User-Agent Interactions in Mixed-Initiative Learning

Dorin Marcu, Mihai Boicu, Gheorghe Tecuci

Learning Agents Laboratory, Department of Computer Science, MSN 4A5, George Mason University, Fairfax, VA 22030
{dmarcu, mboicu, tecuci }@gmu.edu

Abstract

Mixed-initiative learning integrates complementary human and automated reasoning, taking advantage of their respective reasoning styles and computational strengths in order to solve complex learning problems. Mixed-initiative learning is at the basis of the Disciple approach for developing intelligent agents where a subject matter expert teaches an agent how to perform complex problem solving tasks and the agent learns from the expert, building and refining its knowledge base. Implementation of practical mixed-initiative learning systems, such as those from the Disciple family, requires advanced user-agent interactions to facilitate user-agent communication, the distribution of tasks between them, and the shift of initiative and control. This paper discusses some of these user-agent interaction issues in the context of the mixed-initiative rule learning method of the most recent version of the Disciple system.

Introduction

Mixed-initiative learning integrates complementary human and automated reasoning, taking advantage of their respective reasoning styles and computational strengths in order to solve complex learning problems. Although the basic idea underling mixed-initiative learning seems very simple, developing an effective mixed-initiative learning method is actually a very complex research issue. It requires the human and the automated agent to share representations, to communicate naturally, to properly divide their tasks and responsibilities, to be able to coordinate their actions, to take the initiative and to release the control.

We address the problem of mixed-initiative learning in the context of developing tools for enabling a subject matter expert who does not have knowledge engineering experience to build knowledge bases and agents. This will provide a solution to the well-known knowledge acquisition bottleneck in the framework of the Disciple theory (Tecuci, 1998, Tecuci et al., 1999). The approach is to develop a powerful learning agent that can be taught directly by a subject matter expert. The expert should be able to communicate his or her expertise to the learning agent in a very natural way, similar to how the expert would communicate it to a human apprentice while solving problems in cooperation. Through natural interactions, the agent will be guided in learning complex problem solving rules, and in extending and correcting its knowledge base.

Building the knowledge base of the agent is an example of a problem that, by its very nature, requires a mixed-initiative solution. Indeed, neither the subject matter expert, nor the learning agent can solve this problem independently. While the subject matter expert has the knowledge to be represented in the knowledge base, he is not a knowledge engineer and cannot properly formalize it. On the other hand, the learning agent obviously does not have the knowledge to be represented, but it can incorporate knowledge engineering methods to formalize expert's knowledge. The goal is then to divide the responsibility between the expert and the agent for those elements of knowledge engineering for which they have the most knowledge and aptitude, such that together they form a complete team for knowledge base development. As mentioned above, this will require the coordination of the interaction between the expert and the agent, and the shift of initiative and control during knowledge acquisition and learning.

In this paper we are discussing some of the user-agent interaction issues involved in mixed-initiative learning, using as an example the rule learning method of the latest system from the Disciple family that is currently under development in the Learning Agents Laboratory (http://lalab.gmu.edu).

Agent Building Methodology

The current Disciple agent shell consists of an integrated set of knowledge acquisition, learning and problem solving modules for a generic knowledge base structured into two main components: an ontology that defines the objects from a specific application domain, and a set of problem solving rules expressed with these objects. The problem solving approach of a Disciple agent 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. Therefore, the rules from the knowledge base are task reduction rules. The ontology consists of hierarchical descriptions of objects and features, represented as frames, according to the knowledge model of the Open Knowledge Base Connectivity protocol (Chaudhri et al. 1998).

The process of building the knowledge base of a Disciple agent includes three main phases: domain modeling, ontology development, and agent teaching.

In the domain modeling phases the expert and the knowledge engineer define, at a conceptual level, a model of the application domain that will make explicit how the expert performs his tasks, based on the task reduction paradigm. They consider a set of specific tasks that constitute a representative set for the 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. This process also produces an informal specification of the objects needed to be represented into the agent's ontology (Bowman et al., 2000).

During the ontology development phase the knowledge engineer and the expert import some of the object concepts identified in the previous phase from existing repositories of knowledge, and define the rest of them. This phase results in an initial knowledge base that contains an incomplete ontology but no rules (Boicu et al., 1999).

The third phase of agent development is a mixed-initiative teaching and learning process, where the subject matter expert teaches the agent how to perform its tasks in a way that resembles how the expert would teach a human apprentice when solving problems in cooperation. As a result, the agent will learn problem solving rules from the expert, and will also extend and update its ontology. Then the developed agent can be used by a non-expert user, or it could be an assistant to an expert.

Disciple has been applied in the High Performance Knowledge Bases program supported by DARPA and AFOSR (Cohen et al., 1998), to develop a knowledge-based agent for solving the workaround challenge problem (Jones, 1998). This workaround agent has to determine the best plan of actions for a military unit to bypass or reconstitute damage to a transportation infrastructure. For instance, unit91010, located at site100, has to cross a river, but the bridge over the river at site203 has been destroyed. The generated plan may involve obtaining bridge building and rubble clearing equipment from other units, bringing them to the site of the bridge, preparing the area, and installing a new bridge. The plan has to indicate a minimal and expected time of completion, the resources needed, and the transportation capacity of the new bridge.

In the following sections we discuss the user-agent interactions occurring during the mixed-initiative learning of workaround rules.

Mixed-initiative Rule Learning

The expert teaches the Disciple-workaround agent by jointly developing workaround plans, as described in the following. First, the expert formulates the workaround task to be performed. Then Disciple tries to successively reduce this task to simpler tasks by applying the task reduction rules from its knowledge base. The expert has to analyze each reduction proposed by the agent, deciding whether to accept it or to reject it. In both cases the rule refiner is invoked to either generalize or to specialize the rule that generated the solution. If the agent was not able to propose any solution or the proposed solution was rejected, then the expert has to provide a solution. In this case the rule learner is invoked to learn a new rule. This situation is illustrated in Figure 1. The left hand side represents the reasoning process of the expert:

The task to accomplish is:
Workaround destroyed bridge at site100 over site203 by unit91010
What engineering technique could be used?
Fixed military bridge because it has a high enough MLC rating and the gap length is only 12m
Therefore accomplish the task:
Workaround destroyed bridge at site100 over site203 by unit91010 using a fixed military bridge with minor preparation

As one can see, expert's reasoning is represented in natural language, at an abstract and qualitative level. The QUESTION and its ANSWER provide an explanation of why the top task in Figure 1 is reduced to the bottom task. While this explanation is very natural to a human expert, a learning agent cannot understand it. The explanation that would be understood by the agent is represented in the upper right part of Figure 1, and consists of various relations between

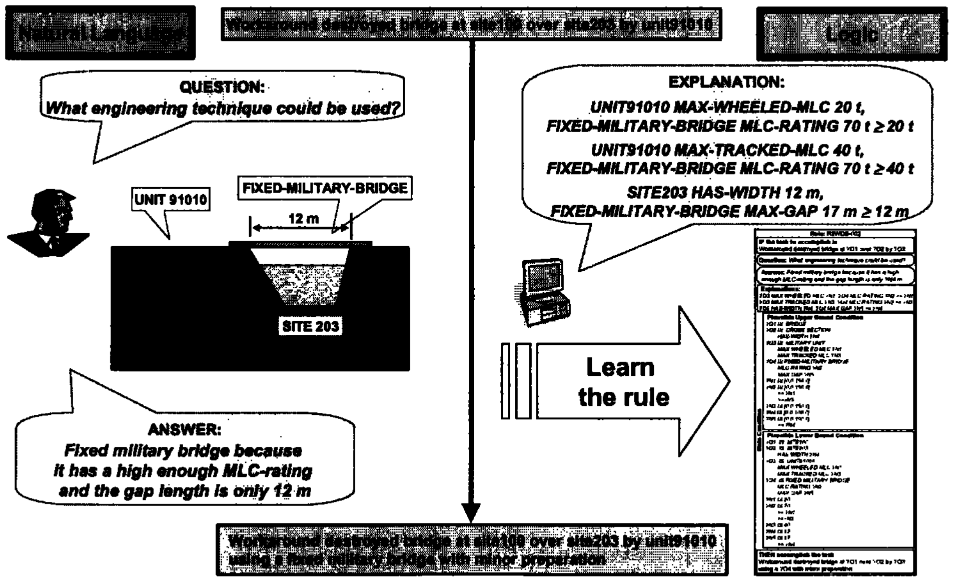
Natural Language
Workaround destroyed bridge at site100 over site203 by unit91010
Logic
QUESTION:
What engineering technique could be used?
UNIT 91010
FIXED-MILITARY-BRIDGE
12 m
SITE 203
ANSWER:
Fixed military bridge because it has a high enough MLC-rating and the gap length is only 12 m
EXPLANATION:
UNIT91010 MAX-WHEELED-MLC 20 t,
FIXED-MILITARY-BRIDGE MLC-RATING 70 t ≥ 20 t
UNIT91010 MAX-TRACKED-MLC 40 t,
FIXED-MILITARY-BRIDGE MLC-RATING 70 t ≥ 40 t
SITE203 HAS-WIDTH 12 m,
FIXED-MILITARY-BRIDGE MAX-GAP 17 m ≥ 12 m
Learn the rule
Workaround destroyed bridge at site100 over site203 by unit91010 using a fixed military bridge with minor preparation

Figure 1: Teaching the Disciple agent.

certain elements from the agent's ontology.
The first explanation piece is:

UNIT91010 MAX-WHEELED-MLC 20 t,
FIXED-MILITARY-BRIDGE MLC-RATING 70 t ≥ 20 t

This explanation piece states, in Disciple's formal language, that the heaviest wheeled vehicles of unit91010 weigh 20 tons and this is less than the 70 tons that may be supported by a fixed military bridge. The second explanation piece expresses a similar thing with respect to the tracked vehicles of unit91010. Collectively, these two explanation pieces correspond to the expert's explanation that the fixed military bridge has a high enough MLC-rating. The last explanation piece states that the width of the river gap over which the fixed military bridge is to be installed is within the range of this type of bridge. This explanation corresponds to the expert's explanation "the gap length is only 12m". While an expert can understand the meaning of these formal expressions, he cannot define them because he is not a knowledge engineer. For one thing, he would need to use the formal language of the agent. But this would not be enough. He would also need to know the names of the potentially many thousands of concepts and features from the agent's ontology.

While defining the formal explanations of this task reduction step is beyond the individual capabilities of the expert and the agent, it is not beyond their joint capabilities. Finding these explanation pieces is a mixed-initiative process of searching the agent's ontology, an explanation piece being a path of objects and relations in this ontology. In essence, the agent will use analogical reasoning and help from the expert to identify and propose a set of plausible explanation pieces from which the expert will have to select the correct ones.

One analogical reasoning heuristic is the following one:

1. Look for a rule R_k that reduces the current task (i.e. **"Workaround destroyed bridge at ?O1 over ?O2 by ?O3"**), even though this rule is not applicable in the current situation.
2. Extract the explanations E_g from the rule R_k.
3. Look for explanations of the current task reduction (i.e. the task reduction in Figure 1) that are similar with E_g, and propose them to the expert.

This heuristic is based on the observation that the explanations of the alternative reductions of a task tend to have similar structures. The same factors are considered, but the relationships between them are different. For instance, to workaround a destroyed bridge one could use a fixed military bridge with minor preparations (as in Figure 1), or with gap reduction, or with slope reduction. In a particular situation, the decision of which of these reductions to perform depends upon the specific relationships between the dimensions of the bridge and the dimensions of the river gap. If the agent has already learned a rule corresponding to any of these reductions, then learning the rules corresponding to the other reductions is much simpler because the agent can propose explanations by analogy with those of the learned rule.

Let us consider again Figure 1. To provide the solution to the agent, the expert uses the Example Editor that is already initialized with the task to be reduced. Then he has to specify the question, the answer, and the subtask. Notice, however, that the above heuristic relies only on the task to be reduced. Therefore, the agent can start immediately to search for plausible explanations, in parallel with the completion of the example by the expert. Nevertheless, as the completion of the example advances, the agent could extract additional hints from the expert's actions. For instance, once the expert has specified the question and the answer, the objects and the relations referred there (such as fixed military bridge, MLC-rating, 12m) could be used as searching hints. Indeed, they tell the agent that the potentially useful explanations are those that contain these elements. After the expert has completed the definition of the example, the agent may initiate other analogical reasoning heuristics that take into account the entire form of the example (not just its top task). The expert could also give explicit hints to the agent. A hint is a fragment of an explanation, such as an object or a relationship between two objects, or any other abstraction of the explanation.

From the task reduction example and its explanations shown in Figure 1, the agent automatically generates the general task reduction rule shown in Figure 2. The generalization method is presented in (Tecuci, 1998). This rule is a complex IF-THEN structure that specifies the condition under which the task from the IF part can be reduced to the tasks from the THEN part. Partially learned rules, such as the one shown in Figure 2, 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. In addition to the main condition, the learned rule also includes generalizations of the explanations, and of the question and its answer, which basically represent the same information at higher levels of abstraction and formalization.

An important aspect of the interactions between the expert and the agent, that was illustrated above, concerns the exchange of information between them, in each direction, in forms that the source can easily create and the recipient can easily understand. There is a clear complementariness between the expert and the agent with respect to their language generation and understanding capabilities. The expert can easily generate and understand natural language. He can also understand quite easily expressions in a formal language but he cannot generate such expressions. On the contrary, the agent has very limited natural language understanding capabilities, but its generation capabilities, especially with respect to formal expressions, are powerful. Our mixed-initiative approach takes these different capabilities into account. For instance, when the agent needs some information from the expert, rather than asking the expert to provide it (which would be difficult because it would require the expert to create a sentence in agent's formal language), the agent can formulate the question and possible answers and ask the expert to indicate the correct one. Or, when the agent cannot hypothesize the correct answer, it could ask the expert to only provide a hint and will use this hint to

```
Rule: R$WDB-002
IF the task to accomplish is
Workaround destroyed bridge at ?O1 over ?O2 by ?O3
Question: What engineering technique could be used?
Answer: Fixed military bridge because it has a high
enough MLC-rating and the gap length is only ?N4 m
Explanations:
?O3 MAX-WHEELED-MLC ?N1, ?O4 MLC-RATING ?N2 >= ?N1
?O3 MAX-TRACKED-MLC ?N3, ?O4 MLC-RATING ?N2 >= ?N3
?O2 HAS-WIDTH ?N4, ?O4 MAX-GAP ?N5 >= ?N4

Main Condition
  Plausible Upper Bound Condition
  ?O1   IS   BRIDGE
  ?O2   IS   CROSS-SECTION
             HAS-WIDTH   ?N4
  ?O3   IS   MILITARY-UNIT
             MAX-WHEELED-MLC   ?N1
             MAX-TRACKED-MLC   ?N3
  ?O4   IS   FIXED-MILITARY-BRIDGE
             MLC-RATING   ?N2
             MAX-GAP   ?N5
  ?N1   IS-IN   [0.0 150.0]
  ?N2   IS-IN   [0.0 150.0]
        >=   ?N1
        >=   ?N3
  ?N3   IS-IN   [0.0 150.0]
  ?N4   IS-IN   [0.0 100.0]
  ?N5   IS-IN   [0.0 100.0]
        >=   ?N4

  Plausible Lower Bound Condition
  ?O1   IS   SITE100
  ?O2   IS   SITE203
             HAS-WIDTH   ?N4
  ?O3   IS   UNIT91010
             MAX-WHEELED-MLC   ?N1
             MAX-TRACKED-MLC   ?N3
  ?O4   IS   FIXED-MILITARY-BRIDGE
             MLC-RATING   ?N2
             MAX-GAP   ?N5
  ?N1   IS-IN   [20 20]
  ?N2   IS-IN   [70 70]
        >=   ?N1
        >=   ?N3
  ?N3   IS-IN   [40 40]
  ?N4   IS-IN   [12 12]
  ?N5   IS-IN   [17 17]
        >=   ?N4

THEN accomplish the task
Workaround destroyed bridge at ?O1 over ?O2 by ?O3
using a ?O4 with minor preparation
```

Figure 2: The learned rule.

hypothesize possible answers. Also, when the expert has to communicate something to the agent, he can use a graphical interface to point and click. For instance, he can give a hint for an explanation by simply pointing to an object in the current example.

In the following section we will discuss some of the mechanisms that enable the type of user-agent interactions illustrated above.

Formalization of User-Agent Interactions

We have defined a *Mixed Initiative Description Language* to formally represent the algorithms to be executed through mixed-initiative reasoning. This is a rule-based language where each rule describes the execution of a higher level task in terms of simpler subtasks, their preconditions, the messages that could be exchanged between them, different possible flows of execution of these tasks, their possible results, and who has to execute them (the expert or the agent, or both of them). Thus the tasks to be performed by the expert and by the agent are described in terms of their subtasks. This provides a useful similarity with the general problem solving approach of Disciple.

For example, Figure 3 is a graphical representation of the decomposition rule corresponding to the mixed-initiative rule learning algorithm described in the previous section and illustrated in Figure 1. The Learn_rule task is decomposed into four subtasks: Define_example, Explain_example, Generate_explanations and Create_rule. The first two subtasks are non elementary, each of them being decomposed by other rules into even simpler tasks, some to be executed by the agent and some by the expert. The last two subtasks are elementary agent tasks.

The rule also represents the control and the communication flow between the component tasks. The solid arrows represent the main execution flow. For instance, the Define_example and the Generate_explanations tasks are executed in parallel. Indeed, as discussed in the previous section, the agent can start immediately to look for explanations, based only on the top task (i.e. the IF task) of the example. The dashed arrows indicate the messages that could be exchanged by the tasks during their execution. For instance, as the expert defines the question and the answer part of the example, this information is communicated to the Generate_explanations task (through "Example_updated *example*"), guiding the agent in filtering the generated explanations or in generating new ones. The Define_example task ends with a complete example. This is sent to the Explain_example task that can now start its execution. The communication between the Explain_example task and the Generate_explanations task is also represented in the rule. The Explain_example task can provide hints to the Generate_explanations task, the Generate_explanations task can propose a list of plausible explanations, and the Explain_example task can indicate the selection of some of these explanations by the expert. The Explain_example and Generate_explanations tasks end when the expert is satisfied with the generated explanations. This initiates the Create_rule task where the agent generates the learned rule and provides it as the result of the Learn_rule task. It is important to stress that the rule in Figure 3 only indicates which tasks can potentially be executed. In a given rule learning situation only some of these tasks will actually be

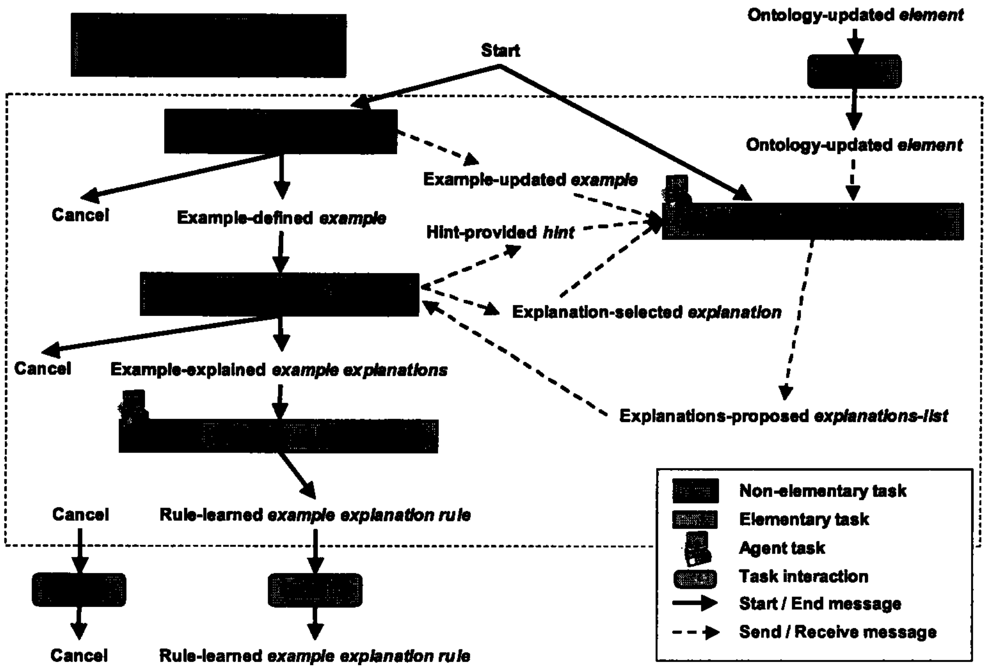

Figure 3: Sample rule in the Mixed-Initiative Description Language.

executed. For instance, the user may cancel the definition of the example, in which case the other tasks are no longer executed and the Learn_rule task ends with the "Cancel" result.

This representation allows for the description of subtle interactions between various tasks. For instance, an interesting indirect hint for explanation generation is given by any ontology modification made by the expert during the rule learning process. This will likely be related to some object or feature that should appear in the explanation. As indicated in the top right part of Figure 3, any ontology modification by the expert will trigger a message to the Generate_explanations task, specifying the updated element.

As can be seen, this rule description language allows breaking down complex tasks that need to be jointly performed by the expert and by the agent, into simpler subtasks, and to clearly divide the responsibility between the expert and the agent for those of these tasks for which they have the most aptitude. For instance, Define_example falls mostly under the responsibility of the expert, but Create_rule is entirely under the responsibility of the agent. The rule expresses also the potential shift in initiative and control. For instance, the agent takes the initiative of looking for explanations even before the example has been completely defined. This is important because explanation generation is a computationally intensive process and the example definition does not require significant resources.

The Task Agenda

The interpreter of the Mixed-Initiative Description Language provides for a mixed-initiative control of task execution. An important component of this interpreter is a task agenda that displays all the tasks relevant to the user at different levels of abstraction, providing an increasing efficiency of interaction as the human expert becomes more experienced in collaborating with the agent. The task agenda will also highlight the tasks that are currently being executed and those that can be scheduled for execution at that time, to keep the expert informed with both the current and the anticipated status of the execution. This task execution framework allows the agent to reason about the current tasks performed by the expert and to exhibit proactive behavior by executing support tasks in advance, in order to facilitate the initiation or completion of an important expert task that will follow. There are three complementary views of the task agenda: the decomposition view, the execution view and the continuation view. They will be briefly presented in the following.

The Decomposition View

The decomposition view is illustrated in Figure 4. It shows all the possible subtasks of a complex task, independent of

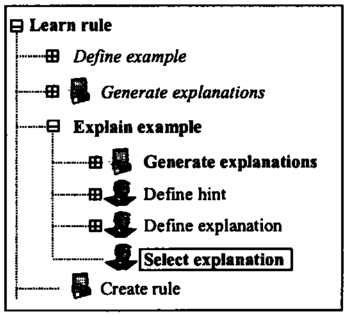

Figure 4: The decomposition view of the task agenda.

whether they are executed or not. The agent's tasks are prefixed by a computer icon and the expert's tasks are prefixed by an expert icon For example, the complex task "Learn rule" is decomposed into four simpler subtasks, "Define example", "Generate Explanations", "Explain example" and "Create rule". The "Generate Explanations" and "Create rule" tasks are performed by the agent. The others are mixed-initiative tasks that are further decomposed into simpler subtasks, which are either agent tasks or expert tasks. Figure 4 also shows the decomposition of the task "Explain example" into four subtasks. The tasks that have already been executed are shown in italics (e.g. "*Define example*"), those that are currently in execution are shown in bold (e.g. "**Generate explanation**") and those that require the expert's input are surrounded by a border line (e.g. "Select explanation"). The "Generate explanations" task that appears as subtask of "Learn rule" is the task that generates explanations during the definition of the example. The "Generate explanations" task that appears as subtask of "Explain example" is the task that generates explanations after the example was completely defined.

The Execution View

The dynamics of the task execution process is displayed in the task execution view (see Figure 5). In this view the tasks are structured in a hierarchy based on the dependencies between them and the flow of control. A task is completed when one of its children is completed – the other children are just alternative options. For example, in order to complete the task "Learn rule", the task "Define example" was activated and completed, followed by "Explain example", which requires the completion of the task "Generate explanations". "Generate explanations" waits for the completion of "Select explanation", which is the task that currently requires the expert's input. The current execution chain is displayed in bold. In order to complete the task "Explain example", the expert has several alternatives, shown in Figure 5. He may define some hints, triggering the "Define hint" task. He may manually define explanations and therefore start the "Define explanation" task. Or he may initiate the "Create rule" task by selecting the "Complete rule learning" alternative.

The execution view is synchronized with the decomposition view and it contains similar information but focuses the expert on the task execution chain.

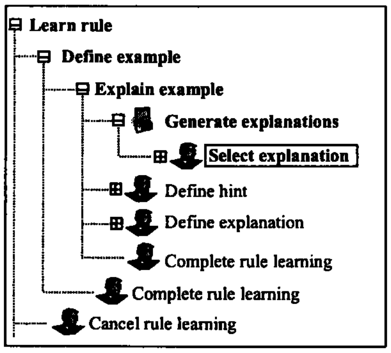
```
⊟ Learn rule
├─⊟ Define example
│  ├─⊟ Explain example
│  │  ├─⊟ Generate explanations
│  │  │  └─⊞ Select explanation
│  │  ├─⊞ Define hint
│  │  ├─⊞ Define explanation
│  │  └── Complete rule learning
│  └── Complete rule learning
└── Cancel rule learning
```

Figure 5: The execution view of the task agenda.

The Continuation View

The continuation view is the most helpful view for the user. It displays all the possible ways to continue or complete the current active task in either the decomposition or the execution views. In Figure 6 the continuation view shows all the possible continuations of the task "Select explanation", which is the active task in both the decomposition view (Figure 4) and the execution view (Figure 5). After the expert selects an explanation the task "Select explanation" is completed, and the expert may define a new hint or a new explanation and therefore starting the tasks "Define hint" or "Define explanation" respectively, complete the rule learning process or cancel it by initiating the corresponding tasks. The agent may also start the "Generate explanations" task. Some of the options displayed in the continuation view can also be found in the interface associated with the task (for example completing the rule learning task can be done by clicking the "Complete" button in the interface or by selecting and executing the "Complete rule learning" task from the continuation view). However, others options are not directly accessible from that interface (but are accessible from a previous interface). We plan to transform this view into a help wizard which, in addition to indicating the possible task continuation options, will also provide an explanation of each option and the expected effects on the current state.

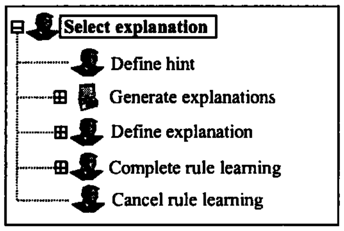
```
⊟ Select explanation
├── Define hint
├─⊞ Generate explanations
├─⊞ Define explanation
├─⊞ Complete rule learning
└── Cancel rule learning
```

Figure 6: The continuation view of the task agenda.

Conclusions

In this paper we have addressed some of the user-agent interactions in the mixed-initiative rule learning method of Disciple. While some of these results are still preliminary, they have already been partially validated in the High Performance Knowledge Bases program. In this program, the Disciple-Workaround agent and the Disciple-COA agent (another Disciple agent developed for the course of action challenge problem), have been evaluated by Alphatech in several intensive studies, demonstrating better results than the other systems developed for the same challenge problems. Descriptions of these evaluations and of their results are presented in (Tecuci et al., 1999), for the workaround challenge problem, and in (Tecuci et al., 2000a), for the course of action challenge problem.

To test the claim that domain experts can teach a Disciple agent, we conducted a knowledge acquisition experiment at the US Army Battle Command Battle Lab, in Fort Leavenworth, Kansas. In this experiment, four military experts that did not have any prior knowledge engineering experience received around 16 effective hours of training in

Artificial Intelligence and the use of Disciple-COA. They then succeeded in training Disciple to critique military courses of actions with respect to the Principle of Offensive and the Principle of Security, in about three hours, following a modeling of the critiquing process that was discussed with them at the beginning of the experiment. At the end of the experiment they completed a detailed questionnaire that revealed high scores for the perceived usefulness and usability of Disciple. A detailed description of this experiment is presented in (Tecuci et al., 2000b).

We give credit for these successes of the Disciple approach to the extensive use of mixed-initiative teaching and learning methods.

The Disciple approach continues to be developed in the new DARPA program called "Rapid Knowledge Formation." This time the challenge problem is the identification and testing of the strategic center of gravity candidates of opposing forces in military conflicts. This work in done in cooperation with the US Army War College.

Acknowledgments. 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, and by the Air Force Office of Scientific Research (AFOSR) under grant no. F49620-00-1-0072. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA, AFOSR, AFRL, or the U.S. Government. Bogdan Stanescu, Michael Bowman, Cristina Cascaval, Catalin Balan, Elena Popovici, and other members of the LALAB have contributed to successive versions of Disciple.

References

Boicu M., Tecuci G., Bowman M., Marcu D., Lee S.W., and Wright K. 1999. A Problem-Oriented Approach to Ontology Creation and Maintenance. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence, Workshop on Ontology Management*, Menlo Park, CA: AAAI Press.

Boicu M., Tecuci G., Marcu D., Bowman M., Shyr P., Ciucu F., and Levcovici C. 2000. Disciple-COA: From Agent Programming to Agent Teaching. In *Proceedings of the Seventeenth International Conference on Machine Learning*, Stanford, CA: Morgan Kaufmann.

Bowman M., Tecuci G., and Boicu M. 2000. A Methodology for Modeling and Representing Expert Knowledge that Supports Teaching-Based Intelligent. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and the Twelfth Conference on Innovative Application of Artificial Intelligence*, 1065. Menlo Park, CA: AAAI Press.

Chaudhri V., Farquhar A., Fikes R., Park D., and Rice J. 1998. OKBC: A Programmatic Foundation for Knowledge Base Interoperability. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, 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.

Jones E. 1998. HPKB Year 1 End-to-End Battlespace Challenge Problem. Specification, Burlington, MA.

Tecuci, G. 1998. *Building Intelligent Agents: An Apprenticeship Multistrategy Learning Theory, Methodology, Tool and Case Studies*. London, England: 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. In: *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, 250-257. Menlo Park, CA: AAAI Press.

Tecuci G., Boicu M., Marcu D., Bowman M., Ciucu F., and Levcovici C. 2000a. Rapid Development of a High Performance Knowledge Base for Course of Action Critiquing. In: *Proceedings of the Seventeenth National Conference on Artificial Intelligence and the Twelfth Conference on Innovative Application of Artificial Intelligence*, 1046-1053. Menlo Park, CA: AAAI Press.

Tecuci G., Boicu M., Bowman M., Marcu D., Shyr P., and Cascaval C. 2000b. An Experiment in Agent Teaching by Subject Matter Experts. *International Journal of Human-Computer Studies* 53: 583-610.