Goal-Based Operations of an Antenna Array for Deep Space Communication

Joshua S. Choi, Adam J. Coleman
Bach X. Bui, Daniel L. Dvorak, Joseph O. Hutcherson,
Michel D. Ingham, Cin-Young Lee, Paul A. Wolgast
Systems & Software Division
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA, USA
{Joshua.S.Choi, Adam.J.Coleman,
Bach.X.Bui, Daniel.L.Dvorak. Joseph.O.Hutcherson,
Michel.D.Ingham, Cin-Young.Lee, Paul.A.Wolgast}@jpl.nasa.gov

Abstract

NASA is currently evaluating the benefits of transitioning to a highly reconfigurable network of arrayed dish antennas to support an increasing number of deep space missions. The next-generation Deep Space Network (NG-DSN), as currently conceptualized, would require extensive automation to reduce operations cost and handle the increased complexity associated with monitoring and controlling the larger number of antennas. This paper presents a prototype operations architecture for the proposed NG-DSN that is fundamentally based on three concepts: physical state variables of the system to be controlled, expressions of operational intent for those state variables ("goals"), and models describing the behavior of these state variables and their interactions. These concepts shape the software design of an automated control system, the model-based systems engineering analysis that feeds this design, and the human operator interface to the control system. This control system provides for automation of capabilities such as resource allocation and fault recovery (both localized and system-wide). This paper describes the development and demonstration of the control system on prototype antenna array hardware at the Jet Propulsion Laboratory.

1. Introduction

Onboard control systems on spacecraft and control systems of the ground communication infrastructure that support the spacecraft are both becoming more and more complex. These increasingly complex systems demand the development of increasingly complex control software, and as such automation, coordination, robustness, and scalability become crucial features. System engineers responsible for such control systems therefore design them so that the *intent* of the users is made explicit during operations. This enables the system to make more "educated" decisions on how and when to execute individual tasks that satisfy both the operator's *goals* as well as the system constraints. Such systems present information to the operator at a higher level of abstraction, so that minimal human intervention is required. This ultimately results in cost savings.

This paper discusses one such control software system that has been designed and developed by a research and technology development team at the Jet Propulsion Laboratory (JPL). The purpose of this prototype system is to monitor and control a large array of deep space communication antennas—potentially such as that of the proposed next-generation Deep Space Network (NG-DSN) (discussed in Section 2)—and as such support simultaneous tracking of multiple spacecraft. A distinguishing characteristic of this prototype system, which we refer to as Array Monitor and Control Prototype (AMCP) in this paper, is that its operation is entirely goal-based—the system behaves autonomously to satisfy the original intent of the operator.

We applied a model-based system engineering methodology called State Analysis (SA) [1] (Section 3) throughout three phases of this software system's life cycle: system engineering (Section 4), state-based

software design shaped by the products from system engineering (Section 5), and system operations. For the operations aspect, we dedicate a section on how we engineered our system to support goal-based operations (Section 6) and how the user interface was designed to complement it (Section 7). We conclude by presenting our results from the operations demonstrations (Section 8.1 on simulated demonstration and 8.2 on demonstration using real hardware), followed by our final thoughts on the results achieved from this project.

Prior to our main discussion, however, we first present some background on the application domain and our motivation for developing the AMCP.

2. Antenna Arrays in NASA's Deep Space Network

The DSN is "an international network of antennas that supports interplanetary spacecraft missions."[2] Dozens of missions today rely solely on the DSN to establish communication links between spacecraft and ground mission control, and each of these links are established at DSN complexes situated around the globe. The problem lies in the fact that each DSN complex is comprised of only a few antennas that have large apertures, and the functional loss of a single antenna can have significant, undesirable impact on the multiple missions scheduled to use it. Each antenna represents a single point of failure for these missions.

NASA is currently evaluating a modernization strategy to address risks such as the one stated above. A potential part of this strategy is the deployment of large arrays of small antennas to replace the deep space communication capabilities of the aging, large aperture antennas. (For the purposes of this paper, we use the term NG-DSN, as mentioned previously, to refer to this proposed antenna array infrastructure.) It is possible to achieve the equal effective aperture of a single, large antenna by combining the signals from a precisely coordinated array of smaller antennas (leveraging the principle of "interferometry") [3]. The current NG-DSN concept is to deploy as many as hundreds of antennas per DSN complex [4]. In addition to the antennas, it is expected that a significant number of heterogeneous DSN subsystems will exist at each complex to support spacecraft communication. Given these attributes—the sheer number of antennas and the variety of subsystems that behave differently—monitoring and controlling of the NG-DSN system would present a difficult challenge. This monitor and control problem is made even more difficult because antenna arrays need to be tightly synchronized and seamlessly coordinated if they are to properly achieve the effective apertures of larger antennas.

Our AMCP software aims to address this complexity issue. By taking a goal-based operations approach, the system adapts and scales regardless of the number of antennas to be used, the number of simultaneous spacecraft to be supported, or the number of different subsystems to be controlled. Because the system makes control decisions to satisfy the operator's goals, it can also autonomously allocate resources and resolve system anomalies.

In the next section, we present the model-based systems engineering methodology that was applied to the development of AMCP.

3. State Analysis Approach to System Engineering

A novel model-based systems engineering methodology, called State Analysis (SA) [1], has been developed to complement the traditional functional decomposition approach and better address the complexity challenge described previously. It provides a methodical and rigorous approach for:

- Modeling physical behavior in terms of system state variables and the relationships between them (*state-based behavioral modeling*);
- Capturing mission objectives in detailed scenarios motivated by operator intent (*goal-directed operations engineering*); and
- Describing the methods by which objectives are achieved (*state-based software engineering*).

The following three sections of the paper describe in more detail the application of each of these facets to the development of AMCP. For detailed information on SA refer to references [1,5,6].

4. Modeling Physical Behaviors

The state-based behavioral modeling aspect of SA begins with the identification of the important state variables in the system. It goes on to describe the causal effects among the state variables, commands and measurements (under both nominal and off-nominal situations) in the form of *state effects models*, *measurement models*, and *command models*. It is important to note that these models may be expressed using any appropriate representation, e.g., differential equations, tables, state charts, pseudo-code, plain text, etc. The granularity of the models is at the discretion of

the system engineer, based on the abstractions and assumptions (s)he defines.

Our models for the AMCP required careful analysis of how the antenna hardware would operate, how the received signal would be processed, how the targets (spacecraft) would be located and tracked, and other physical behaviors of the NG-DSN's system under control. (Our antenna hardware model was actually based on an existing, prototype array antenna at JPL, which we used to demonstrate AMCP's monitor and control capabilities using real hardware—Section 8.2 discusses the result of this demonstration.) The result of our modeling work is graphically summarized in the State Effects Diagram (SED) shown in Figure 1. In this section, we focus the discussion on the region contained within the dashed border, that is, only on the behaviors related to the "mechanical" aspect of the individual antennas including antenna pointing. In the SED, circles represent state variables, triangles represent measurements from the system under control, and inverted triangles represent commands to the system under control.

As can be seen in Figure 1, there are four pertinent state variables in the bounded region: (1) **Ant N Mechanical Health**, (2) **Ant N Mechanical Power and OpMode**, (3) **Ant N Pointing Profile**, and (4) **Ant N Pointing**. The "Ant N" prefix in the names indicates that these state variables exist for each antenna in the system.

The **Ant N Mechanical Health** state variable models the health state of the mechanical components of an antenna. We defined this state variable to have a binary value of **healthy** or **unhealthy.** For the purposes of this prototyping effort, we have chosen to represent uncertainty for this and all other discrete state variables in our system as either **known** or **unknown**, though more refined uncertainty representations (e.g., probabilistic belief states) could be used.

Ant N Mechanical Power and OpMode represents the *power* state and the *operational mode* of an antenna. Therefore, the state variable in fact contains two separate states. The possible operational modes were defined as **manual override** (where the antenna is not under the control of our automated control system) **shutdown** (where the antenna is under our system's control but the motor is powered off), **idle** (where the antenna is under our system's control and the motor is powered on, but is inactive), or **tracking** (where the antenna is under our system's control and the motor is on, and is actively tracking a target according to the pointing profile). The power state is

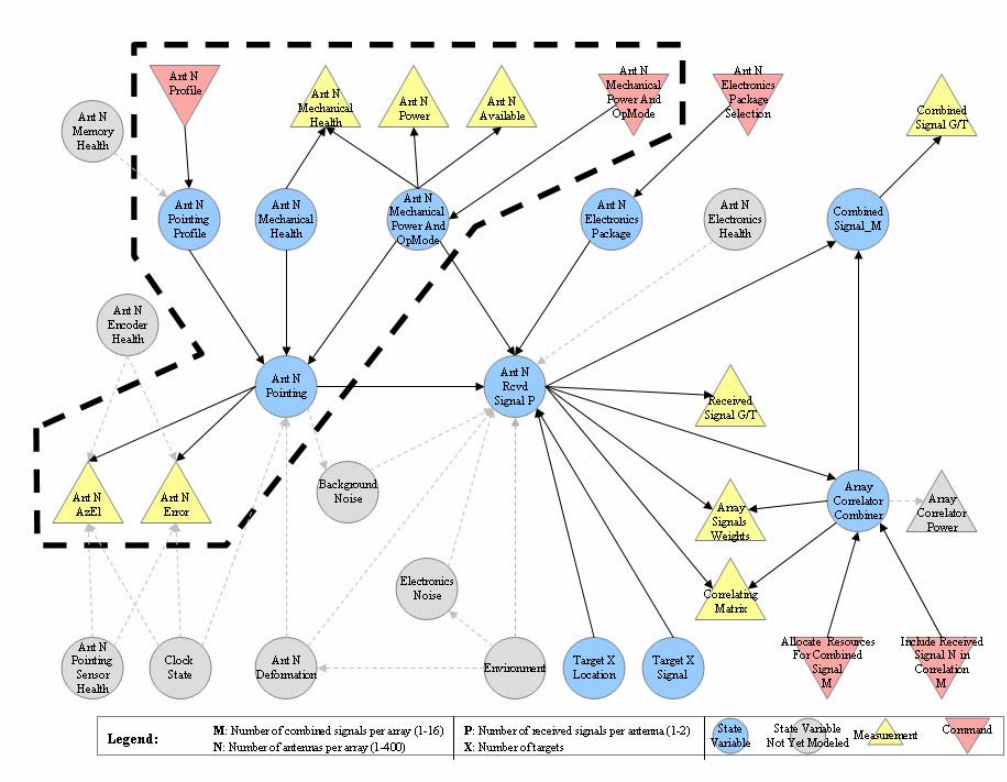

Figure 1. The state effects diagram of the antenna array "system under control"

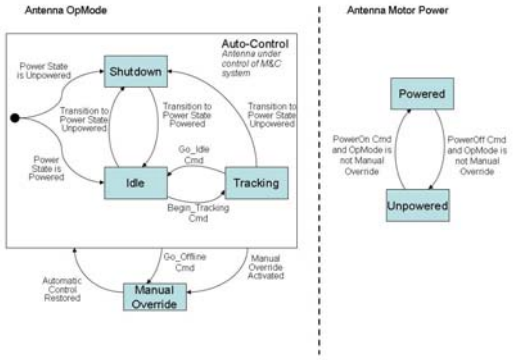

Figure 2. State effects model for the Ant N Mechanical Power and OpMode state variable.

either **powered** or **unpowered.** The state effects model for this state variable is shown in Figure 2.

We must mention that the two state variables introduced thus far model *discrete* states, and they may be modeled as finite state machines, for example. However, we now introduce state variables that model *continuous* states. SA allows both discrete and continuous events to be represented in the models. In fact, SA makes no restrictions on model types, as mentioned above.

Ant N Pointing Profile represents the time-ordered set of coordinates that the antenna should track. The coordinates are specified in azimuth and elevation degrees and are usually the pointing coordinates for a target spacecraft (but not always, as in the case where antennas may be tracking multiple "nearby" spacecraft simultaneously).

Ant N Pointing represents the actual, real-time pointing coordinates of the antenna. Its state effects model is of a continuous nature. For example, assuming that the antenna is **powered**, **tracking**, and **healthy**, **Ant N Pointing**'s coordinates will be:

1. the azimuth and elevation specified by one of the entries in the pointing profile, if the entry's time is the current time;
2. interpolated (or extrapolated) from the pointing profile if the profile does not have an entry with time that corresponds to the current time; or
3. the last pointing coordinates if either: (a) coordinates from 1 or 2 above are physically unrealizable or (b) the pointing profile is an empty list.

Note that this model represents an idealized model of behavior, in that it does not allow for the antenna pointing to stray from the pointing profile, e.g., due to environmental effects or imperfections in the antenna hardware or low-level software design. A higher-fidelity model would include such effects.

State variables can affect each other, but they are also affected by *commands.* For the four state variables under discussion, the **Ant N Profile** command affects the **Ant N Pointing Profile** state variable by causing our system under control to hold a new pointing profile in memory. Similarly, the **Ant N Mechanical Power and OpMode** command affects the state variable of the same name by causing the antenna mechanical state to change (see Figure 2).

State variables, in turn, can affect the *measurements* provided by sensors in the system under control. Intuitively, when a state changes in the physical world, the change may be observed by the control system via one or more measurements. Three of our state variables have an effect on measurements: (1) **Ant N Pointing** affects the **Ant N AzEl** and **Ant N Error** (the as-measured azimuth and elevation coordinates and error values) measurements produced by the antenna system under control, (2) **Ant N Mechanical Power and OpMode** affects the **Ant N Power** and **Ant N Available** (indicates that the antenna may be controlled by our system, i.e., not in manual override mode) measurements, and (3) both **Ant N Mechanical Power and OpMode** and **Ant N Mechanical Health** affect the **Ant N Mechanical Health** measurement (because the health measurement is unavailable when the antenna is unpowered).

We used a State Database [5] implemented in a structured wiki tool to capture the models and descriptions produced from our system engineering process. It allowed our models to be expressed using different representations and supported our highly collaborative engineering environment.

Behavioral modeling can be a painstaking process, often requiring several iterations and corrections. Nevertheless, the time and energy invested in this stage pays dividends because:

1. The models directly inform the software design, which can be approached as an incremental step to the behavioral modeling, rather than a complete design from scratch;
2. The models shape the goal-based operations engineering artifacts, as discussed in Section 6;
3. The behavioral models provide the design for the simulation software.

In the next section, we discuss the first benefit, in the context of the AMCP.

5. Translation into Software Design

Using SA, system engineers use the models of the system under control to specify the control system software design in a manner that allows for direct translation into code. Software engineers are not required to make guesses as to the intent of the system engineers, thus making implementation more straightforward and less prone to errors due to misinterpretation of the system engineering artifacts. This direct translation is possible because our software framework adheres to a specific architecture, as shown in Figure 3. In this section, we describe the AMCP software in terms of the following structures and functions: *goal network, mission planning and execution, goal achievement,* and *simulation.*

The goal network (GN) structure is the embodiment of operator intent. It contains all of the top-level goals and supporting goals necessary to accomplish the task asked of the system by the operator. When a goal comes into the system, it specifies a *constraint* on the value of a state variable over its duration, which is delimited by a starting timepoint and an ending timepoint. The GN also consists of *temporal constraints,* which express min-max constraints between timepoints (e.g., they are used to constrain the duration of a goal, or the relative ordering of goals).

Each GN goes through stages of processing during which it is refined into an executable set of "intent timelines". A hierarchy of goals supporting the operator-specified goals is iteratively produced during the *elaboration* stage, and the resulting set of goals are provided specific, conflict-free relative orderings on the timelines during the *scheduling* stage. Then, the GN is approved for *promotion* once all the goals have been scheduled. Finally, the *execution* of the GN begins. See [6] for more information on each of these processes.

The mission planning and execution (MPE) function is responsible for setting up the GN and managing it. Along with GN elaboration, scheduling, promotion, and execution, MPE also verifies that executing goals are being satisfied, and runs appropriate failure response procedures if a goal failure occurs, as described in Section 6.

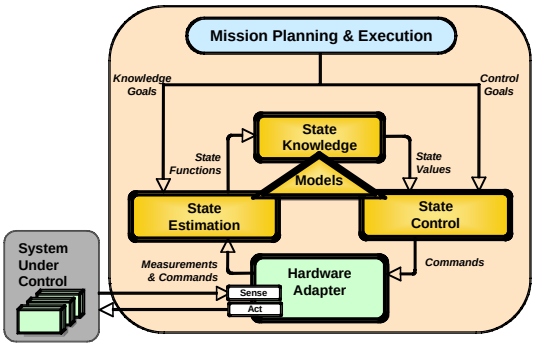

Figure 3. State-based control architecture

The software framework uses components known as *achievers* to ensure that the system behaves as intended. *Goal achievement* is performed by two types of achievers: *controllers* and *estimators*. The architecture in Figure 3 includes the state-based "control diamond" pattern, which implements the goal achievement function in the architecture. For example, when the GN execution process asserts that the '**Transition to Tracking**' goal is to be placed on the **Ant N Mechanical Power and OpMode** state variable, the MPE function issues this goal to this state variable's controller. Upon execution, the controller looks at the constraint specified by that goal. The constraint dictates that the value of the **Ant N Mechanical Power and OpMode** state variable should become **tracking**. The controller checks the current estimated state of this state variable. If the current estimated state is, say, **idle**, then the controller issues the 'Begin_Tracking' command to the Hardware Adapter (HWA). The HWA interprets this command for the system under control and relays the appropriate directive to the hardware. Ideally the hardware will react by placing the antenna in **tracking** mode, as indicated in the state effects model in Figure 2. In the meantime, the HWA retrieves the measurements from the hardware, and forwards them to the estimator. On the basis of this and any other pertinent evidence, the estimator then makes a determination on the state of the hardware. In this case, if the HWA's measurement provides confirmation that the antenna is indeed tracking, then the estimator updates the **Ant N Mechanical Power and OpMode** state variable's operational mode as **tracking**.

At this point the system has made a complete cycle of the control diamond (Figure 3), for the **Ant N Mechanical Power and OpMode** state variable. The other state variables follow a similar pattern, with their achievers' logic tailored appropriately per the models of the system under control.

The actual hardware for the system under control may not always be available, in which case a simulation may be used in its place. Further discussion of our results from operating against hardware simulation is detailed in Section 8.1.

Design documentation of the achievers and HWAs were captured using the same State Database described in Section 4. The software engineers referred to the same models created by the system engineers. Flexible representations supported by the wiki minimized the

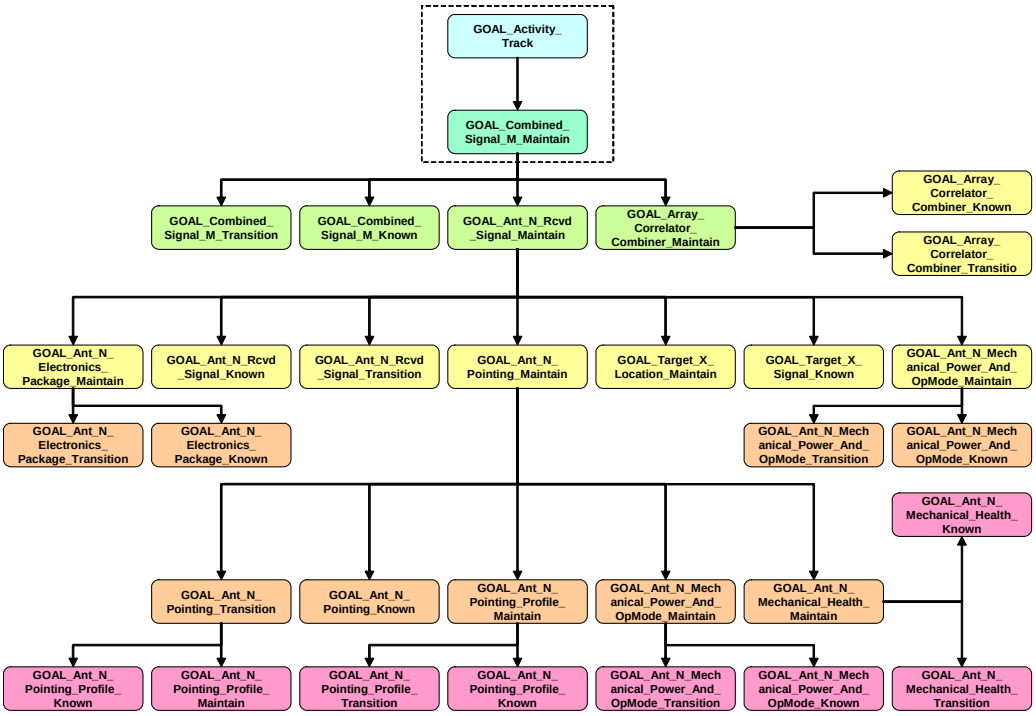

Figure 4. AMCP Goal Elaboration Hierarchy (note: colors indicate the level within the hierarchy).

ambiguity that usually plagues the translation of high-level requirements into software.

6. Engineering Goal-Based Operations

As specified in [1], design of goal-based operations in SA consists of defining: (1) types of goals that can be issued, (2) goal elaborations that specify supporting goals for each goal, and (3) system-specific logic required to correctly plan and execute goals.

Defining goal types is a straightforward endeavor, as each state variable defined in the SA is associated with multiple *maintenance goals* and/or *transition goals*. A maintenance goal, as the name suggests, is a constraint to maintain either the value (control goal) or our level of certainty in the value (knowledge goal). In other words, maintenance goals provide a way to specify what state we want and how trustworthy that state information needs to be. For simplicity, we adopted a binary **known** or **unknown** constraint for all knowledge goals (which is consistent with the uncertainty representations of the state variables, as mentioned Section 4). Transition goals are necessary precursors to maintenance goals that provide constraints in order to "transition" state variables from their current values to the starting values of the maintenance goals. In AMCP, when we place a maintenance goal on an antenna's pointing profile, in most instances the antenna will not initially be oriented in the same direction as the starting point of the pointing profile. A transition goal is then required to take the antenna from its current position to the starting point of the pointing profile.

Goal elaboration is a way to use the state effects model to decompose a goal into supporting goals (see Figure 4). For our project, we translate a service request (consisting of a target, track times, and minimum number of antennas) into a single goal on the combined signal state variable. This goal is then elaborated into supporting goals as predicated by the effects relationships in the SED. Each supporting goal is in turn elaborated until there are no more supporting goals to be specified. Guidelines for how to define goal elaborations are outlined in [1], including "a control goal on a given state variable may elaborate into a supporting knowledge goal on the same state variable and supporting control goals on its affecting state variables", and "a maintenance goal should elaborate into a transition goal on the same state variable".

In cases where there is more than one set of supporting goals that can be specified for a given goal, elaboration *tactics* capture these options, and define the logic for selecting the appropriate supporting goals to achieve the original goal, given the current system

state. For example, in the elaboration of our combined signal goal, antennas are chosen based on availability and health. This corresponds to creating received signal goals on the assigned antennas. Such tactics provide flexibility in achieving goals under different system conditions and can also be used to recover from system faults. In our control framework, when a goal fails, responsibility for handling of the failure is passed up the elaboration hierarchy until a goal elaborator is found to have a tactic that helps to resolve the failure (somewhat akin to exception handling). In our system, when an antenna fails for any reason, the combined signal goal is re-elaborated using the following tactic: replace the signals from any unhealthy antennas with signals from available and healthy antennas, if possible.

The complete goal elaboration tree for the AMCP is shown in Figure 4. These elaborations were specified using the SED in Figure 1. Central to our elaboration hierarchy is the multiplicity of goals generated from the combined signal goal. The combined signal goal elaborates into multiple goals on received signals, which corresponds to multiple assigned antennas, as denoted by the "Ant_N" prefix in a goal's name.

The third aspect of designing goal-based operations is the specification of system-specific logic required to correctly plan and execute goals. For example, systems engineers need to specify the logic that dictates whether two goals may be executed concurrently or sequentially on a given state variable's timeline. More detail on how to specify such logic is provided in [1].

7. User Interface for Operations

The user interface (UI) is built upon the Eclipse Rich Client Platform, Java Message Service (JMS), and Relational Database Management System (RDBMS). Leveraging Eclipse and JMS, the UI supports monitoring from multiple perspectives (e.g., state estimates and measurements). In near real-time, the operators can monitor service requests in both mission-centric and antenna-centric views. The service requests and service fulfillment information are rendered in a Gantt chart that affords the operators a bird's-eye view of all progress. A tabular view contains detailed service fulfillment information and is linked to the Gantt chart. This provides the operator the ability to drill down to needed information should human decision and intervention be required. The UI also provides operators the ability to correlate the antenna profile, pointing commands and the actual pointing measurements into a stereo plot. These graphs are useful to quickly diagnose antenna-related problems, especially in an arraying scenario. Finally, the RDBMS allows the UI to record all data generated as a response to a service request. All discrete information items are assigned globally unique IDs and automatically converted to relational records for storage. This persistent aspect of the UI allows data mining for history and trends.

8. Operations Demonstration

In this section, we present results from demonstrations of the AMCP.

8.1. Simulated Antenna Array

An actual NG-DSN array consisting of hundreds of antennas does not exist and is only conceptual at this time. Furthermore, tests on real antenna hardware are costly due to staffing of antenna personnel and maintenance costs. So in order to demonstrate our system's fully scalable capabilities, we interfaced our control system to a software-simulated antenna array.

In the AMCP system under control, each antenna was modeled per the state variables given in Section 4. A simulator was written for each of these state variables. The simulators were modeled after the expected behavior of the actual antenna hardware components, as captured in the state effects models. The simulated hardware elements served as the system under control shown in Figure 3. The simulators were also capable of off-nominal behavior (e.g., antenna failure) in order to test our control system's fault recovery mechanisms.

Our tests typically involved from five to fifty simulated antennas. A common test case had the operator issue a service request to track a spacecraft using five antennas. Once all five antennas were "on point", or actively tracking the spacecraft, a failure would be injected into one of the antennas. We then verified that our control system detected the failed antenna, and brought up a new available antenna to take its place. Various permutations of this test case were performed.

One of the main advantages of autonomous control of the AMCP is the ability to monitor and control multiple concurrent activities with minimal operator intervention. For this reason, a suitable test was to have the operator submit multiple service requests for multiple target spacecraft. After doing so, the control system allocated the necessary number of antennas to each target spacecraft, and proceeded to track each target. For example, five antennas pointed to one

spacecraft, and six antennas to another, and four antennas to another, and so on.

Although testing using the simulated antenna array was essential and very productive, the viability of our control system was demonstrated even further when interfaced to actual antennas, as described in the next section.

8.2. Antenna Array Hardware

During the development of AMCP, a separate team at JPL was prototyping the viability of the proposed NG-DSN concept. Their breadboard setup included two 6-meter dish antennas. We decided to demonstrate our control system on these actual antennas.

These antennas were already being monitored and controlled using software that the aforementioned array prototype team had developed. We analyzed their software, broke it down into application layers, and identified appropriate interface points for our AMCP software.

During these hardware-based demonstrations, our AMCP software behaved as expected. Upon submitting a service request through the UI (the same ones used in the simulated environment), we observed the actual hardware antenna slewing and tracking according to the pointing profile of the target. We simulated antenna faults by killing the UNIX processes responsible for tunneling the measurements from the antenna hardware to our control software's HWA—thus creating a situation where the antenna(s) in question stopped providing measurements. Fault-handling (goal re-elaboration and re-scheduling) occurred immediately, and the expected robustness of our control system was verified.

9. Conclusion

We have successfully applied the State Analysis systems engineering methodology and utilized a state-based control framework to produce and demonstrate a robust and scalable monitor and control software system for the proposed NG-DSN. This approach yielded several benefits:

- Direct mapping from systems engineering specifications to software design minimized errors of translation and omission;
- Goal-directed operations engineering allowed our system to respond to faults in an intelligent manner—trying different methods to ultimately satisfy the user's intentions; and
- Use of a structured wiki State Database to capture our models promoted a high level of collaboration within the team, and communication of the requirements was straight-forward.

Future efforts will focus on infusion of this technology into other service-oriented DSN systems, not limited to the proposed NG-DSN, as well as other embedded system applications.

10. Acknowledgments

The work described in this publication was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

The authors gratefully acknowledge the invaluable contributions of past members of the project team: Ohanes Dadian, Margaret Stringfellow Herring, Mark Indictor, and Jay Torres. We would also like to thank Carl Miyatake and Barzia Tehrani for their help with the prototype array's antenna hardware interface.

11. References

[1] Ingham, M., Rasmussen, R., Bennett, M., and Moncada, A., "Engineering Complex Embedded Systems with State Analysis and the Mission Data System", *AIAA Journal of Aerospace Computing, Information and Communication*, Vol. 2, No. 12, Dec. 2005, pp. 507-536.

[2] http://deepspace.jpl.nasa.gov/dsn/

[3] Rogstad, D.H., Mileant, A., and Pham, T.T., *Antenna Arraying Techniques in the Deep Space Network*, John Wiley & Sons, Inc., 2005.

[4] Bagri, D.S., Statman, J.I., and Gatti, M.S., "Operations Concept for Array-based Deep Space Network," IEEE Aerospace Conference, Big Sky, MT, March 2005.

[5] Bennett, M., Rasmussen, R., and Ingham, M., "State Analysis Requirements Database for Engineering Complex Embedded Systems", 15th Annual International INCOSE Symposium, Rochester, NY, July 2005.

[6] Dvorak, D.L., Ingham, M.D., Morris, J. R., and Gersh, J., "Goal-Based Operations: An Overview", AIAA Infotech@Aerospace Conference, Rohnert Park, CA, May 2007.