Building an Adaptive Agent to Monitor and Repair the Electrical Power System of an Orbital Satellite

Gheorghe Tecuci [1,2] **Michael R. Hieb** [1] **Tomasz Dybala** [1]
{tecuci, hieb, tdybala}@gmu.edu
[1] Department of Computer Science, George Mason University, Fairfax, Virginia, USA
[2] Center for Artificial Intelligence, Romanian Academy, Bucharest, Romania

Abstract

Over several years we have developed a multistrategy apprenticeship learning methodology for building knowledge-based systems. Recently we have developed and applied our methodology to building intelligent agents. This methodology allows a subject matter expert to build an agent in the same way in which the expert would teach a human apprentice. The expert will give the agent specific examples of problems and solutions, explanations of these solutions, or supervise the agent as it solves new problems. During such interactions, the agent learns general rules and concepts, continuously extending and improving its knowledge base. In this paper we present initial results on applying this methodology to build an intelligent adaptive agent for monitoring and repair of the electrical power system of an orbital satellite, stressing the interaction with the expert during apprenticeship learning.

1. Introduction

Automating the process of building knowledge bases has long been the goal of both Knowledge Acquisition and Machine Learning. The focus of knowledge acquisition has been to improve and partially automate the acquisition of knowledge from human experts by a knowledge engineer. This approach has had limited success, mostly because of the communications problems between the subject matter expert and the knowledge engineer, which requires many iterations before converging to an acceptable knowledge base.

In contrast, machine learning has focused on mostly autonomous algorithms for acquiring and improving the organization of knowledge. However, because of the complexity of this problem, the application of this approach tends to be limited to very simple domains. While knowledge acquisition research has generally avoided using machine learning techniques, relying on the knowledge engineer, machine learning research has generally avoided involving a human expert in the learning loop. We think that neither approach is sufficient, and that the automation of knowledge acquisition should be based on a direct interaction between a human subject matter expert and a learning system (Tecuci, Kedar, and Kodratoff, 1994).

A human expert and a learning system have complementary strengths. Problems that are extremely difficult for one may be easy for the other. For instance, automated learning systems have traditionally had difficulty assigning credit or blame to individual decisions that lead to overall results, but this process is generally easy for a human expert. Also, the "new terms" problem in the field of Machine Learning (i.e. extending the representation language with new terms when these terms cannot represent the concept to be learned), is very difficult for an autonomous learner, but could be quite easy for a human expert (Tecuci and Hieb, 1994). On the other hand, there are many problems that are much more difficult for a human expert than for a learning system as, for instance, the generation of general concepts or rules that account for specific examples, and the

updating of the knowledge base to consistently integrate the learned knowledge.

Over several years we have developed a multistrategy apprenticeship learning methodology for building knowledge-based systems (Tecuci, 1988, 1992; Tecuci and Kodratoff, 1990; Tecuci and Hieb, 1994; Tecuci et al, 1994). Recently we have developed and applied our methodology to building intelligent agents. This methodology allows a subject matter expert to build an agent in the same way in which the expert would teach a human apprentice. The expert will give the agent specific examples of problems and solutions, explanations of these solutions, or supervise the agent as it solves new problems. During such interactions, the agent learns general rules and concepts, continuously extending and improving its knowledge base. This process produces validated knowledge-based agents, because it is based on an expert interacting with, checking and correcting the way the agent solve problems.

Successive versions of this methodology have been implemented in several systems (e.g. DISCIPLE (Tecuci, 1988; Tecuci and Kodratoff, 1990) NeoDISCIPLE, (Tecuci, 1992; Tecuci and Hieb, 1994) and CAPTAIN (Tecuci et al, 1994)), which have been applied to a variety of domains: loudspeaker manufacturing, reactions in inorganic chemistry, high-level robot planning, question-answering in geography and, more recently, military command agents in distributed interactive simulation environments.

In this paper we present initial results on updating and applying this methodology to build an intelligent adaptive agent for monitoring and repair of the electrical power system of an orbital satellite. The system DISCIPLE-OPS, which implements this methodology, provides an integrated framework that facilitates

1) building intelligent agents through knowledge elicitation and interactive apprenticeship learning from experts; and
2) making these agents adapt and improve during their normal use through autonomous learning.

This paper is organized as follows. Section 2 presents the application domain. Section 3 describes a simulator of the electrical power system to be monitored. Section 4 presents the architecture of the intelligent agent, together with its decision-making and learning methods. Section 5 presents the methodology for building the agent. Finally, section 6 concludes the paper with a discussion of our agent-building approach.

2. An Exemplary Problem

The main objective of the Electrical Power System (EPS) is to provide an Orbital Satellite with a steady supply of electrical power. The EPS is capable of self-preservation in emergencies, but it is not capable of maintaining optimum productivity without outside support. If not controlled, the power production of the EPS will eventually fail, leaving its users unsupported. Therefore, the EPS must be monitored at all times. This function could be fulfilled by an intelligent agent acting as a ground station that monitors telemetry from sensors in the solar powered EPS for anomalous behavior, and generates repairs by forming and uplinking commands to the spacecraft. The agent itself is supervised by a human operator who may correct its behavior. The basic interaction between the spacecraft, the intelligent agent, and the human operator is shown in Figure 1. During such interactions, the agent learns from its own actions and the commands issued by the human operator, gradually acquiring the expertise of the operator until it could operate autonomously.

In the following sections we present a methodology for building the intelligent agent in Figure 1. However, instead of controlling the EPS, the agent will control a simulator of the EPS. This simulator is briefly described in section 3.

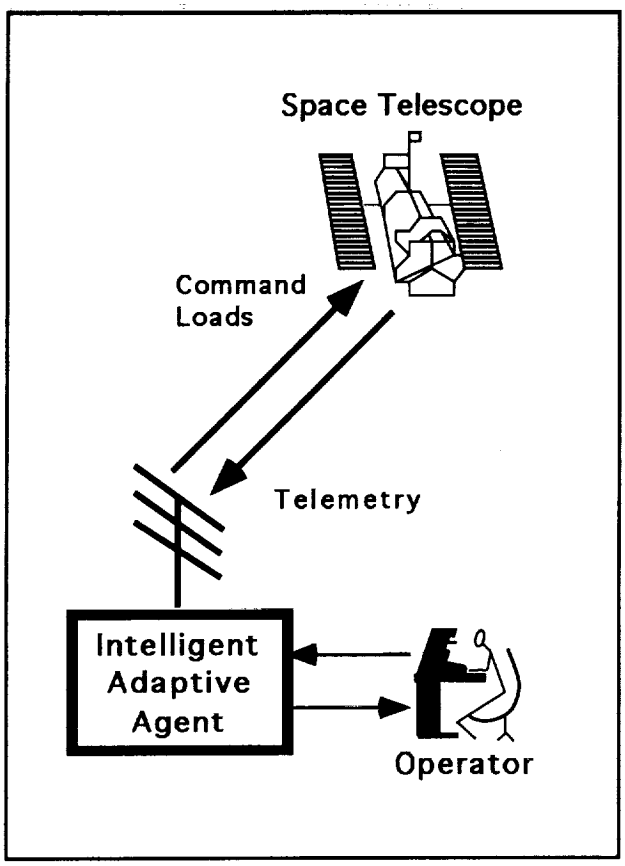

Figure 1. Basic interaction between the spacecraft, the intelligent agent, and the human operator.

3. A Simulator of the Electrical Power System of an Orbital Satellite

A simulator of an Orbital Satellite Electrical Power System has been developed by NASA Goddard Code 522.3 (Silverman et al., 1989; Hieb 1990; Hieb, Silverman & Mezher, 1992). The simulator was not designed to duplicate the EPS of the actual orbital satellite, but is a scaled down version which only simulates selected basic functions and problems. The goal of the design was to capture the essence of EPS problems and implement them in the simulator. Therefore, this software simulator provides a challenge to the intelligent agent which has close similarities to the real problems encountered at NASA control centers. Figure 2 is a diagram of the EPS simulator. The following components are represented in the simulator.

Solar arrays. There are two solar array panels in the simulator, each with ten solar cells. Power production takes place in the solar arrays. Orientation and cell errors are randomly generated with given certain limits and probabilities. Cell errors are fixed by resetting the appropriate solar array.

The network. The network is a set of power lines equipped with switches and various sensors. The network distributes and directs the power generated by the solar arrays through the system. In the entire network, there are six switches for rerouting current through the system. Switches may cause malfunctions within the EPS. Switch errors are fixed by cycling the specific switch. In this simulation, switches errors are randomly generated. Sensors measure the current at various points on the network. In the entire simulator there are four ammeters and a voltmeter. Network losses are disregarded.

The battery. The battery stores the excess electrical power generated by the solar arrays during the day and then releases it in response to nighttime power requirements.

The bus. The bus represents the load on the EPS. In the simulator the bus power requirements can be adjusted depending on power production or system mission schedule.

Time. A pass, or simulated earth orbit, is always 90 minutes, with 60 minutes of it spent in sunlight.

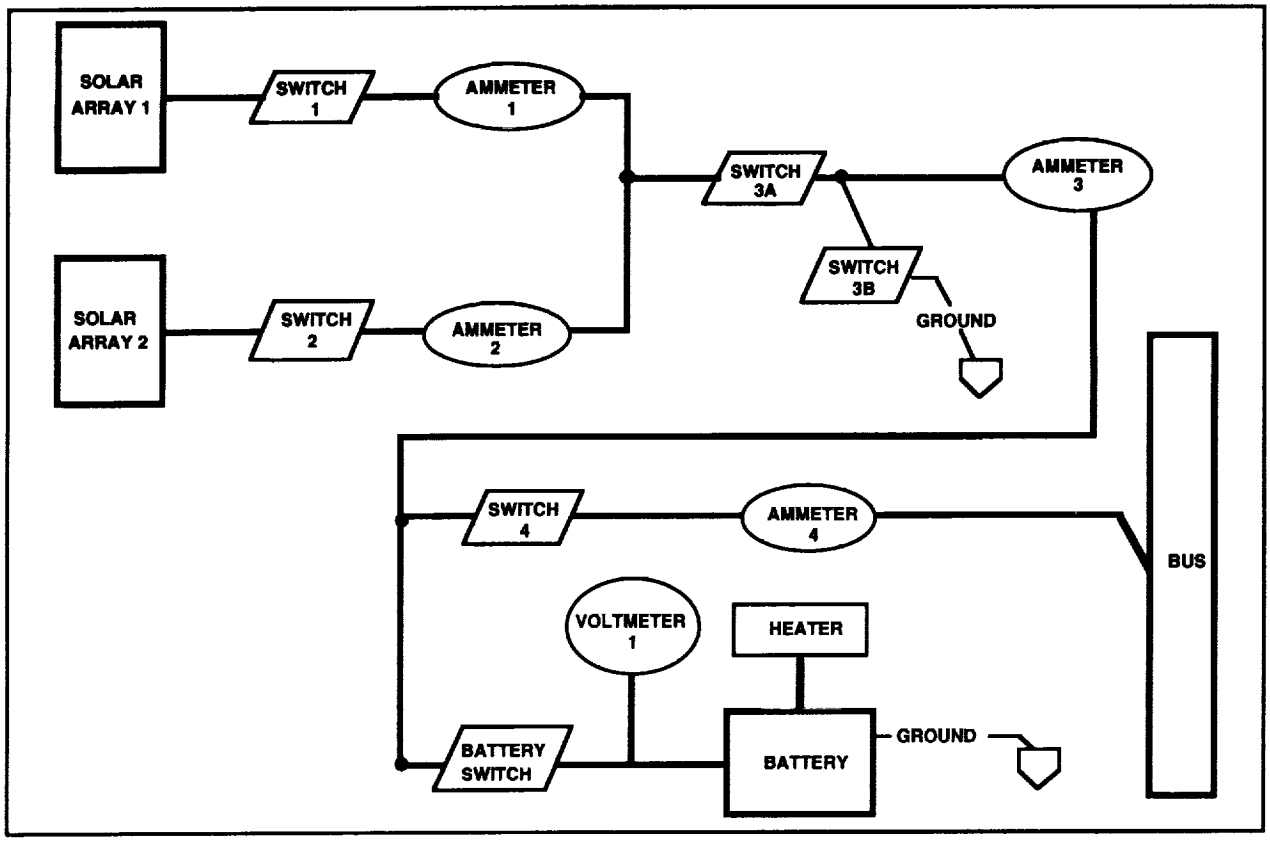

Figure 2. Orbital Satellite Electrical Power System Simulator

4. The Architecture of DISCIPLE-OPS

The intelligent agent, called DISCIPLE-OPS, consists of three main components, the shared knowledge base, the monitoring and repair system, and the multistrategy apprentice learning system, as indicated in Figure 3. The monitoring system uses the shared knowledge base to detect anomalous behaviors of the EPS and to issue repair commands. The learning system extends and corrects the knowledge base as a result of the actions of the monitoring system and the interactions with the human operator.

4.1 The Shared Knowledge Base

The shared knowledge base contains three types of knowledge:

- a hierarchical semantic network representing the electrical power system;
- a set of situation-action rules which detects faults in the EPS and issue repair commands;
- a set of facts representing the current state of the EPS.

A portion of the semantic network from the knowledge base is represented in Figure 4. It consists of a representation of the structure of the EPS, and of the different components of the EPS. This semantic network provides the generalization language for learning.

The knowledge base contains rules of the form:

IF <condition> THEN <action>

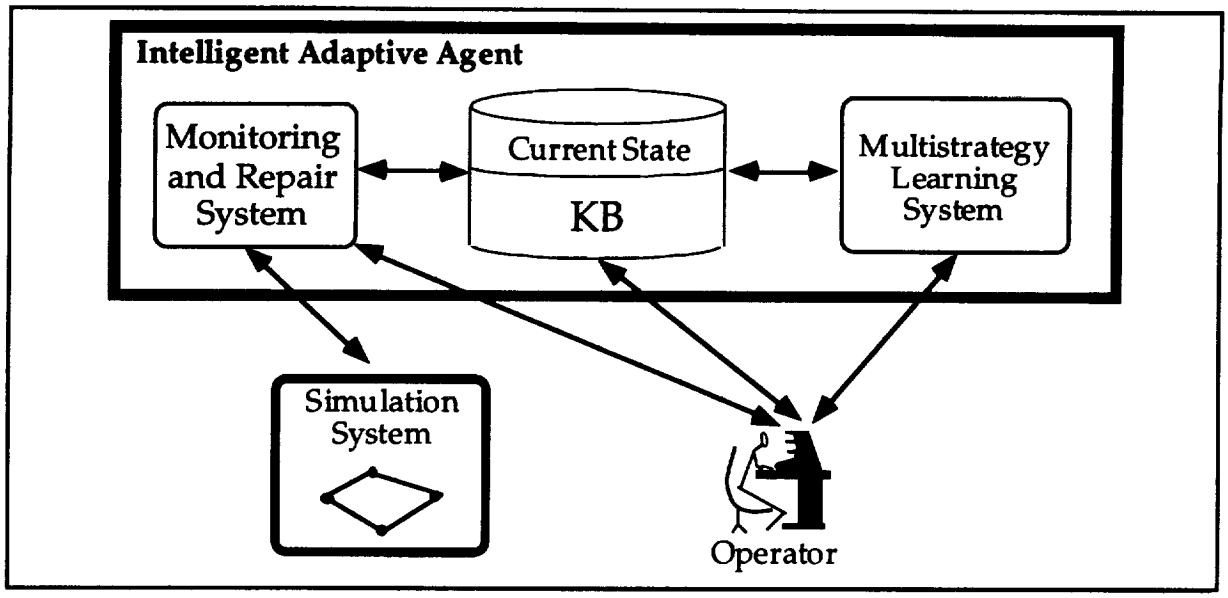

Figure 3. The architecture of the intelligent agent

If the current state of the EPS matches <condition> then the monitoring system will issue the command to perform <action>.

The rules from the knowledge base are learned by the multistrategy apprenticeship learning system from the actions of the human operator. During training many of the rules may not have a single applicability condition, but two conditions, called the plausible upper bound and the plausible lower bound, as it is shown in Figure 5.

The plausible upper bound is supposed to be more general than the exact (but unknown) condition of the rule, and the plausible lower bound is supposed to be less general than the exact condition. The two bounds define a *plausible version space* [Tecuci, 1992] for the exact condition of the rule.

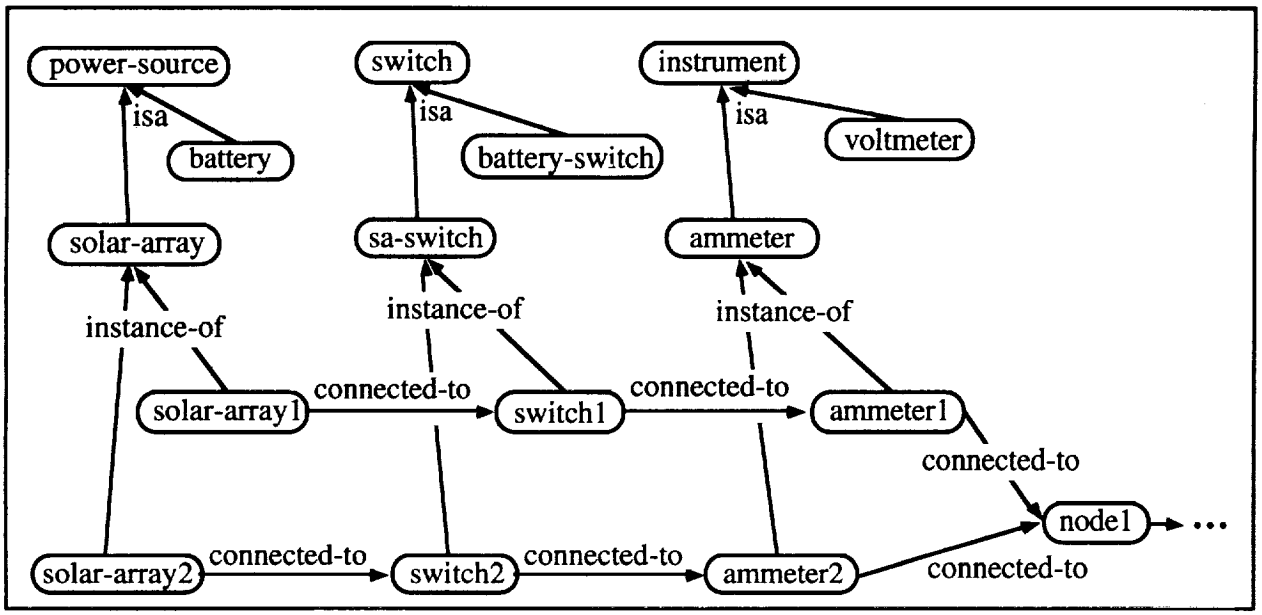

Figure 4. A hierarchical semantic network representing the electrical power system.

```
IF
    plausible upper bound
    (ammeter      a    (reading low))          ; the reading of ammeter 'a' is low
    (clock        c    (time day))             ; during the day
    (switch       sw   (connected-to a)        ; and the switch 'sw'
                       (position open))        ; between the ammeter 'a'
    (power-source sa   (connected-to sw))      ; and the power-source 'sa' is open

    plausible lower bound
    (ammeter      a    (reading low))          ; the reading of ammeter 'a' is low
    (clock        c    (time day))             ; during the day
    (sa-switch    sw   (connected-to a)        ; and the sa-switch 'sw'
                       (position open))        ; between the ammeter 'a'
    (solar-array  sa   (connected-to sw))      ; and the solar-array 'sa' is open
THEN
    CYCLE         sw                           ; cycle switch sw
```

Figure 5. A rule with partially learned conditions.

Each bound is a conjunction of expressions, each expression describing a variable. For instance,

(switch sw (connected-to a) (position closed))

describes 'sw' as being a switch connected to 'a', and being in the 'closed' position. The variable 'a' is described by a different expression from the same bound.

The bounds and the version space are called plausible because they have been initially formed based on an incomplete explanation and its over-generalization (see section 4.3.2).

Also, the learning process takes place in an incomplete representation language that may cause the lower bound to cover some negative examples and the upper bound to fail to cover some positive examples. During learning, the two bounds progressively converge toward the exact applicability condition of the rule. However, due to the incompleteness of the system's knowledge, there is no guarantee that the two bounds will become identical, and therefore equal to the exact applicability condition of the rule. This is not a weakness of the system because it can use the partially learned rules to monitor the EPS system (see section 4.2), and the rules will be continuously improved.

Finally, the current state of the EPS is represented by the readings of the ammeters and the voltmeter, and the states of the switches (open/closed).

4.2 The Monitoring System

The monitoring system is a situation-action production system, in which each rule recognizes a fault type in the EPS and issues the appropriate corrective action.

If the exact condition of a rule matches the current fault state of the EPS then the action from the right-hand side of the rule is called a *routine repair* of the EPS.

Because many of the system's rules are represented as plausible version spaces, the matching process has to take into account the lower and upper bounds of these spaces.

Let us consider, for instance, the following state of the EPS in which the reading of ammeter1 is low during the day, and switch1 is open (see Figure 2).

The plausible lower bound of the rule in Figure 5 matches the current situation because the following expression is true:

```
(ammeter     ammeter1 (reading low))
(clock       clock1   (time day))
(sa-switch   switch1  (connected-to ammeter1)
                      (position open))
(solar-array solar-array1 (connected-to
                           switch1))
```

Because the plausible lower bound of a rule is less general than the exact (but unknown) condition of the rule, the action indicated by the rule (in this case to cycle switch1) is correct. This action will be called a *routine repair* of the EPS.

Let us now consider the case in which the plausible lower bound does not match the current situation. Because this bound is less general than the exact (but unknown) condition of the rule it may still be the case that the exact condition matches the current situation. This can only happen if the plausible upper bound matches the current situation, because this bound is more general than the exact condition. Therefore, if the plausible upper bound of the rule matches the current situation then it is still possible that cycling of switch1 is the appropriate action. This action is an *innovative repair* of the EPS. This repair must be confirmed by the human operator because it is only a plausible solution to the current fault state of the EPS.

Finally, if the plausible upper bound condition of the rule does not match the current situation, then the rule is not applicable.

If no rule applies to the current fault state of the EPS, then the human operator has to indicate a repair action which we call a *creative repair* of the EPS.

One could therefore notice that the plausible version space concept increases system's flexibility in problem solving, allowing it to perform not only deductive reasoning (based on matching exact or plausible lower bound conditions), but also plausible reasoning (based on matching plausible upper bound conditions).

Therefore, depending on which type of rule condition matches the current situation, the monitoring system distinguishes between three types of repairs of the EPS: *routine repair, innovative repair* , and *creative repair.*

4.3 The Multistrategy Apprenticeship Learning System

4.3.1 The learning method

Multistrategy learning is a type of learning which integrates several complementary learning strategies in order to solve more complex learning problems [Michalski and Tecuci, 1994]. Apprenticeship learning is a type of learning from an expert by observing and analyzing its problem solving actions [Mitchell et al., 1985], and is usually based on an interaction with the expert [Tecuci 1988].

DISCIPLE-OPS is both a multistrategy and an apprenticeship learner. A general representation of its learning method is given in Figure 6.

From any creative repair performed by the human operator, DISCIPLE-OPS learns a new situation-action rule which would allow it to make analogous repairs in the future.

First, DISCIPLE-OPS finds an explanation of the creative repair which identifies the important features of the situation. Then, based on this explanation, it defines a plausible version space of a new situation-action rule. This rule is later applied to analogous situations to propose innovative repairs which are accepted or rejected by the human operator.

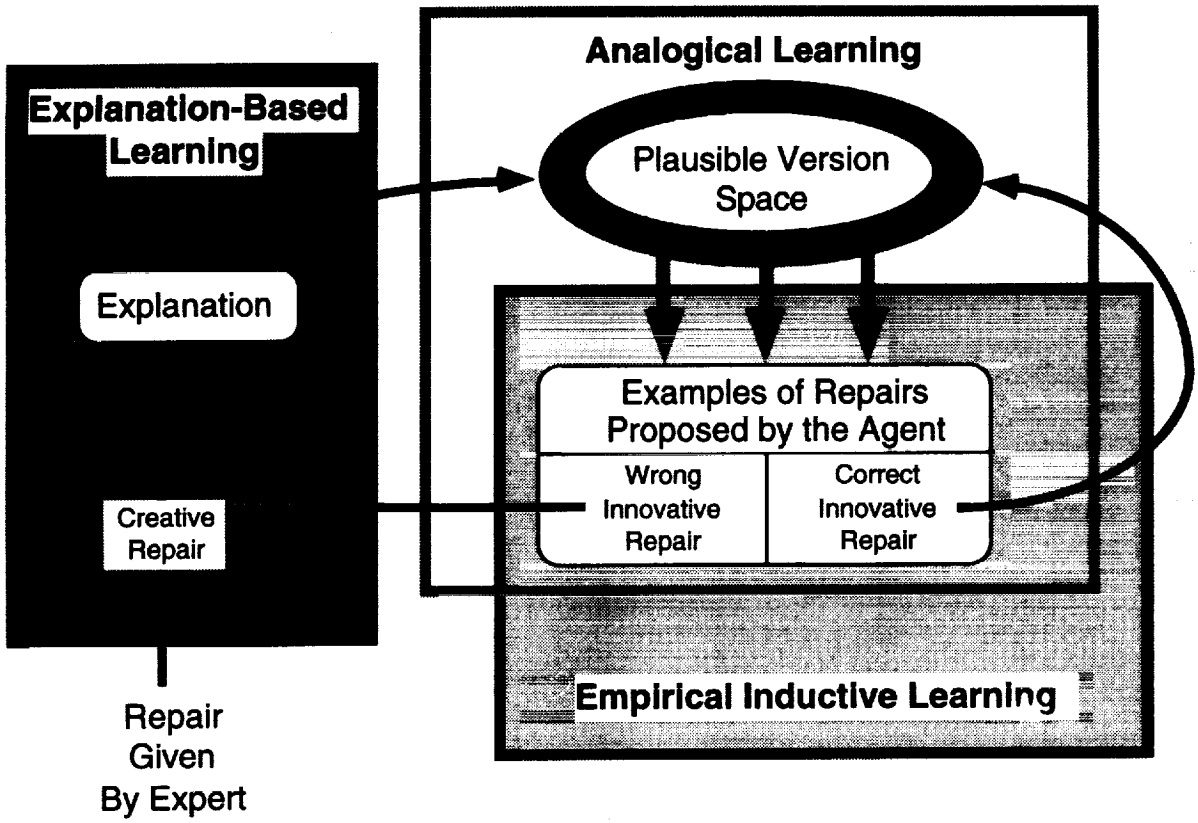

Figure 6. The learning method of DISCIPLE-OPS.

In the case of an innovative repair confirmed by the human operator, the system will generalize the plausible lower bound of the rule so as to cover this repair situation.

In the case of an innovative repair rejected by the human operator, the system will attempt to find an explanation of the failure, and will specialize the plausible upper bound of the rule to no longer cover that situation. In such a situation, the human operator will also have to specify a new creative repair from which the system will learn a new situation-action rule.

The following sections illustrate the different phases of this learning process.

4.3.2 Learning a new rule from a creative repair

Let us consider a state of the EPS for which the human operator proposes the following creative repair:

```
CYCLE switch1
```

First, DISCIPLE-OPS asks the operator to indicate the observations which led to this repair, and receives the following answer:

```
(ammeter1  (reading low))
(clock1    (time day))
(switch1   (position open))
```

Next, DISCIPLE-OPS is trying to find explanations of the fault's cause, in terms of the features and the relationships between the EPS components included in the above observations. It will propose partial pieces of explanations which will have to be accepted or rejected by the operator, as indicated in the following dialog:

```
Choose the relevant explanations of the current failure:

(switch1      (position open)) &
(switch1      (connected-to ammeter1)) &
(ammeter1     (reading low)) ?
   yes
(ammeter1     (reading low)) &
(ammeter1     (connected-to node1)) ?
   no
(solar-array  (connected-to switch1)) &
(switch1      (position open)) ?
   yes
```

The purpose of these explanations is to determine the relevant relationships between the observations and the structure of the network, which will allow the system to recognize similar fault states in the future.

As a result of the above interactions, the following description is identified as characteristic to the current fault state:

```
(ammeter1     (reading low))
(clock1       (time day))
(switch1      (position open)
              (connected-to ammeter1))
(solar-array1 (connected-to switch1))
```

Based on this explanation, DISCIPLE-OPS generates a plausible version space for a new situation-action rule R_i, as indicated in the following.

The plausible lower bound of this rule is just a reformulation of the above explanation, in terms of the variables 'a', 'c', 'sw', and 'sa'. Indeed, these variables can only take the values ammeter1, clock1, switch1, and solar-array1, respectively. Therefore, the lower bound can only match the current fault state (in which it is known that the correct repair is to cycle switch1).

The plausible upper bound is an inductive generalization of the plausible lower bound obtained by turning all the objects into the most general object (called 'something'), turning all the constants to variables, and keeping the relationships between them.

The purpose of the plausible upper bound is to allow the system to propose innovative repairs in future fault states which are similar to the current one. Examples of these cases are presented in sections 4.3.3 and 4.3.4.

```
Ri: IF
      plausible upper bound
      (something   a   (reading x))
      (something   c   (time y))
      (something   sw  (connected-to a)
                       (position z))
      (something   sa  (connected-to sw))

      plausible lower bound
      (ammeter1    a   (reading low))
      (clock1      c   (time day))
      (switch1     sw  (connected-to a)
                       (position open))
      (solar-array1 sa (connected-to sw))
   THEN
      CYCLE        sw
```

4.3.3 Generalizing rules from good innovative repairs

Let us consider a fault state generated by the EPS simulator, characterized by:

```
(ammeter2     (reading low))
(clock1       (time day))
(switch2      (connected-to ammeter2)
              (position open))
(solar-array2 (connected-to switch2))
```

The plausible upper bound of the rule Ri matches this state with the following variable bindings:

```
(a=ammeter2, c=clock1, sw=switch2,
  sa=solar-array2, x=low, y=day, z=open)
```

Therefore the monitoring system proposes the following innovative repair (since the variable sw has been instantiated to switch2:

```
CYCLE switch2
```

Because this repair is accepted by the operator, the plausible lower bound of the rule Ri is generalized *as little as possible* so as to

cover the current situation and to remain less general than the plausible upper bound. The following generalizations are made, based on the generalization hierarchies from Figure 3:

ammeter1, ammeter2 --> ammeter
switch1, switch2 --> switch
solar-array1, solar-array2 --> solar-array

Consequently, rule R_i becomes:

R_i: IF
plausible upper bound
(something a (reading x))
(something c (time y))
(something sw (connected-to a) (position z))
(something sa (connected-to sw))

plausible lower bound
(ammeter a (reading low))
(clock1 c (time day))
(sa-switch sw (connected-to a) (position open))
(solar-array sa (connected-to sw))
THEN
CYCLE sw

4.3.4 Specializing rules from bad innovative repairs

Let us now consider a new fault state generated by the EPS simulator, characterized by:

(ammeter1 (reading low))
(clock1 (time day))
(switch1 (connected-to ammeter1) (position closed))
(solar-array1 (connected-to switch1))

The plausible upper bound of the rule R_i matches this state with the following variable bindings:

(a=ammeter1, c=clock1, sw=switch1, sa=solar-array1, x=low, y=day, z=closed)

Therefore the monitoring system proposes the following innovative repair:

CYCLE switch1

However, this repair is rejected by the operator. In this case, the plausible upper bound of the rule R_i must be specialized *as little as possible* so as to no longer cover the current situation and to remain more general than the plausible lower bound.

In this case, the only possible specialization of the upper bound is to specialize the variable 'z' to the constant 'open'. In general, however, there will be many different ways in which the upper bound could be specialized, and the system would need operator's guidance, as illustrated by the following dialogue:

Compare the fault state in which the correct repair is 'cycle switch1'

(ammeter1 (reading low))
(clock1 (time day))
(switch1 (connected-to ammeter1) (position open))
(solar-array1 (connected-to switch1))

with the current fault state in which the correct repair is not 'cycle switch1'

(ammeter1 (reading low))
(clock1 (time day))
(switch1 (connected-to ammeter1) (position closed))
(solar-array1 (connected-to switch1))

Which are the relevant differences between the current state and the above one?

(switch1 (position open))

Therefore, rule R_i becomes:

R_i: IF
plausible upper bound
(something a (reading x))
(something c (time y))
(something sw (connected-to a) (position open))
(something sa (connected-to sw))

plausible lower bound
(ammeter a (reading low))
(clock1 c (time day))
(sa-switch sw (connected-to a) (position open))
(solar-array sa (connected-to sw))
THEN
CYCLE sw

The operator also indicates that the correct repair is

RESET solar-array1

Consequently, a new rule, R_j, is learned from the current fault state and its repair, as indicated in section 4.3.2:

```
Rj: IF
      plausible upper bound
      (something   a    (reading x))
      (something   c    (time y))
      (something   sw   (connected-to a)
                        (position closed))
      (something   sa   (connected-to sw))

      plausible lower bound
      (ammeter1    a    (reading low))
      (clock1      c    (time day))
      (switch1     sw   (connected-to a)
                        (position closed))
      (solar-array1 sa  (connected-to sw))
   THEN
      RESET        sa
```

Rules are continuously improved in this manner, based on positive and negative examples, generated by the EPS simulator. The learning process decreases the distance between the two plausible bounds. The goal of this process is to make the two bounds identical – at this moment an exact rule is learned. However, because the agent's knowledge is incomplete and partially incorrect, the agent may be unable to learn exact rules and will need to rely on incompletely learned rules, as the one in Figure 5.

4.3.5 Dealing with exceptions

When the agent proposes a routine repair which is rejected by the operator, the corresponding situation-action pair is explicitly associated with the rule, as a covered negative example. Such covered negative examples point to the incompleteness of the agent's knowledge, and are used to guide the elicitation of new concepts and features, by using the knowledge elicitation methods described in (Tecuci & Hieb 1994).

4.4 The monitoring and learning procedure

The procedure in Table 1 summarizes the operation of the intelligent agent.

Table 1. The monitoring and learning procedure.

Monitor:
Let S be the current fault state of the EPS simulator
IF the plausible lower bound of a rule R_i matches S
THEN issue a routine repair command
ELSE IF the plausible upper bound of a rule R_i matches S
THEN issue an innovative repair command
ELSE ask the operator to issue a creative repair command

Learn:
IF the operator agrees with the routine repair proposed
THEN {processing was successful}
ELSE record the current state as an exception of the rule R_i
ask the operator to issue a creative repair command
IF the operator agrees with the innovative repair proposed
THEN generalize the plausible lower bound of R_i to cover the current state of EPS
ELSE specialize the plausible upper bound of R_i to uncover the current state of EPS
ask the operator to issue a creative repair command
IF the operator issued a creative repair command
THEN learn a new rule for the state S and the repair command issued

5 The Methodology for Building a DISCIPLE-OPS Agent

The process of building a DISCIPLE-OPS agent consists of four stages, Knowledge Elicitation, Apprenticeship Learning, Autonomous Learning, and Retraining, as shown in Figure 7. These stages are briefly presented in the following sections.

5.1 Knowledge Elicitation

In the first phase, *Knowledge Elicitation*, the subject matter expert (the human operator) works with a knowledge engineer to define an initial KB which will contain whatever knowledge could be easily expressed by the expert. In the case of the domain considered in this paper, the initial knowledge base consists of a semantic network representing the objects from the EPS (e.g. ammeters, solar-arrays, switches), as well as the structure of the EPS. It will also contain descriptions of the correct states of the EPS, during the day and during the night.

5.2 Apprenticeship Learning

In the second phase, *Apprenticeship Learning*, the agent will learn interactively from the subject matter expert by employing apprenticeship multistrategy learning (Tecuci 1988, 1992, Tecuci et al. 1994), as illustrated in section 4.3. During this phase, the agent's KB is extended and corrected until it becomes complete and correct enough to allow the agent to monitor the EPS autonomously.

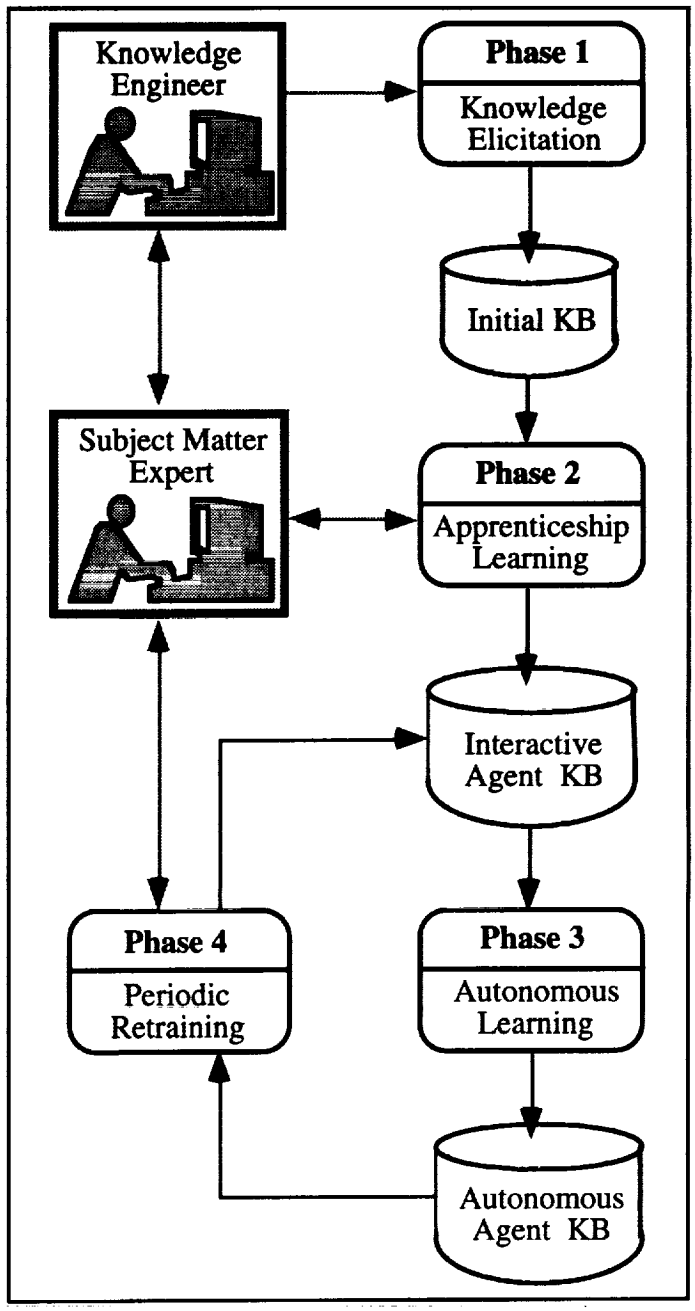

Figure 7. The main stages of building an intelligent adaptive agent.

5.3 Autonomous Learning

When the agent has been trained with examples of the typical problems it should be able to solve, it enters a third phase, *Autonomous Learning*, where it is used to monitor the EPS without the assistance of the subject matter expert. The training received during the Apprenticeship Learning Phase will allow the agent to solve most of the EPS

problems through routine repairs. However, it will also be able to solve unanticipated problems through innovative repairs, and to learn from these experiences, in the same way it learned from the expert. For instance, if the agent issued a successful innovative repair (e.g., applied a rule based on its plausible upper bound condition), it will generalize the lower bound of the rule's condition, to cover the respective situation. If, on the other hand, the agent issued an unsuccessful innovative repair, it will need to specialize the plausible upper bound of the rule. Therefore, the agents developed using this approach will also have the capability of continuously improving themselves during their normal use.

5.4 Retraining

During autonomous learning, the agent accumulates experience and continues to improve its rules. In the same time, it will also accumulate exceptions which correspond to failed routine repairs. After a number of such exceptions have been accumulated, the agent will enter a retraining phase in which it elicits additional knowledge from the operator. Several elicitation procedures which are driven by the goal of eliminating exceptions are described in (Tecuci and Hieb, 1994).

6 Discussion and Future Research

Building intelligent agents is rapidly becoming a major research topic in artificial intelligence (Laird and Rosenbloom 1990; De Raedt et al. 1993; Gordon and Subramanian 1993; Minton 1993; Serge 1993; Van de Velde 1993; Huffman, 1994), due to potential applications of such agents in a variety of domains.

Recently we have been developing a methodology for building intelligent adaptive agents in the framework of our apprenticeship multistrategy learning approach to automated knowledge acquisition (Tecuci 1988; Tecuci and Kodratoff, 1990; Tecuci and Hieb, 1994). This methodology is currently being implemented in the CAPTAIN system (Hille, Hieb, Tecuci, 1994; Tecuci et al., 1994) which is used to build military command agents for distributed interactive simulations.

In this paper we have presented another implementation of our methodology in the DISCIPLE-OPS system which is used to build operator agents. We have also presented initial results on applying DISCIPLE-OPS to build an intelligent adaptive agent to monitor and repair an electrical power system of an orbital satellite.

Our approach to building intelligent adaptive agents which is illustrated by both CAPTAIN and DISCIPLE-OPS has several advantages. Rather than programming their behaviors in a fixed set of procedures or rules, an expert can train the agent as he would train an apprentice. This will result in the agent acquiring a set of rules that govern its behavior. These rules can later be modified in the same manner as the initial training. Another advantage of this approach is that the expert will verify the agent's behavior during training.

Training efficiency is achieved through the use of simple plausible version spaces (Tecuci, 1992) and a human guided heuristic search of these spaces. The plausible version spaces do not suffer from the limitations of the version spaces introduced by (Mitchell 1978). These limitations are:

- the combinatorial explosion of the number of alternative bounds of a version space (there is only one upper bound and one lower bound in the case of a plausible version space);
- the need to have many training examples for the learning process to converge (significantly fewer examples are needed in the case of our method because the expert's explanations identify the relevant features of the examples);

- the use of an exhaustive search of the version space (as opposed to the heuristic search used with plausible version spaces);
- the inability to learn when the representation language is incomplete (as opposed to our method which can learn partially inconsistent rules).

As illustrated in this paper, the use of plausible version spaces also allows a more flexible type of rule matching. Indeed, the agent may perform a limited type of plausible reasoning to address situations that it has not been specifically trained for.

Although this paper shows the potential application of our approach to building an intelligent adaptive agent for monitoring and repair of the electrical power system of an orbital satellite, much work remains to be done until an effective agent is built. Some of the necessary improvements to be performed are the following:

- defining a better representation of the electrical power system which should also include deeper knowledge of the functioning of the EPS;
- developing the explanation capabilities of the agent, so that it can propose more relevant explanations of a given fault situation. Currently, DISCIPLE-OPS uses only domain-independent heuristics for proposing such explanations. There is therefore a need for identifying domain-dependent heuristics.
- developing a domain-dependent method for generating plausible upper bounds of version spaces from explanations of the initial problem solving episodes. Currently DISCIPLE-OPS uses a domain-independent procedure of turning everything except relationships into variables.

However, that fact that DISCIPLE-OPS has been able to efficiently learn rules relying only on a very general representation of the electrical power system and on domain-independent heuristics, indicates that this approach to agent building may be very successful, if the agent will be provided with a better representation of the domain, as well as more specific heuristics for building plausible version spaces.

Future research topics also include:

- development of additional forms of consistency driven elicitation, in order to reduce the burden of explanation of the expert;
- development of more flexible methods of instruction that allows the expert to express whatever instruction is desired at any point in the learning process (Huffman, 1994);
- development of methods manipulating and generalizing numbers since the current implementation is based on a translation between numeric parameters and symbolic parameters;
- further development of the problem solving method based on plausible version spaces;
- integration of experience-based learning into the autonomous portion of building the agent.

References

De Raedt, L., Bleken, E, Coget, V., Ghil, C., Swennen, B. & Bruynooghe, M. (1993). Learning to Survive. *Proceedings of the Second International Workshop on Multistrategy Learning*. Harpers Ferry, West Virginia. 92-106.

Gordon, D. and Subramanian, D. (1993). A Multistrategy Learning Scheme for Agent Knowledge Acquisition. *Informatica*, 17(4), 331-346.

Hieb, M.R. (1990). Machine Discovery in Large Scale Engineering Systems, *M.S. Thesis*, George Washington University, Washington, DC.

Hieb, M.R., Silverman, B.G. and Mezher, T. (1992). Rule Acquisition for Dynamic Engineering Environments. *Heuristics – The Journal of Knowledge Engineering, Winter*, 72-82.

Hille D., Hieb, M.R. & Tecuci, G. (1994). CAPTAIN: Building Agents that Plan and Learn. In *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation.*, May. 411-422.

Huffman, S.B., (1994). Instructable Autonomous Agents. *PhD Thesis.* Department of Computer Science and Engineering. University of Michigan.

Laird J.E. & Rosenbloom P.S. (1990). Integrating Execution Planning and Learning in Soar for External Environments, *Proceedings of AAAI-90.* Boston: AAAI/MIT Press, 1022-1029.

Michalski R.S. and Tecuci G. (Eds). (1994). *Machine Learning: A Multistrategy Approach,* Vol. IV, Morgan Kaufmann, San Mateo.

Minton, S. (Ed) (1993). *Machine Learning Methods for Planning*, Morgan Kaufmann:: San Mateo, CA.

Mitchell, T.M. (1978) Version Spaces: An Approach to Concept Learning, *PhD Thesis,* Stanford University.

Mitchell T.M., Mahadevan S. and Steinberg L.I. (1985), LEAP: A Learning Apprentice System for VLSI Design, in *Proceedings of* IJCAI-85, Los Angeles, Morgan Kaufmann, 573-580.

Serge, A. (1993). Learning how to plan. In W. Van de Velde (Ed.), *Towards Learning Robots.* MIT Press: Cambridge, Mass.

Silverman, B. G., Hieb, M. R., Yang, H. , Wu., L., Truszkowski, W. and Dominy, R. (1989). Investigation of a Simulator-Trained Machine Discovery System for Knowledge Base Management Purposes. Proceedings of IJCAI-89 Workshop on Knowledge Discovery in Databases. Detroit, MI. 327-342.

Tecuci, G. (1988). DISCIPLE: a theory, methodology, and system for learning expert knowledge. *Ph.D. Thesis,* University of Paris-Sud.

Tecuci, G. and Kodratoff Y., (1990) Apprenticeship Learning in Imperfect Theory Domains. In Y. Kodratoff, and R.S. Michalski (Eds), *Machine Learning: An Artificial Intelligence Approach*, Volume III, Morgan Kaufmann.

Tecuci G. (1992). Automating Knowledge Acquisition as Extending, Updating, and Improving a Knowledge Base. *IEEE Transactions on Systems, Man and Cybernetics.* Vol. 22(6), 1444-1460.

Tecuci, G. (1994). An Inference-Based Framework for Multistrategy Learning, In R.S. Michalski & G. Tecuci (Eds), *Machine Learning: A Multistrategy Approach*, Volume 4, Morgan Kaufmann.

Tecuci, G. & Hieb, M.R. (1994). Consistency-driven Knowledge Elicitation: Using a Machine Learning-oriented Knowledge Representation to Integrate Learning and Knowledge Elicitation in NeoDISCIPLE. *Knowledge Acquisition Journal,* Vol. 6(1), 23-46.

Tecuci G. , Kedar S. and Kodratoff, Y., Guest Editors. (1994). *Knowledge Acquisition Journal, Special Issue on the Integration of Machine Learning and Knowledge* Acquisition. 6(2).

Tecuci G., Hieb M.R., Hille D. and Pullen J.M. (1994). Building Adaptive Autonomous Agents for Adversarial Domains. in *Proceedings of the AAAI Fall Symposium on Planning and Learning,* November.

Van de Velde, W. (Ed). (1993). *Towards Learning Robots.* MIT Press: Cambridge, Mass.

omit

Planning, Scheduling, and Control