SIMULATING SATELLITE TELEMETRY

Arthur N. Blasdel, Jr.

Artificial Intelligence Group
Ford Aerospace Corporation
Sunnyvale, California

ABSTRACT

Ford Aerospace Corporation has been investigating the use of intelligent systems to automate space mission support functions since the early 1980's. A product of this research is Paragon, a model-based development environment for intelligent systems. Paragon has been used to develop functional models of satellites that are robust in their abilities to represent satellite behavior. The models have been used to simulate both nominal and anomalous temporal behavior. This paper describes our simulation approach and how the telemetry output from the system can be used during training and rehearsals to provide a closed-loop, interactive response to a wide variety of scenarios.

INTRODUCTION

Space mission operations for both NASA and the DoD are complex and require extensive training before operations personnel can be certified. In addition, regularly scheduled training sessions (rehearsals) are an important part of all space missions, especially for impending launches or other critical maneuvers and situations. To date, dynamically simulating the behavior of ground-based equipment during training or rehearsal has proven to be a manageable problem. However, dynamically simulating the behavior of a space vehicle has proven much more complex. Most space missions are using static data bases for producing limited pre-set space vehicle telemetered parameters during rehearsals. Static data bases produce limited scenarios for training. Also, data bases are cumbersome to maintain, and developing new scenarios becomes time consuming. There exists a need within both NASA and the DoD for a development environment that can create and support software that produces space vehicle telemetry which dynamically and accurately represents all aspects of a vehicle's complex behavior, for both nominal and anomalous scenarios.

Ford Aerospace, Sunnyvale Division, has been investigating the use of intelligent systems for supporting space mission functions since the early 1980s. In our early research, we

applied rule-based systems to problems within the space mission support domain. We found that for large, complex systems, rule bases were difficult to manage, maintain, verify and validate, and limited the use of generic processing (2). The available commercial development environments lacked the flexibility we needed for the space mission support domain. We concluded that model-based systems technology had the potential to overcome the limitations of rule-based systems. Our research then began the transition from rule-based to model-based technology. In the process, we began developing our own model-based environment (1).

This environment, called Paragon, is developing and evolving today and has become central to our research and development of intelligent systems for space mission support. One important focus in our research has been knowledge representation techniques that support dynamic simulation. To date, we have a knowledge representation capability that has successfully created satellite simulation models that support a wide variety of dynamic scenarios. The simulation models are fully commendable, reusable, and respond as either a healthy or unhealthy satellite, depending upon the scenario.

This paper describes how Paragon supports simulation modeling for satellites and discusses how the resulting models can be used during training and rehearsals.

PARAGON MODELING AND SIMULATION APPROACH

The Paragon environment allows a model builder to create a knowledge base of facts, relationships, and behavior descriptions that is a model of the domain. This knowledge base is used to automatically generate a simulation model. Presently, the simulation model is in Lisp code (other languages are possible, but not implemented at this time). The simulation model is separate from both the development environment and the knowledge base. A knowledge base has other intelligent systems applications besides simulation that are beyond the scope of this paper. Figure 1 displays the Paragon environment, with the shaded areas depicting elements that support simulation.

Our knowledge representation approach allows for an intuitive representation of the domain. A model consists of the concepts (physical or non-physical components) that comprise the domain, the appropriate attributes of the concepts (e.g., voltage, temperature, weight, etc.), the relationships with other domain concepts (supplies voltage to, receives current from, etc.), and the behavior of each concept. Component behavior is temporal and is described in terms of the states in which a concept exists, how the concept’s attributes behave while in each state (specified by equation), and what causes a concept to transition from one state to another (specified by logical equation). A modeling example is presented in the next section.

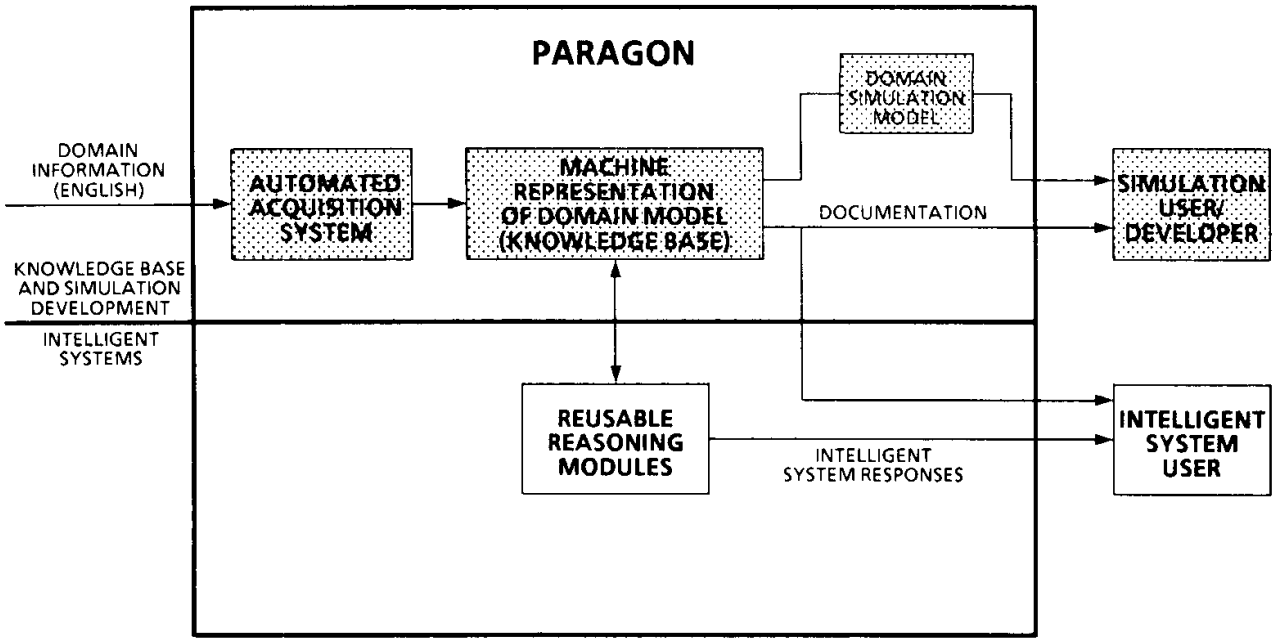

Figure 1. Paragon Environment

The attribute values for the various concepts in the model are potential telemetry measurements. Measurement sensors can be modeled as well as the pulse code modulation equipment that constructs the telemetry wavetrain. However, we have found that simply creating a simulation model that accurately represents the telemetered engineering values is sufficient for creating a robust simulator. The nature and depth required for a simulation model is dependent upon mission training and other system requirements.

Our modeling approach supports full cause/effect behavior through relationships. Cascading behavior across components is inherently represented. This allows the model builder to concentrate on the behavior of individual components within the model and their relationships to other components. Components within the model can be tested individually as the model is being built. As development progresses, subsets of components can be tested. A model is therefore built and tested incrementally. At the system level, complete satellite subsystems can be modeled separately and linked later to form a larger satellite model, with each of the subcomponents of the larger model being fully tested before the full system is integrated.

Model development requires no formal programming skills. A powerful graphics interface is used that is window/menu driven. Options are mouse selectable. The use of keyboard entry is limited to entering concept names, attribute names, state names, etc. Automatic documentation features are also included .

MODELING EXAMPLE

This section provides an overview of the knowledge representation portion of the Paragon environment. We will look at the Electrical Power Subsystem (EPS) for a satellite. First, the EPS in question is composed of many parts, usually called components, and this information or knowledge must be captured in the model. The model builder creates concepts that represent the components and links them in a manner that represents the EPS composition. This representation is termed a composition hierarchy. The model's composition hierarchy is identical to the satellite's component design hierarchy. Figure 2 displays the EPS composition for our example satellite. From this point forward, we will concentrate on one EPS component, the LOAD SHED 1 TIMER, and its relationships to certain other EPS components (see arrows in Figure 2). The design and functionality of the timer is briefly presented next, followed by a description of how knowledge about the timer is represented using Paragon.

The LOAD SHED 1 TIMER receives a solar array current level and a battery discharge signal (via the boost converters) to determine if a condition exists whereby load shedding is warranted to prevent damage to the batteries (excessive discharge). The LOAD SHED 1 TIMER is a twenty minute time out device that activates when the solar array current is greater than 1.2 amps (indicating sun on the solar panels or no eclipse) and the boost converter's current is greater than .15 amps (indicating battery discharge). The timer is designed not to activate during an eclipse (no sun on the solar panels). The timer is an analog voltage device that registers approximately .4 volts when at rest (zero time) and increases at .0036 volts per second when activated. The timer's voltage saturates at a level of 4.4 volts. When saturated, the timer sends a "timed out" signal to the Load Control Unit (LCU). The LCU then automatically sheds level 1 loads (turns off components) to reduce or decrease the undesired battery discharge. This automatic load shedding can only be stopped by ground commanding; that is, the satellite operators in a ground station can send commands to disable the load shedding circuitry in the LCU. The solar array current, boost converter current, load shed timer voltage and load shed circuitry configuration are all telemetered to the ground station.

During the modeling process, the model builder creates concepts for the components on the satellite and defines attributes and relationships for each concept. For example, the BOOST CONVERTERS concept has an attribute named CURRENT, the PCE PWR CTL has the attribute SAC (solar array current) and the LOAD SHED 1 TIMER has the attribute VOLTAGE. Each attribute is given a value class specification (real numbers, symbolic, array, etc.) which defines its legal values. As for relationships, the LOAD SHED 1 TIMER "detects current from" the BOOST CONVERTERS, "detects solar array current from" the PCE PWR CTL and "passes timer status to" the LOAD SHED CIRCUITRY. These concepts, attributes, and relationships are displayed in Figure 3 as they appear in the

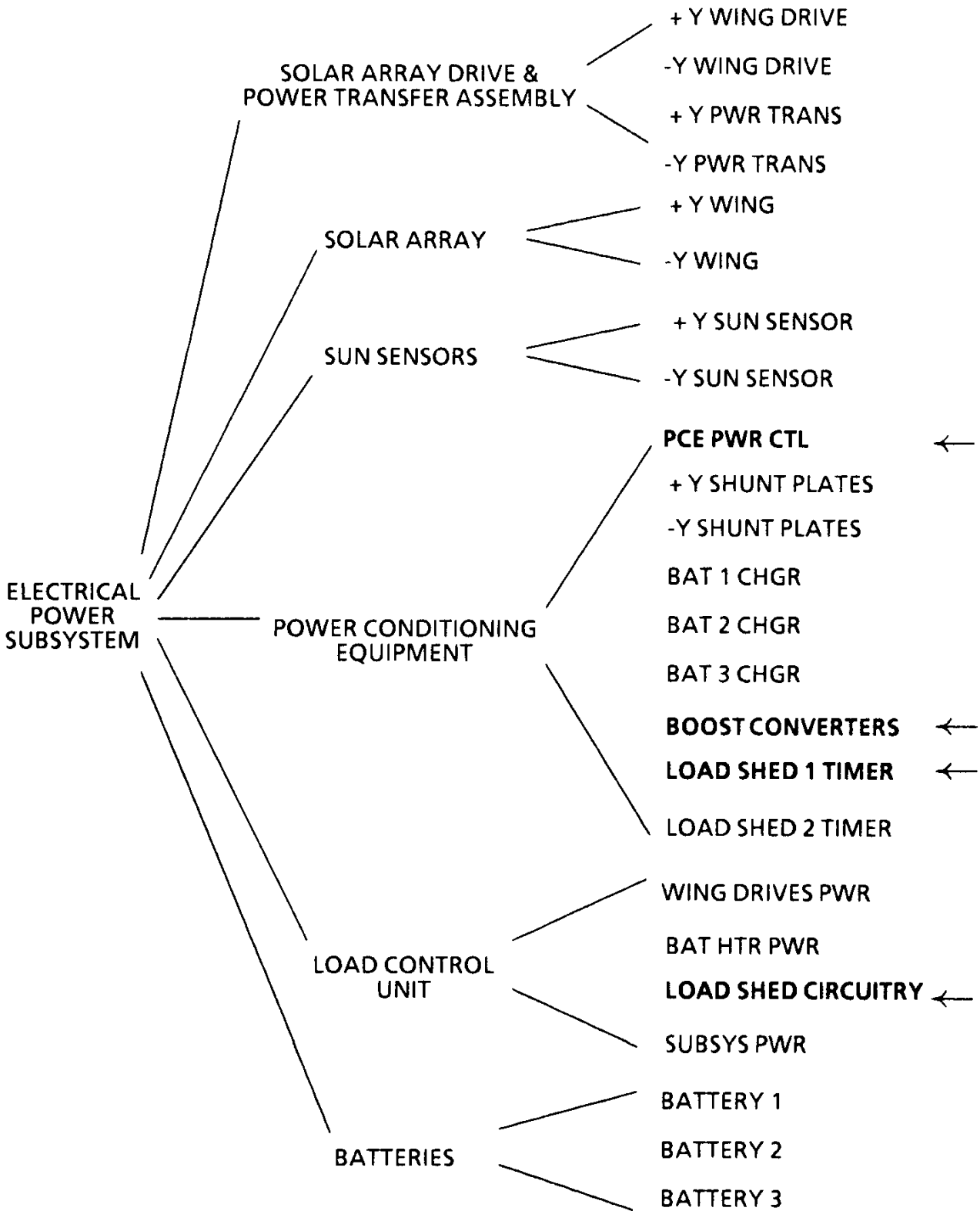

Figure 2. EPS Composition - The model composition hierarchy captures the satellite component design.

Paragon user interface from the perspective of the LOAD SHED 1 TIMER concept. In Figure 3, it can be seen that concepts are linked in a cause/effect (or relationship) hierarchy in addition to the composition hierarchy mentioned earlier. Each concept has both internal attributes and external attribute relationships that allow the model builder to localize the concept's behavior. Before we discuss behavior, we must first discuss classification.

Along with the composition and relationship hierarchies, the model builder also creates a classification hierarchy. This hierarchy allows the description of components (concepts) as members of a class. For example, the three nickel cadmium batteries on the satellite are identical in design (and behavior) and therefore belong to the same class of objects. When concepts are members of the same class, the model builder need only specify the attributes, relationships, and behavior for a single member and the other members inherit these specifications. Figure 3 shows that the LOAD SHED 1 TIMER is a "member of" the class < 20 MINUTE TIMER>. Every < 20 MINUTE TIMER> on the satellite (there can be more than one) has the same relationships, attributes, and behavior. The uniqueness between the various <20 MINUTE TIMER>s is that they can have unique cause/effect links to other concepts in the model, they have different names (other than LOAD SHED 1 TIMER), and they have their own locations in the composition hierarchy.

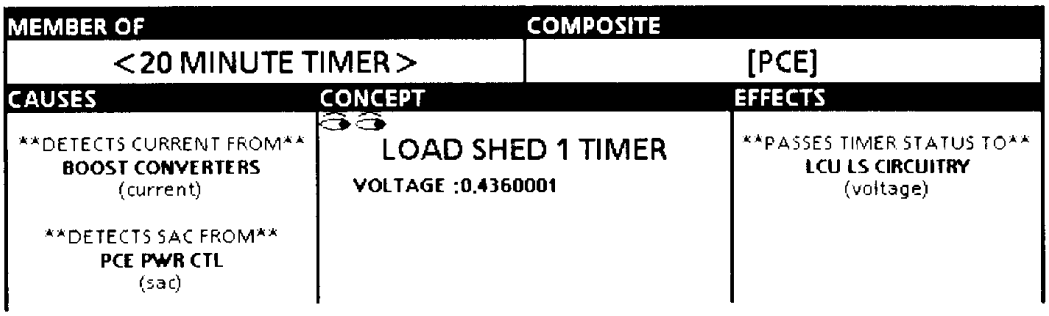

Figure 3. The concept display interface allows the model builder to view a concept's attributes, cause/effect relationships, classification (member of) and composition (composite) representation.

Behavior is temporal in nature and this is captured in the form of a state diagram. Figure 4 displays the state diagram for the LOAD SHED 1 TIMER. This diagram is created to represent the different behavior states that the timer can exist in, and the temporal relationships for transitioning from state to state. The transition conditions must also be specified. For example, the transition conditions from REST to TIME UP and from TIME UP to SATURATED are displayed in Figure 5. When the timer is in the TIME UP state, its voltage behaves according to the equation displayed in Figure 6. The time base is one second and is not displayed. This equation is called an event equation. Attribute behavior is described using event equations for each behavior state that exists within a concept.

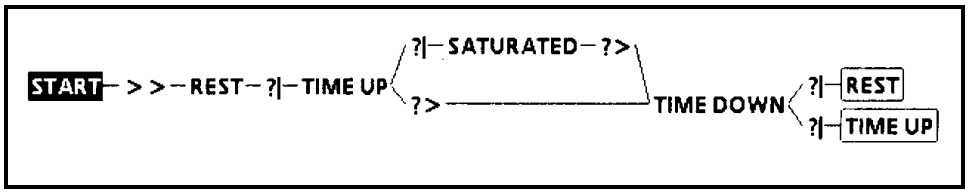

Figure 4. The behavior states for concepts are represented in a temporal state diagram. This diagram is a specification for the LOAD SHED 1 TIMER.

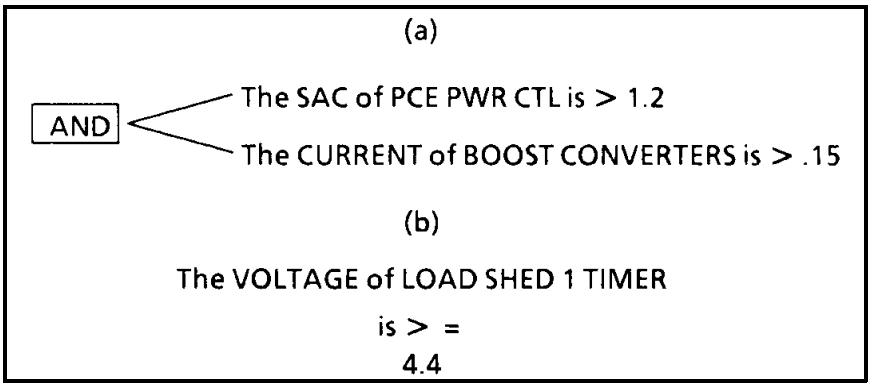

Figure 5. Transition conditions from (a) REST to TIME UP and from (b) TIME UP to SATURATED for the LOAD SHED 1 TIMER.

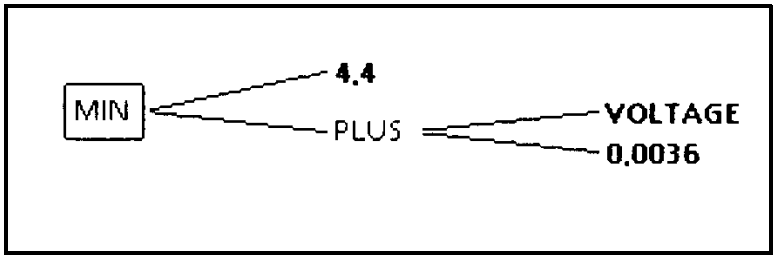

Figure 6. This equation represents the behavior of the attribute VOLTAGE for the concept LOAD SHED 1 TIMER while in the TIME UP state.

Once concept behavior is specified, a simulator option automatically generates the simulation model. The attributes in the simulation model correspond to telemetry measurements. The simulator interface includes RUN, IDLE, STEP, and STOP functions. A capability exists to introduce faults and degradations during a simulation. Faults and degradations can be built into the model and selected when desired, or they can be created before a simulation and deleted later. The model will respond to external inputs such as

satellite operator sent commands or a changing environment. In short, the model builder can create a functional satellite model for a wide variety of complex scenarios. This introduces our next section.

SATELLITE SIMULATION FOR TRAINING AND REHEARSALS

The preceding modeling example can easily be used as part of a training scenario stressing proper operator response during an EPS anomaly. The LOAD SHED 1 TIMER concept can be faulted so that the timer fails and begins timing up without the nominal conditions (boost converter current greater than .15 and solar array current greater than 1.2) being present. During a training session, the satellite operator(s) would view the EPS telemetry measurements, properly diagnose the situation (hopefully), and send the proper commands to reconfigure the load shed circuitry to stop any undesired load shedding. The simulation model would respond dynamically to the commanding, including any incorrect commands. For example, if the load shed circuitry was not reconfigured in time to prevent unwanted load sheds, the load sheds would occur and this would be represented in the "satellite's" telemetry.

The LOAD SHED 1 TIMER model is easily modified to support this training scenario. The model builder would simply add a FAULT MODE attribute to the concept LOAD SHED 1 TIMER. The values for FAULT MODE would be OK (for no fault) or FAILED. The transition condition in Figure 5 (a) would be changed to include the fault. This is shown in Figure 7. At the appropriate point in the training scenario, the attribute FAULT MODE for the concept LOAD SHED 1 TIMER would be set to FAILED by the simulation engineer through the simulation user interface (part of Paragon). From this point forward in the training scenario, the "satellite" would produce anomalous EPS telemetry and respond to any operator commanding according to the modeled fault conditions.

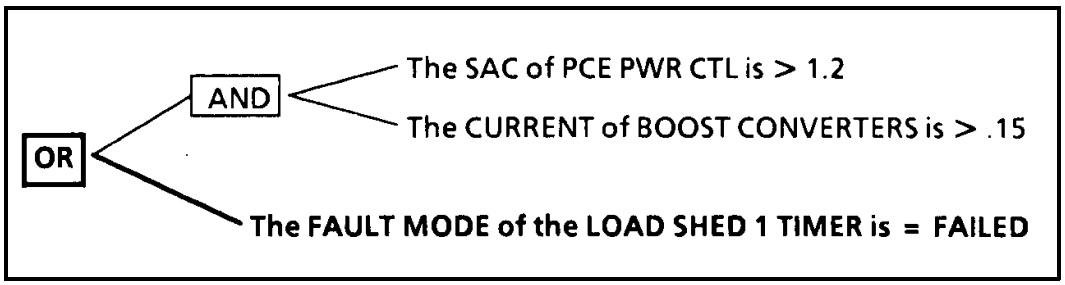

Figure 7. A simple modification in a transition condition can introduce a fault into the model.

Proper operator response during anomalies is only one type of scenario exercised during training. Other scenarios may include executing a satellite performance test or executing a normal maintenance function. Both performance testing and maintenance functions can

require complex planning and precise commanding by the operations personnel and can be supported with our simulation approach.

CONCLUSION

The ideal option for training satellite support personnel is to use actual satellite hardware (or a real satellite) that can be configured to support any desired training scenario. This, of course, is unrealistic for a variety of reasons, the most important being prohibitive cost. The next best option is a software model that acts just like the real thing and can be easily configured to support any desired scenario. Ideally, the simulation software would be easy to build, reusable, easy to maintain, and easy to verify and validate. Model-based representation techniques, like those discussed in this paper, support this type of simulation software.

BIBLIOGRAPHY

(1) Ferguson, J. C., 1986, Beyond Rules: The Next Generation of Expert Systems, Proceedings of the Air Force Workshop on AI Applications for Integrated Diagnostics.

(2) Golden, M., Siemens, R., and Ferguson, J. C., 1986, What's Wrong with Rules, IEEE Westex.