 1) is a graphical representation of the terrain, locations, compositions and missions of the friendly and enemy units. The COA statement explains what the units in a course of action will do to accomplish the assigned mission. This is expressed in a restricted but expressive subset of English, such as "BLUE-TASK-FORCE1, a balanced task force, attacks to seize OBJ-PASS, in order to enable the completion of seize OBJ-SLAM by BLUE-ARMOR-BRIGADE1".

Figure 1: The sketch of COA421.

In the HPKB program, the COA challenge problem was solved by building an integrated system composed of several critiquers, each built by a different team, to solve a part of the overall problem. The participating teams were Teknowledge-Cycorp-AIAI, ISI/Expect, ISI/Loom, U.Mass, and GMU Disciple. All the teams had to share an input ontology and to use the same internal representation of the input generated automatically by Teknowledge and AIAI from COA descriptions provided by Alphatech.

We have developed a COA critiquer, called Disciple-COA, that identifies the strengths and the weaknesses of a course of action with respect to the principles of war and the tenets of army operations. There are nine principles of war: objective, offensive, mass, economy of force, maneuver, unity of command, security, surprise, and simplicity. They provide general guidance for the conduct

"There is a weakness in COA421 with respect to the Principle of Security because there are no actions taken to destroy RED-CSOP1 which is an enemy unit assigned to collect intelligence and protect against surprise. The COA fails to call for aggressive security/counter-reconnaissance actions, destroying enemy intelligence collection units and activities."

Figure 2: A solution generated by Disciple-COA.

of war at the strategic, operational and tactical levels. The tenets of army operations describe the characteristics of successful operations. They are: initiative, agility, depth, synchronization and versatility. Figure 2, for instance, shows a weakness identified by Disciple-COA in COA421.

3. The Current Version of Disciple

The architecture of the current version of Disciple, called Disciple-COA, is represented in Figure 3. At a general level, it is organized into three main components:

- the intelligent user interface component that allows the experts to communicate with Disciple in a manner as close as possible to the way they communicate in their environment,
- the multistrategy learning and problem solving component which is responsible for knowledge formation and problem solving, and contains specific modules for rule learning and refinement, as well as cooperative and autonomous problem solving,
- the knowledge management component that contains modules for knowledge base management, import of ontologies, and knowledge translation between CYC and Disciple.

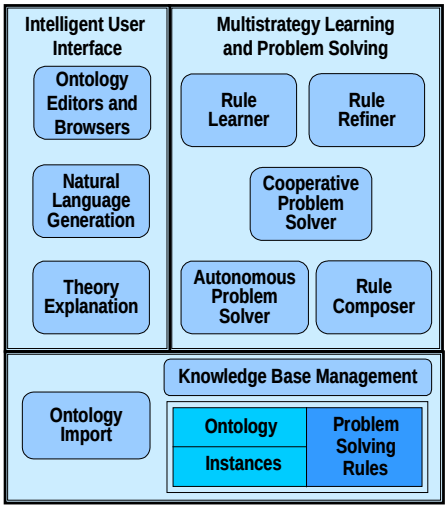

Figure 3: The architecture of Disciple-COA.

The process of building the Disciple-COA critiquer included two main phases, the development of the domain ontology and the teaching of the agent how to critique COAs.

One significant challenge was that each critiquer had to use the same domain ontology that was built by the Teknowledge-Cycorp team for their critiquer developed with the CYC system (Lenat, 1995). Consequently, the ontology of Disciple-COA was created by importing the ontology from CYC and by further extending it with other necessary elements. Disciple's ontology includes hierarchies of objects, features and tasks, represented as frames, according to the OKBC knowledge model

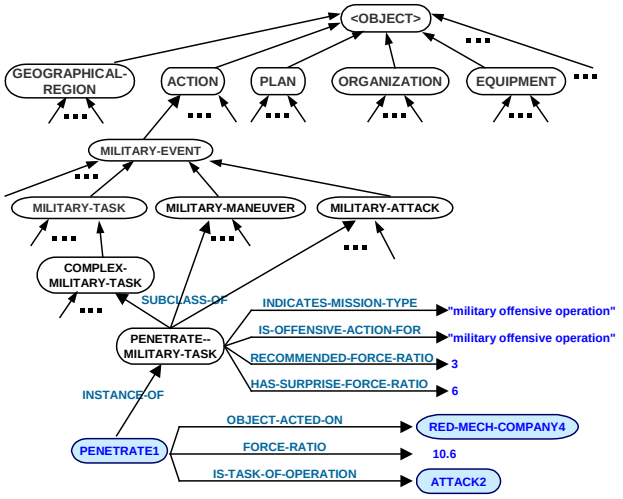

Figure 4: Fragment of the Disciple-COA ontology.

(Chaudhri et al. 1998). A fragment of the object ontology is presented in Figure 4. It shows a part of its upper level and a detail from its lower level. The ontology defines the main concepts from the COA domain, such as organizations, equipment, military tasks, purposes, and geographical concepts. A significant role of the object ontology in Disciple is that of being the generalization hierarchy for learning.

The teaching of Disciple to critique COAs is done by using the cooperative problem solver. The problem solving framework is task reduction, where a task to be accomplished by the agent is successively reduced to simpler tasks until the initial task is reduced to a set of elementary tasks that can be immediately performed. The middle part of Figure 5 shows a sequence of task reduction steps, created jointly by the domain expert and Disciple-COA, for assessing to what extend COA421 conforms to the Principle of Security. In order to perform this assessment, the expert and Disciple need a certain amount of information about COA421 which is obtained by asking a series of questions. These questions and the corresponding answers guide the reduction of assessment tasks into more informed ones, and ultimately into final assessments. For instance, the second question asks whether there is any enemy reconnaissance unit present in COA421. The answer identifies RED-CSOP1 as being such a unit because it is performing the task SCREEN1. Therefore, the task of assessing security of COA421 with respect to countering enemy reconnaissance is now reduced to the better defined task of assessing security when enemy reconnaissance is present. The next question to ask is whether the enemy reconnaissance unit is destroyed or not. In the case of COA421, RED-CSOP1 is not destroyed. Therefore one can conclude that there is a weakness in security because enemy reconnaissance is not countered. As can be seen in Figure 5, tasks are represented by a name and a sequence of feature-value pairs. In addition, the expert may provide their description in natural language. This natural language description (which is not

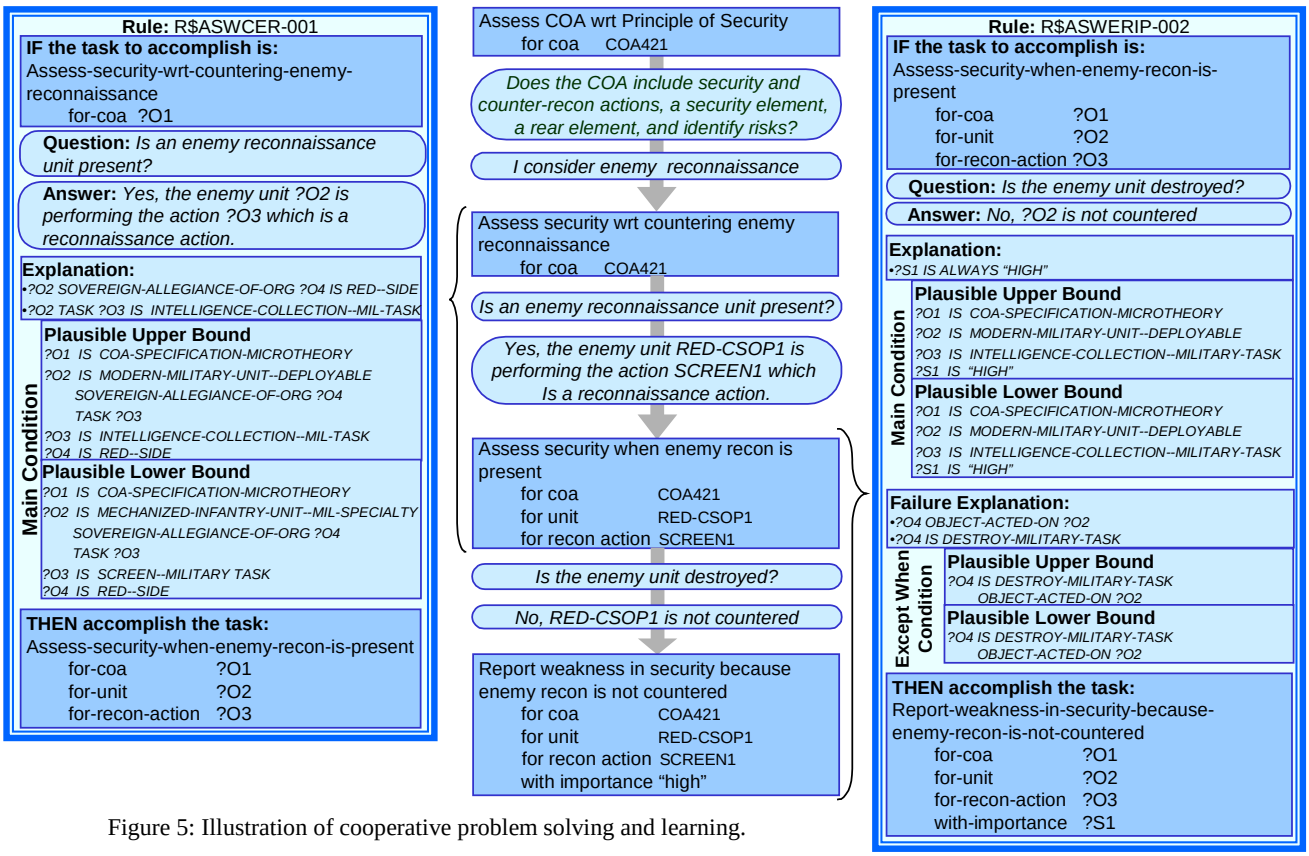

Figure 5: Illustration of cooperative problem solving and learning.

shown in Figure 5 because of the lack of space) is generalized into a natural language pattern, allowing Disciple-COA to generate solutions in natural language, like the one from Figure 2.

The task reduction steps, such as the ones from the middle of Figure 5, are generated through a cooperative problem solving process, where some steps are contributed by Disciple-COA, and some are contributed by the domain expert. Through this process the expert will teach the agent in a natural manner, similar to how the expert would teach a human apprentice. The goal of the agent is to learn from the expert and from its problem solving attempts, developing a knowledge base that would allow it to exhibit the same problem solving performance as the human expert. We call the set of all correct solutions generated with this "final" knowledge base the *Target Solution Space* (see Figure 6). However, part of the *Target Solution Space* is not even included in the *Current Representation Space* of the agent which will have to be extended by introducing new terms in the ontology. The agent-expert interactions are guided by the cooperative problem solver of Disciple-COA that can distinguish between four types of problem solving situations: routine, innovative, inventive and creative (see Figure 6). Initially, when Disciple has few problem solving rules, most problem solving situations are creative because no rules are applicable and the task reductions need to be provided by the expert. From each such creative task reduction step Disciple will learn a new task reduction rule. For instance, from the last task reduction step in Figure 5 (consisting of a task, a question, an answer and a subtask), Disciple-COA learned the rule shown in the right hand side of Figure 5. In essence, a rule is a complex IF-THEN structure that specifies one or several conditions under which the task from the IF part can be reduced to the task(s) from the THEN part. Each rule includes a main condition that has to be satisfied in order for the rule to be applicable. In addition, it may include several except-when conditions (that should not hold in order for the rule to be applicable), "except-for" conditions (that specify instances that are negative exceptions of the rule) and "for" conditions (that specify positive exceptions). Partially learned rules, such as the ones showed in Figure 5, do not contain exact conditions, but plausible version spaces for these conditions. Each such plausible version space is represented by a plausible upper bound condition which, as an approximation, is more general than the exact (but not yet known) condition, and a plausible lower bound condition which, as an approximation, is less general than the exact condition. The rule from the right hand side of Figure 5 contains a plausible version space for the main condition and a plausible version space for an except-when condition. These version spaces are generated automatically by Disciple-COA from the creative reduction and its explanations, as discussed below. The generalizations of these explanations are also

included in the learned rule. In addition, the rule also contains the generalizations of the natural language phrases representing the Question and its Answer from the creative reduction. The Questions and the Answers play multiple roles in Disciple-COA. They help the expert in formalizing the problem solving process as task reduction. They are used by the natural language generation module of Disciple to generate the question and answer part of a task reduction step obtained by instantiating a rule. The elements recognized by Disciple in these natural language phrases (such as "RED-CSOP1") represent hints for the explanation generation process discussed below.

To learn a rule from a creative reduction, the agent will first try to find an explanation of why the reduction is correct. An explanation is a path of objects and features in the object ontology, and identifies the important characteristics of the objects from the creative reduction that should be kept in any correct generalization of the reduction. This significantly limits the number of candidates for the rule to be learned from the current creative reduction. All these candidate rules are represented by Disciple-COA as a plausible version space IF-THEN task reduction rule (Tecuci, 1998).

Finding the explanation for an example is also a cooperative process, where the agent proposes explanations ordered by their plausibility and the expert selects the correct ones. To generate the explanations, the agent uses an ordered set of heuristics for analogical reasoning. The heuristics are based on the hierarchies of tasks and features from Disciple's ontology, and on different types of structure similarity between the current example and the existing rules. In essence, Disciple identifies the rules that include tasks similar to the current example. Then it uses the explanations from which these rules have been learned as a guide to search for explanations of the current example. The expert may also help the agent in finding the explanations by proving hints. Guidance is also provided by the question and the answer from the example that identify the objects that should be part of the explanation, even though Disciple-COA does not have the ability to understand these natural language phrases.

As Disciple-COA learns plausible version space rules, it can use them to propose routine, innovative or inventive solutions to the current problems. The *routine solutions* are those that satisfy the plausible lower bound conditions of the task reduction rules and are very likely to be correct. Those that are not correct are kept as exceptions to the rule. The *innovative solutions* are those that satisfy the plausible upper bound conditions. These solutions may or may not be correct, but in each case will lead to a refinement of the task reduction rules that generated them. For instance, the second task reduction step in Figure 5 is an innovative solution that has been accepted by the expert and has led to the generalization of the plausible lower bound condition of the rule shown in the left hand

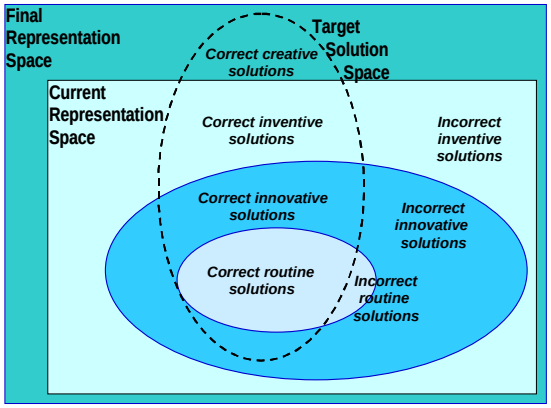

Figure 6: Types of generated solutions.

side of Figure 5. The *inventive solutions* are based on weaker forms of plausible reasoning (such as partial matching of the plausible conditions of the rules, and tasks similarity based on the structure of the ontology). An inventive task reduction step is based on several rules, and is generally a novel reduction of the tasks from these rules. From inventive solutions the agent will learn new plausible task reduction rules, as with the creative solutions, except that in such cases Disciple has more knowledge to guide the learning process.

Our research on plausible version spaces has its origins in Mitchell's influential work on version spaces and his candidate elimination algorithm (Mitchell, 1997), extending them along several dimensions, and leading to a powerful and practical mixed-initiative multistrategy learning approach that synergistically integrates a wide range of knowledge acquisition and machine learning strategies, including apprenticeship learning, empirical inductive learning from examples and explanations, and analogical learning. This approach is based on a powerful knowledge representation language that includes the frame-based OKBC knowledge model for the representation of the ontological knowledge, and complex task reduction rules with multiple conditions. Moreover, we do not make the assumption that the representation space for learning needs to be completely defined before learning can take place. On the contrary, the representation language is assumed incomplete and partially incorrect, and is itself evolving during rule learning through the improvement of the ontology. Because the learning process takes place in an evolving representation language, the various plausible bounds of a rule are subject to heuristic transformations that involve both generalization and specialization operations for each bound. These transformations are guided by hints, explanations and analogies. Therefore, the learning process is very efficient and does not suffer from any combinatorial explosion. Also, learning may take place even in the presence of exceptions, when there is no rule that discriminates between all the positive examples and all the negative examples.

4. Evaluation of Disciple-COA

With respect to the Disciple approach and its current implementation in Disciple-COA, we formulate the following claims that have been tested during the intensive evaluations of the HPKB program:

- they significantly speed up the process of building and updating a high performance knowledge base;
- they enable rapid learning of problem solving knowledge from domain experts, with limited assistance from knowledge engineers;
- the learned problem solving knowledge is of a good enough quality to assure a high degree of correctness of the solutions generated by the agent;
- the acquired problem solving knowledge assures a high performance of the problem solver.

The main DARPA evaluation took place during the period July 6-16, 1999, and was organized in five evaluation items of increasing difficulty. Each item consisted of several COAs and assessment questions to be answered. Item1 consisted of COAs and questions that were previously provided by DARPA to guide the development of the COA critiquing agents. Item2 included new test questions about the same COAs. Items 3, 4, and 5 consisted of previously unseen COAs that were increasingly more complex and required further development of the COA agents in order to properly answer the assessment questions. Each of the Items 3, 4 and 5 consisted of two phases, a testing phase, and a repair phase. In the testing phase each team had to provide initial system responses. Then the evaluator issued the model answers and each team had a limited amount of time to repair its system and to perform further knowledge acquisition, to generate revised system responses.

In addition to GMU, three other research groups developed COA critiquers as part of the HPKB program. Teknowledge and Cycorp developed a critiquer based on the CYC system (Lenat, 1995), taking advantage of CYC's large knowledge repository and inferential capabilities. The Expect group from ISI based its critiquer on the Expect shell for problem solving and knowledge acquisition (Kim and Gil, 1999). Finally, the LOOM group from ISI developed a case-based critiquer as an extension to the Loom system (MacGregor, 1999). The responses of each system were scored by a team of domain experts along the following dimensions and associated weights: Correctness-50% (matches model answer or is otherwise judged to be correct), Justification-30% (scored on presence, soundness, and level of detail), Lay Intelligibility-10% (degree to which a lay observer can understand the answer and the justification), Sources-10% (degree to which appropriate sources are noted), and Proactivity-10% extra credit (appropriate corrective actions or other information suggested to address the critique). Based on these scores, several classes of metrics have been computed, including Recall and Precision. Recall is obtained by dividing the score for all answers provided by a critiquer by the total number of model answers provided by the evaluator. "Precision" is obtained by dividing the same score by the total number of correct answers (both the model answers provided by the evaluator and the new answers provided by the critiquer). The results obtained by the four evaluated critiquers are presented in Table 1.

Table 1. Evaluation of the critiquers' performance

Metric	Tek/Cyc	ISI-Expect	Disciple	ISI-Loom
Recall	**56.81%**	**63.71%**	**114.69%**	**70.20%**
Precision	**62.61%**	**76.01%**	**81.99%**	**57.48%**

Figure 7 presents the breakdown by criteria of the Recall and Precision metrics, showing that Disciple-COA not only obtained the best overall results, but was also the only system to obtain high scores at each of these criteria.

Figure 8 compares the number of model answers provided by the evaluators (the 1st bar corresponds to evaluation items 3 and 4, and the 3rd bar corresponds to the evaluation Item 5), with the number of correct answers generated by Disciple-COA (the 2nd and the 4th bar). The bottom part of the 2nd and the 4th bar show the number of model answers matched by Disciple-COA, and the upper part shows the additional correct answers generated by Disciple-COA. The large proportion of these new answers demonstrates that a very knowledgeable expert can train Disciple to exhibit much of his or her expertise.

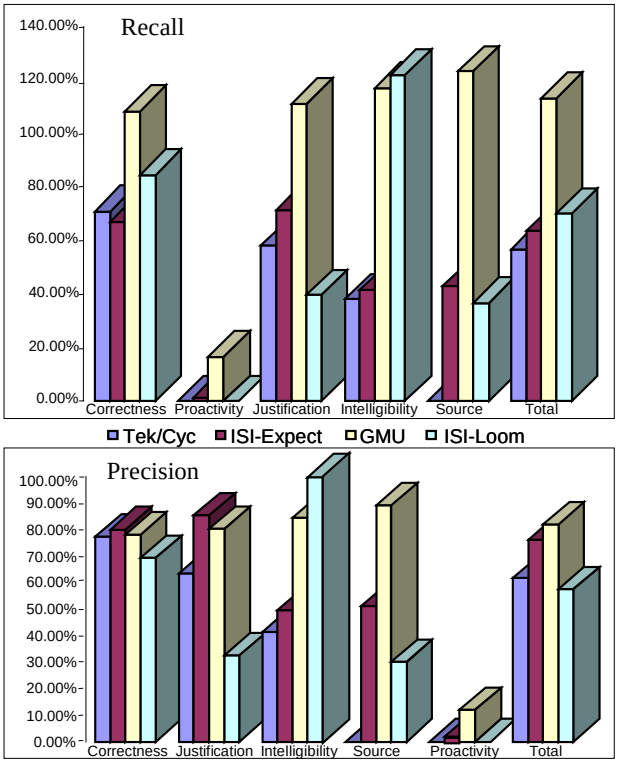

Figure 7: Recall and Precision breakdown by criteria.

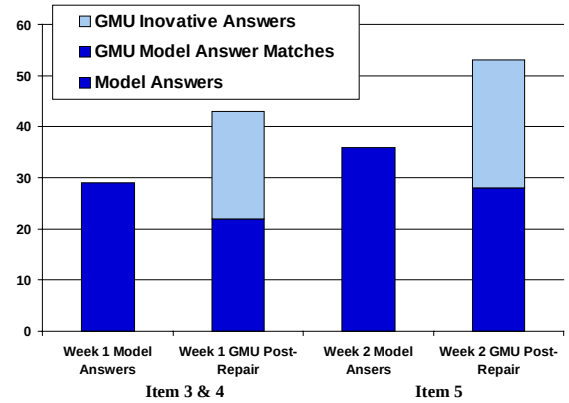

Figure 8: Disciple's model and new answers.

A very significant aspect is that all these results were obtained with an agent that was taught very rapidly. Figure 9 shows the evolution of the knowledge base of the Disciple-COA critiquer during the evaluation phase. Overall, the KB increased by 46% in 8 days, from a size of 6229 simple axioms equivalent to a size of 9092 simple axioms equivalent. The final knowledge base contained 801 concepts, 444 object and task features, 360 tasks and 342 PVS rules. Also, each COA was represented with around 1500 facts.

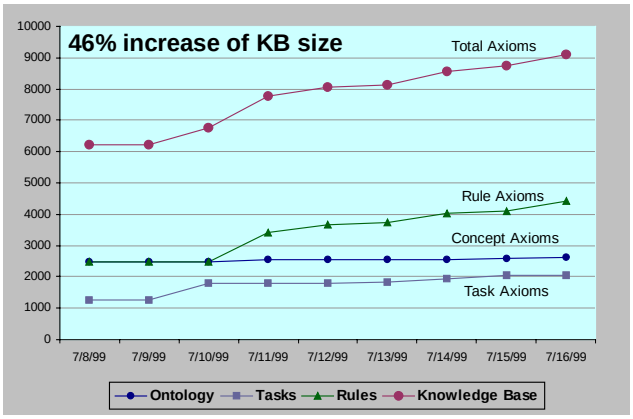

Figure 9: The evolution of the KB during evaluation.

5. Direct Knowledge Acquisition from Experts

In August 1999 we conducted a one week knowledge acquisition (KA) experiment with Disciple-COA at the US Army Battle Command Battle Lab in Fort Leavenworth, Kansas, to test the claim that domain experts that do not have prior knowledge engineering experience can teach Disciple-COA. The experiment involved four such military experts and had three phases: a joint training phase (day 1 to 3), an individual KA experiment (day 4), and a joint discussion of the experiment (day 5). The entire experiment was video-taped. The training for the experiment included a detailed presentation of Disciple's knowledge representation, problem solving and learning methods and tools. For the knowledge acquisition experiment itself, each expert received a copy of Disciple-COA with a partial knowledge base. This KB was obtained by removing the tasks and the rules from the complete KB of Disciple-COA. That is, the KB contained the complete ontology of objects, object features, and task features. We also provided the experts with the descriptions of three COAs, COA411, COA421, and COA51, to be used for training Disciple. These were the COAs used in the final phases of the DARPA's evaluation of all the critiquers. Finally, we provided and discussed with the experts the modeling of critiquing these COAs with respect to the principles of offensive and security. That is, we provided the experts with specific task reductions like the one from the middle of Figure 5, to guide the teaching of Disciple by the experts. After that, each expert taught Disciple-COA independently, while being supervised by a knowledge engineer who's role was to help the expert if he reached an impasse while using Disciple.

Figure 10 shows the evolution of the KB during the teaching process for one of the experts, being representative for all the four experts. In the morning the expert taught Disciple to critique COAs with respect to the Principle of Offensive and in the afternoon he taught it to critique COAs with respect to the Principle of Security. In both cases the expert used first COA411, then COA422 and then COA51. As one can see from Figure 10, Disciple initially learned more rules, and then the emphasis shifted to rule refinement. Therefore, the increase in the KB size is greater toward the beginning of the training process for each principle. On average, the teaching for the Principle of Offensive took 101minutes. During this time Disciple learned 14 tasks and 14 rules (147 simple axioms equivalent). The teaching for security took place in the afternoon and consisted of 72 minutes of expert-Disciple interactions. During this time Disciple learned 14 tasks and 12 rules (136 simple axioms equivalent). There was no or very limited assistance from the knowledge engineer with respect to teaching. The knowledge acquisition rate obtained during the experiment was very high (~ 9 tasks and 8 rules / hour expert, or 98 simple axioms equivalent/hour). At the end of this training process, Disciple-COA was able to correctly identify 17 strengths and weakness of the 3 COAs with respect to the

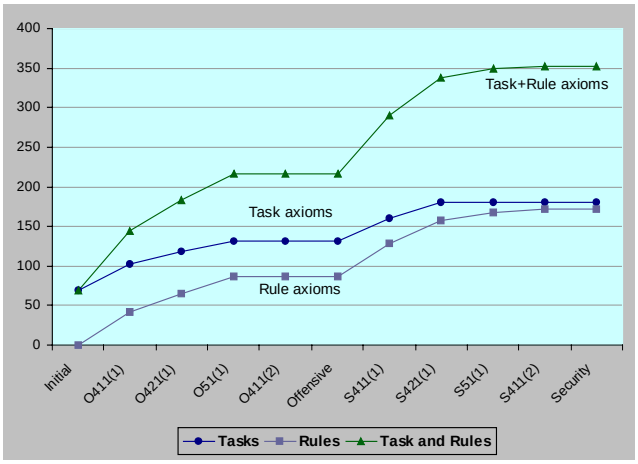

Figure 10: KB's evolution during the KA experiment.

principles of offensive and security.

After the experiment, each expert was asked to fill in a detailed questionnaire designed to collect subjective data for usability evaluation. The questionnaire was organized into three major sections: a) 6 overall questions, b) 40 detail questions, and c) comments and recommendation. The questions addressed three main dimensions of evaluation: effect on task performance, system usability, and system fit. In addition, each such dimension considered various criteria (e.g. system fit with the user or system fit with the organization), and even sub-criteria. All the answers took into account that Disciple-COA was a research prototype and not a commercial product, and were rated based on a scale of 1 to 5, with 1 denoting not at all and 5 denoting very. For illustration, Table 2 shows three questions and the answers provided by the four experts. An analysis of all the answers revealed again very high scores for the Disciple approach (82.39% on the fitness of the Disciple critiquing agent to their organizations, 76.32% in the effect that Disciple-COA would have on their task performance, and 73.72% in system's usability).

Table 2. Sample questions answered by the experts.

Questions	***Answers***
Do you think that Disciple is a useful tool for Knowledge Acquisition?	• Rating 5. Absolutely! The potential use of this tool by domain experts is only limited by there imagination - not their AI programming skills. • 5 • 4 • Yes, it allowed me to be consistent with logical thought.
Do you think that Disciple is a useful tool for Problem Solving?	• Rating 5. Yes. • 5 (absolutely) • 4 • Yes. As it develops and becomes tailored to the user, it will simplify the tedious tasks.
Were the procedures/ processes used in Disciple compatible with Army doctrine and/or decision making processes?	• Rating 5. As a minimum yes, as a maximum—better! • This again was done very well. • 4 • 4

6. Conclusion

The evaluation results demonstrate that the Disciple approach has reached a high level of maturity, being usable for teaching agents to solve complex real world problems. Figure 5 suggests the usefulness of this approach for knowledge acquisition. In the traditional, knowledge engineering approach to agent development, a knowledge engineer would need to manually define and debug rules like the ones in Figure 5. With Disciple, the domain expert (possibly assisted by a knowledge engineer) needs only to define specific reductions like those in the middle of Figure 5, because Disciple will learn and refine the corresponding rules. The current results provide a very encouraging support to the vision that, through further development of the Disciple approach, it will be some day possible for a normal user that has very limited knowledge engineering experience, to develop his or her personal assistant, as easily as such a person uses an email program, a word processor, or an internet browser today.

Acknowledgements

This research was supported by AFOSR and DARPA through the grant F49620-97-1-0188, as part of the HPKB program. The evaluation of the COA critiquers was conducted by Alphatech. The experts that participated in the BCBL knowledge acquisition experiment were LTC John N. Duquette, LTC Jay E. Farwell, MAJ Michael P. Bowman, and MAJ Dwayne E. Ptaschek. Bogdan Stanescu, Liviu Panait, Cristina Cascaval and Tom Jenkins are contributing to the new version of Disciple.

References

Alphatech, Inc. (1999). *HPKB Course of Action Challenge Problem Specification*. Burlington, MA.

Boicu M., Wright K., Marcu D., Lee S.W., Bowman M. and Tecuci G. (1999). The Disciple Integrated Shell and Methodology for Rapid Development of Knowledge-Based Agents. AAAI-99/IAAI-99. *Proceedings of the Intelligent Systems Demonstrations*, AAAI Press, Menlo Park, CA

Chaudhri, V.K., Farquhar, A., Fikes, R., Park, P.D., and Rice, J.P. (1998). OKBC: A Programmatic Foundation for Knowledge Base Interoperability. *Proceedings of the AAAI-98*, pp. 600 – 607, Menlo Park, CA: AAAI Press.

Cohen P., Schrag R., Jones E., Pease A., Lin A., Starr B., Gunning D., and Burke M. (1998). The DARPA High-Performance Knowledge Bases Project, *AI Magazine*, 19(4),25-49.

Kim, J. and Gil, Y. (1999). Deriving Expectations to Guide Knowledge Base Creation. *Proceedings of the AAAI-99/IAAI-99*. AAAI Press, Menlo Park, CA.

Lenat, D.B. (1995). CYC: A Large-scale Investment in Knowledge Infrastructure *Comm of ACM* 38(11):33-38.

Mitchell, T.M. (1997). *Machine Learning.* McGraw-Hill.

MacGregor, R. (1999). Retrospective on LOOM. Available online as: http://www.isi.edu/isd/LOOM/ papers/macgregor/Loom_Retrospective.html,

Tecuci, G. (1998). *Building Intelligent Agents: An apprenticeship multistrategy learning theory, methodology, tool and case studies*. London, Academic Press.

Tecuci, G., Boicu, M., Wright, K., Lee, S.W., Marcu, D. and Bowman, M. (1999). An Integrated Shell and Methodology for Rapid Development of Knowledge-Based Agents, *Proceedings of the AAAI-99/IAAI-99*. AAAI Press, Menlo Park, CA.