AI Magazine Volume 14 Number 2 (1993) (© AAAI)

AI Research and Application Development at Boeing's Huntsville Laboratories

Steve Tanner and Ray Carnes, with George Williams, Mike Brady, Al Underbrink, Rodney Daughtrey, Kellie Miller, Jeff Jackson, Amber Olson, Frank Craig, Cindy Kamhieh, David Throop, Randy Fennel, Alan Johnson, Ronnie Long, Mike Fouche, Jill Nordwall, Bill Davis, Dannie Cutts, and Ethan Scarl

■ This article contains an overview of recent and ongoing projects at Boeing's Huntsville Advanced Computing Group (ACG). In addition, it contains an overview of some of the work being conducted by Boeing's Advanced Civil Space Systems Group. One aspect of ACG's charter is to support the efforts of other groups at Boeing. Thus, AI is not considered a stand-alone field but, instead, is considered an area that can be used to find both long- and short-term solutions for Boeing and its customers. All the projects listed here represent a team effort on the part of both ACG researchers and members of other Boeing organizations.

The Advanced Computing Group (ACG) was established at Boeing's Huntsville laboratories in 1985. Its goal was to act as a satellite laboratory for the Advanced Technology Center (now Computer Science) at the Seattle facility. It represented an effort to facilitate AI technology transfer from the Seattle laboratory to the Huntsville facilities. This satellite center was to support local Boeing organizations and their customers (primarily the National Aeronautics and Space Administration [NASA] and the Department of Defense [DoD]). Since this time, ACG has become a separate group under the direction of the local Boeing Defense and Space Organization. ACG supports between 15 and 20 full-time researchers and uses expertise from many other groups and subcontractors, whenever it is justified for a given project. ACG maintains a variety of computer equipment and links into Boeing's rather large internal network. ACG's goals remain firmly planted in the pursuit of basic research and the support of technology transfer.

The following sections describe some of the major projects currently in progress or recently completed at the Boeing Huntsville facilities. For the most part, they are in no particular order, although an attempt was made to group them based on the major technologies being explored. The primary groupings are (1) planning, scheduling, and optimization; (2) knowledge-based and expert systems; (3) neural networks; (4) user interfaces and virtual reality; (5) model-based reasoning; and (6) other projects. Several projects were omitted from this article because of their proprietary nature; space requirements; or, simply, a lack of time.

… even small increases in efficiency translate into huge savings …

Planning, Scheduling, and Optimization

Because many of the problems that our customers face deal with trying to work with oversubscribed resources, the areas of planning, scheduling, and optimization are of particular interest to us. We are often working with portions of systems that ultimately cost billions of dollars. For this reason, even small increases in efficiency translate into huge savings. Included here are two of the projects we've undertaken to help deal with such problems.

Operations Automation

This project is an ambitious, long-term effort that encompasses many research areas. The general problems addressed deal with supporting Space Station *Freedom* and its need to continuously perform a wide range of activities over its 30-year life cycle. Crew time will always be a scarce resource on such complex spacecraft, and all intravehicular activity (IVA) is expensive. Even if only a few of the targeted activities can be automated, a huge increase in productivity and reduction in costs can be achieved.

Great potential exists in the areas of spacecraft housekeeping, maintenance, and science assistance. It doesn't make much sense to select well-educated, highly skilled, highly trained crew members and then send them into orbit to do nothing but clean toilets, stock supplies, and change air filters. IVA robots and automated devices can perform such routine activities, freeing the crew to perform more demanding tasks. However, the transition of work load from crew members to automated systems must be accomplished in a safe, consistent, and robust manner. Successful automation will potentially save billions of dollars throughout the life of the space station and on upcoming Lunar and Mars endeavors and will continue to forge a path for performing space operations that are otherwise impossible because of the lack of personnel.

Current investigations for spacecraft automation and robotics center on an integrated test bed for generat-

ing plans for spacecraft operations and validating and monitoring their execution by crew members, robots, and software systems. Specific areas of development include (1) automated planning, (2) automated explanation, (3) dexterous robotics, and (4) plan diagnosis.

The automated planning activities form the basis of this project research. The planner is based on James Allen's (1983) temporal logic, with some extensions. Plan actions are selected, ordered in a nonlinear fashion, and decomposed into lower-level actions that achieve the system goals. The final plan is consistent with temporal constraints imposed on the planning environment.

To maintain temporal consistency, a computationally intensive algorithm must be executed, checking the temporal constraints transitively. Upon decomposition, a short sequence of high-level actions can become dozens (or even hundreds) of atomic actions. An algorithm to cluster temporal relationships in an abstraction hierarchy using reference intervals was developed to reduce the expense of the transitivity computation (Davis and Carnes 1991). The technique clusters and maintains the reference hierarchy and posts constraints across hierarchical boundaries.

The planning environment is composed of object models of the tasks to be performed, the items to be manipulated, and the agents to perform the tasks (that is, crew members or robots). Task descriptions identify which objects are affected by action, including goals achieved and any side-effects. Agent models describe the particular capabilities an agent can have, and object models specify the manner in which objects are affected by actions.

A major benefit of the object models is the ability to perform agent-independent planning (Davis 1990). *Agent-independent planning* is the generation of activity plans without regard to the particular agent that will perform the plan. Planning proceeds from the standpoint that the task sequence is the same (at an abstract level), independent of the performing agent's capabilities. This technique simplifies the planning process by reducing the number of constraints to be considered, in effect implementing a form of constraint propagation. Once the agent-independent plan is generated, agent constraints are drawn into the plan for further processing. This two-step approach helps simplify the planning process and enables the ability to explore plan execution by different agents.

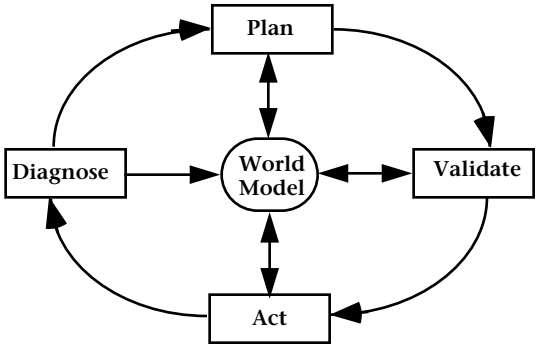

Figure 1. Autonomous Control System Architecture.

The planner is part of an autonomous control system for a space station IVA robot (Carnes and Underbrink 1991). The other components of the control system are a *validation agent* (a system that uses explanation techniques when a human is the executing agent), an *actuation agent* (for example, a crew member or a robot), and an *evaluation agent* (a monitoring and diagnosing system). The components share a world model that is composed of the agent, the object, and task descriptions (figure 1).

These systems work effectively to control a hardware system and robotic manipulator without human intervention. This autonomous control system is a major step toward increasing man's productivity in space. As the Intelsat rescue and repair demonstrated in May 1992, the presence of humans in space is sometimes essential. However, robotic and automated systems can substantially relieve the work load on spacecraft crew members.

Contact: Al Underbrink

The Automated Logistics Element Planning System

Space Station *Freedom* will begin assembly in 1995. Approximately every 90 days, a space shuttle will transport cargo carriers from earth, packed with experiment payloads, crew supplies, spare parts, and other items essential to maintaining station operations. The primary cargo carrier, the pressurized logistics module (PLM), is about the size of a small school bus. The floor, ceiling, and sides of PLM are lined with several smaller modular carriers, called *racks*, that contain either scientific experiments or a set of variable-sized individual stowage drawers. The task of determining what cargo to take, when to take it, where to place it, and how and when to return it to earth is known as *load planning*.

At the highest level, the load-planning task has one main requirement: to ensure that needed supplies, spare parts, and experiments are transported to and from the station in a timely manner. This task must be accomplished while many additional interdependent, possibly conflicting constraints are satisfied. Examples of some of these constraints are shown in table 1.

In addition to satisfying these constraints, significant cost savings or increased throughput can be achieved through the judicious use and optimization of critical resources (for example, available stowage volume or weight capacity). If stowage volume use can be increased by 5 percent, the increase in the amount of used stowage volume over the station's 30-year lifetime would be equivalent to the volume of 4.7 PLMs—volume that could be used for transporting additional experiments, payloads, and other supplies to the station.

To help satisfy load-planning requirements, ACG is teaming up with the Boeing Launch Operations

and Resupply Group to build the automated logistics element planning system (ALEPS). The system is a comprehensive, multiuser planning and automation tool designed to ease the complex load-planning task.

ALEPS is designed to support the concurrent development and smooth integration of load plans. The system automatically integrates resource use and physical characteristics across all levels of the packing hierarchy. In addition to these semiautomated facilities, ACG has developed and continues to enhance a suite of algorithms that are designed to perform a variety of load-planning tasks. Algorithms developed so far use a variety of techniques to arrive at solutions, including simulated annealing (Daughtrey, Fennel, and Schwaab 1991), heuristically guided search schemes, and simple greedy approaches. Some basic research into problem size estimation and reliable convergence to near-optimal solutions has also been done. In many test cases, the simulated annealing and heuristically guided algorithms have outperformed human packers working on the same load-planning task, producing higher-quality solutions in significantly less time.

Contact: Rodney Daughtrey

Constraint Type	**Example**
Cargo Item Priority	Food supplies are more critical than personal items
Functional Grouping	Store all food supplies in the same area
Physical Cargo Item Properties	Fragility, magnetic sensitivity
Center of Gravity	Packed drawers, racks and modules

Table 1. Example Constraints on Solutions for the Automated Logistics Element Planning System.

Knowledge-Based and Expert Systems

ACG has developed and delivered into production several expert systems. A few of them are briefly discussed here. For the most part, these systems are all straightforward rule-based expert systems in which we use as many off-the-shelf tools as possible. We have found that maintenance of the fielded systems is a problem. With this fact in mind, we developed a three-tiered approach to expert system development (Craig et al. 1990). The three design areas are (1) the user interface, (2) the rule base, and (3) the database. In general, as much of an expert system as possible is pushed into either the user interface or the database, yielding a system that is easier to maintain. Far more people know how to deal with a database or a user interface tool than know how to maintain a rule base.

One of the results of many of our expert system efforts has been to simply codify, in a consistent and usable manner, the rules involved in a given project. Many times, this coding process is far more important than the expert system itself. By the time we put together a rule base, we have mapped out the process flow of a given task, which, in turn, generally leads to significant improvements in the process itself, even before the system is in place. Expert systems are seen as a tool for both significant cost savings and process improvement.

The Rack Equipment Placement and Optimization System Expert System

Space Station *Freedom* comprises several modules, including the Habitation, the U.S. Laboratory, the European Laboratory, and the Japanese Experiment Module. Each of these modules contains many racks, and each of these racks can hold one or more rack equipment items.

The placement of these rack equipment items is a complicated process of studying the effects of potential placement layouts based on a large and diverse set of physical, functional, and operational constraints. The REPO system allows a user to study a set of rack equipment and examine where to integrate the individual items (Tanner and Fennel 1991). The system helps the user by generating a set of potential locations in which a chosen piece of equipment would best fit based upon the represented constraints. The final system also generates a near-optimal layout of all the rack equipment based on these constraints.

Contact: Steve Tanner

Network Analysis and Help-Desk Assistant Expert System

The End User Services Group within Boeing's Huntsville branch must enhance, maintain, and repair the computer communications network that encompasses the Boeing Huntsville facilities, local leased buildings, Boeing equipment used on location at NASA Marshall facilities, and other Boeing sites outside Alabama. Additional duties include problem diagnosis and repair of hardware and software. The performance of these tasks requires experience with a wide range of communication protocols, hardware devices, software languages, and problem diagnosis and repair procedures. The use of an expert system was viewed as a way to address most of these problems.

This system provides functions in two key areas: First, when experts are trying to solve a problem with the network, they often need to know the network equipment configuration and, in particular, the connectivity between two nodes on the network. A dynamically created graphic display is able to show the user this information. The user can also display diagnostic procedures and information about the specific types of equipment.

Second, the system can be used as a help-desk assistant. In this role, the system responds to information entered by a help-desk analyst by offering the analyst a procedure to follow for the diagnosis of the problem being considered. The session is also stored in a file so that the information can be used if in the future we incorporate case-based reasoning.

Contact: Frank Craig

Boeing Employee Information System

The tracking of information about personnel, equipment, software, and other related data is a time-consuming obligation that the Boeing Company deals with on a daily basis. Keeping up with this information and using it in an intelligent way is the goal of the Boeing employee information system (BEIS). It is used to help with both tracking requirements and analysis efforts, such as resource allocation, network load balancing, and personnel placement.

Contact: Ronnie Long

Adaptive Modeling System

The adaptive modeling system (AMS) models a user's accesses to a relational database (Tanner and Graves 1990). AMS changes the model to constantly adapt to the way in which the database management system is being accessed by a particular user. Intelligent use of the knowledge stored in the model enables the system to recognize patterns and trends. This information can be used by both the users and the administrator of the database management system. For example, the system offers recommendations in real time to assist in the creation of query commands.

Contact: Steve Tanner

Neural Network Applications of the Advanced Civil Space Systems Group

The Advanced Civil Space Systems Group at Boeing is focusing on applying neural network technology to space exploration, specifically to spacecraft missions in the following areas: (1) aerocapture closed-loop guidance, (2) rocket (ascent) closed-loop guidance, (3) satellite trajectory data compression and storage, and (4) atmosphere modeling.

Guidance and control systems for aerobraking space missions will require smart computer technology. Environmental conditions (such as those in the Martian atmosphere) are not always well quantified. Therefore, the guidance and control computer must be able to adapt to unknown and changing conditions while it performs aerobraking maneuvers. Current advanced guidance algorithms, also called *predictor-correctors,* can only provide accurate aerobraking guidance for relatively small atmospheric perturbations. This guidance is computationally intensive (requires a high-powered and expensive flight computer) and might not converge on a solution.

Neural network technology is ideally suited for implementation as a smart computer. It reduces the software storage and maintenance requirements. It is robust and adaptive, and it is fast because the system is inherently a parallel processor that can be implemented in a hardware chip. There is no need to search for a solution on the fly because the network can be trained before the flight. The network already contains many solutions for the different types of perturbations the spacecraft might encounter in the atmosphere. Additionally, the network's ability to extrapolate based on its experience adds up to a system that is robust.

Preliminary results are excellent. The neural network guidance algorithm is accurate and has proven more robust than a standard advanced guidance algorithm for cases studied to date. It generates the solutions in two orders of magnitude less time.

Satellite trajectory information storage is another application being explored. Some satellite missions (such as the *TOPEX-Poseidon*) require that the satellite have real-time, onboard knowledge of its own trajectory and that of others for use in attitude, determination, control, and high-gain antenna pointing. Software storage on board a satellite is expensive; therefore, the objective is to minimize software storage requirements yet maintain a high level of accuracy. Engineers at Boeing and Thomas Kelly, an assistant professor at the University of Dayton, designed a technique using advanced mathematics and neural networks to compress and store satellite trajectory data. Preliminary results demonstrate that the technique satisfies the objective better than other methods.

Atmosphere modeling by neural networks is another application being researched. The purpose is to design a method that in a short time can model a particular region of atmosphere through which an aircraft or aerobraking spacecraft will be flying. The model can enhance guidance and control performance by providing the flight computer a priori knowledge of the environment through which it will be maneuvering. A neural network is trained to learn the nonlinear altitude and density profiles at specific locations (training points) in a region of atmosphere. The data can be obtained from light detection and ranging (LIDAR) devices placed at the training locations. After successful learning, the network is prompted for the density profiles at test points in the same region of atmosphere. Initial simulation results indicate that this system is promising (Fouche et al. 1991).

Contacts: Michael Fouche and Jill Nordwall

User Interfaces and Virtual Reality

We often have customers who need to interact with information or users in unique ways. Much of our work in this area has grown out of our experience with user interfaces that we've had to develop for our expert systems. These interfaces tend to work best when they're graphic and mouse driven, with some sort of a model of the user's behavior built in. In addition, we've begun to explore the use of virtual reality as a method for allowing users to interact and visualize information. The following subsections present a few of the projects in this arena.

Graphic Payload Accommodation Tool

As the Space Station Program matures, the complexity and quantity of multidisciplinary and engineering data grow at an exponential rate. This growth presents an overwhelming problem. However, electronic information systems have the capability to provide compact storage of

phenomenal quantities of complex data in various formats, including textual, pictorial, audio, and numeric. The graphic payload accommodation tool (GPAT) is a computer-aided training system (Daughtrey 1987) developed to provide users of Space Station *Freedom* with basic information needed for payload accommodations.

The system is intended for use by payload and experiment developers. However, this tool can also be used by new employees as an orientation to the program, by design engineers and payload integrators as an information repository, and for general program awareness. GPAT represents a practical, cost-effective approach to familiarization training for payload accommodations. It has been accepted as an improvement in the quality of communication of multidisciplinary engineering data.
Contact: Amber Olson

Virtual Reality

Virtual reality is in its infancy, and for the most part, the equipment and tools are slow and cumbersome and of unsatisfactory resolution. However, the potential is clearly there for far more interesting things to come. When virtual reality is viewed as simply another input-output medium, it doesn't really fall under the general heading of AI. However, if one considers how it can be exploited, it can be viewed as a system-modeling tool with an intimate user interface.

The directions and project areas in which virtual reality is being used at ACG revolve around two primary areas. The first and most easily accomplished area deals with data visualization. The goal is to present information in new ways that take advantage of the real-time modeling capabilities of virtual reality. A primary objective is to allow users to see data in a more intuitive way and, thus, help overcome the information overload that often accompanies large sets of data or to allow users to see data in different ways that allow them to make serendipitous observations.

The second area of application for virtual reality deals with the interactive and immersive modeling of man-machine interfaces. The design of Space Station *Freedom* and DoD projects make extensive use of computer-aided design and computer-aided machine capabilities and prototype studies. If a designer could enter the models themselves, they could test their ideas without having to build expensive physical models. Of course, physical prototypes will need to be built in the end, but virtual reality offers the chance to eliminate some of the more glaring problems before they get too expensive to correct.

We are currently working on several virtual reality projects. One of them is a lunar rover radiation effects model. The Advanced Civil Space Systems Group is in the process of designing Lunar and Mars rovers for future manned missions. One of the problems that must be addressed is radiation shielding. Virtual reality allows the designers into their models where they can observe first hand the dosage levels of both primary and secondary radiation coming from the various surfaces within the model.

Another project is a human factors study in which the goal is to bring the engineering design documents into the virtual world and support the designers in the analytic verification of their human-engineering requirements. Initially, such things as color schemes or lighting placement can be studied. As the models become more sophisticated, human interaction will come into play. Models of different classes of humans will be inserted into the system to study the man-machine problem areas.

Our long-term goals are fairly ambitious given the state of virtual reality equipment today, but some of the simpler tasks are approachable now. The current problems with the hardware technology (speed, resolution, and tracking abilities) are steadily improving, but our links to other more sophisticated system-modeling tools are not yet available.

Contact: Steve Tanner and Cindy Kamhieh

Model-Based Reasoning

Some of the most interesting work is also a tremendous technical challenge—providing the environmental control and life support system (ECLSS) for Space Station *Freedom*. The Advanced Automation Project (AAP) is one aspect of ECLSS and one of the larger long-range research and development efforts that we've been involved with. The first project listed here is a general overview of the model-based reasoning efforts that are part of AAP. The other projects described within this section highlight work on diagnosability. (*Diagnosability* is a characteristic of system design that determines the extent to which faults in a system can be detected, isolated, and predicted.)

The Advanced Automation Project

The aim of AAP is to provide advanced diagnostic and monitoring methods for ECLSS of Space Station *Freedom* (Johnson 1990). The project has two major objectives: (1) to use advanced diagnostic and monitoring methods to improve ECLSS automation during both design and operation and (2) to provide feedback to the space station's advanced control community on the tools, procedures, and difficulties involved in using standard knowledge acquisition, knowledge engineering, development, integration, and verification.

The objectives for the project are to demonstrate monitoring, fault detection, and diagnostic capabilities for the carbon dioxide removal assembly (CDRA) of ECLSS. The technology focus has been on model-based reasoning using the knowledge-based autonomous test engineer (KATE) tool (Scarl, Jamieson, and DeLaune 1985; Scarl and Marcotte 1991).

ECLSS will sustain a safe "shirt sleeve environment" for the crew and payloads of Space Station *Freedom*. Its functions have been divided into six interconnected subsystems, shown in figure 2. The subsystems must supply water and air for extended durations without the crew's intervention (Dewberry 1989, 1990a, 1990b). Methods for performing diagnostic tasks on these subsystems can be divided into three categories: reactive, heuristic, and comprehensive.

Reactive diagnoses are sensed

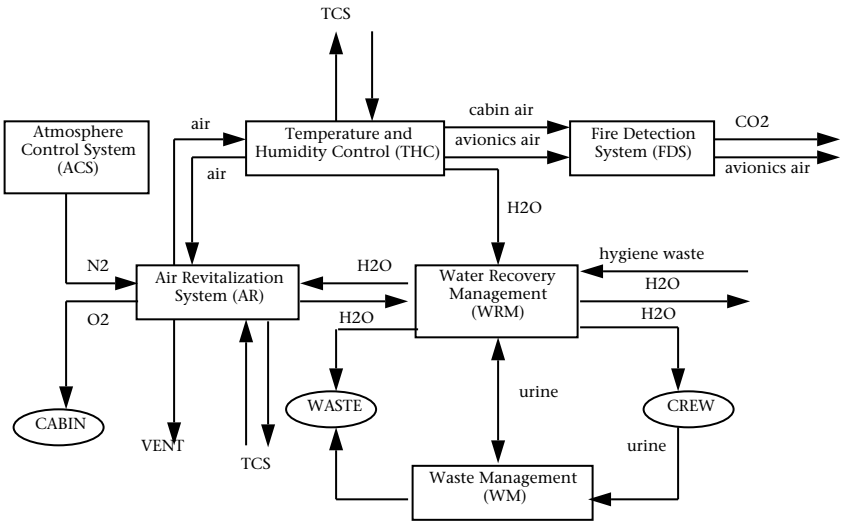

Figure 2. Functionally Interconnected Subsystems of the Environmental Control and Life Support System.

directly, and their specification is already built into the baseline ECLSS design and is not part of AAP. Heuristic diagnoses are made using rules of thumb. These rules of thumb are typically implemented using associative representations (if-then rules, fault trees, causal networks, directed graphs, and so on) and describe the mechanism's behavior in terms of its possible faults. These representations are shallow and are sensitive to mechanism configuration-mode and component state changes.

Comprehensive diagnosis uses mechanism descriptions that are grounded in fundamental domain principles. These principles govern each component's behavior whether it is broken or fault free. Most automated programs for comprehensive diagnosis use *model-based diagnosis*, which represents the mechanism using component-connection models. It then uses the causal propagation of parameter values through component networks to discriminate between diagnostic hypotheses. Recent work in the project has addressed the issue of integrating the model-based work with the associative systems already in place for the Space Station Program.

The KATE diagnostic algorithm scans a set of sensor readings, comparing them to a set of predicted values. Discrepancies between the mechanism's expected behavior and the sensor values are taken as indications of some fault. When a measurement discrepancy is noted, the diagnoser is invoked to localize the fault to some particular component. *Diagnosis* is the search for one or more faults that can explain the mechanism's observed behavior. The strength of this modeling lies in its ability to hypothesize faults from the information given by discrepant sensor readings (Scarl, Jamieson, and DeLaune 1985) and then to subsequently evaluate the hypotheses, searching for one that makes the model's behavior match the observed mechanism behavior. The diagnostic algorithm is discussed in more detail in Scarl, Jamieson, and DeLaune (1987). An earlier, more structurally oriented diagnostic algorithm is discussed in Scarl, Jamieson, and DeLaune (1985).

Contact: David Throop

Formalizing Diagnosability in Dynamic Systems

This work is an investigation into a formalism for a theoretical definition of diagnosability from system models (Carnes and Fisher 1992; Misra et al. 1992). The diagnosability of some device is a measure of whether a particular fault in the device can be detected uniquely, isolated, or predicted by the observable sensor evidence in the current system configuration. As such, we are investigating three characteristics of diagnosability: detectability, distinguishability, and predictability.

According to our investigations, the application of discrete-event dynamic system (DEDS) theory and formalism in diagnosis and diagnosability research offers several advantages: (1) a robust modeling paradigm that is less sensitive to modeling uncertainties; (2) analytic and design tools that are synergistic with the inherently discrete nature of a broad class of diagnostic and fault-recovery problems; and (3) a rigorous theoretical foundation for the diagnosis and diagnosability of large-scale, or hybrid, dynamic systems.

Our definition of diagnosability characterizes the structure of the system and the method of its observation and does not focus on a specific diagnostic technique. Consequently, our definition of diagnosability strongly depends on a modeling formalism that represents the system behavior. In the case of linear dynamic systems, the concept of diagnosability is well defined and is based on analytic observability. Unfortunately, the dynamic system models cannot be used to represent a large class of large-scale, heterogeneous systems. Without a precise modeling formalism, the definition of diagnosability is weaker and usually is not separated from the diagnostic algorithm used. Our purpose in this project is to develop a modeling formalism for this system category and provide a strong definition of diagnosability.

Our work focuses on logic-based models that capture the description of DEDS trajectories in terms of a small set of logical operations on functions of state and events. We use language coverings within the logical model to characterize specific diagnostic behavior, then diagnostic control is formally developed through techniques from language observability.

We believe that DEDS theory offers

a fertile area for research in diagnosis. Current research addresses problems in three areas: First is the formal description of diagnosis-related problems. Techniques from DEDS theory are meaningful in diagnosability and can offer a precise interpretation of concepts that are defined intuitively in other results. Second is DEDS modeling of large-scale, hybrid systems. DEDS models have been used primarily to describe inherently discrete process and real-time software systems. Using these models as an approximation of dynamic systems is a new area of research. Third is algorithm development. The success of the practical application of the DEDS models in diagnosis largely depends on algorithms that meet the requirements of diagnosis and fault recovery. Although some results are readily available, this research area is wide open for further contributions.

Contact: Ray Carnes

Inductive-Learning Approaches to Sensor Placement and Diagnosis

Proper sensor placement is vital for diagnosis and monitoring. Each sensor has resource costs relating to power, mass, and design complexity. We are using machine-learning methods to uncover behavioral patterns from snapshots of system simulations that will aid sensor design management and diagnosis, with an eye toward minimality, fault coverage, and noise tolerance.

We are using inductive machine-learning methods to identify categories of system behavior that have similar measurable quantities. We use a supervised learning system known as C4.5 (Quinlan 1987) to form a diagnostic rule base in the form of a decision tree. The decision tree is used to discriminate the system perturbations generated during a simulation-learning procedure. Our research has produced three aspects of this inductive analysis that are of particular interest:

First, the fault-granularity ratio indicates the degree to which a behavior's fault can be isolated using a rule base. Inversely, the ratio is a measure of the extent that we will have to rely on other sources of knowledge and diagnostic procedures to discriminate the fault.

Second, the parameter-compression ratio indicates the proportion of system parameters that are needed for diagnosis over a population of behaviors. This ratio is a guide to the number of sensors required if diagnosis is to rely simply on sensor values.

Third, the diagnostic accuracy is the percentage of behaviors correctly categorized as one of several possibilities. Diagnostic accuracy measures the reliability of diagnosis with the rule base. Each of these aspects can be used to describe the success of a diagnostic task and provide guidelines for designing with diagnosability. Our particular concern in this latter regard is with sensor placement.

Clustering systems automatically discover categories of observations (events or objects) that are similar along some dimension(s). Once uncovered, these categories can suggest features that characterize the observed data or facilitate predictions about the nature of future data.

A clustering system constructs a classification scheme over a set of observations. We use COBWEB (Fisher 1987) to discover categories of fault conditions over system simulations. In diagnosis, an observation might be a set of symptoms that collectively indicate a class of events that share a common diagnosis. We believe that discovered clusters can be used dynamically for automated diagnosis and that an engineer can use clusters over simulated behavior to facilitate design, in this case, sensor placement.

The benefits of clustering are at least twofold: First, it is difficult for engineers to completely design in advance against system faults. Collectively, simulation and clustering identify fault models that benefit design decision making. Second, a COBWEB classification tree can be used to facilitate fault diagnosis. In particular, categories discovered through clustering observable features with component faults lead to the identification of behavioral anomalies.

Contact: Ray Carnes

Robust Diagnosis and Diagnosability Analysis

In the highly automated environment of Space Station *Freedom*, sensors provide the data needed for control, monitoring, and diagnosis of systems. The reliability and cost of these sensors are important issues. We have developed an application to model a system using a multiple-aspect, hierarchical modeling paradigm (Karsai et al. 1992), which captures the system dynamics in terms of a fault-propagation model (Padalkar et al. 1991).

The first part of this work deals with providing reliable diagnosis of failures in a system given the possibility that sensors in the system might fail. Because the diagnoser has to locate the fault(s) by interpreting the observations, the correctness of diagnostic results depends on the reliability of observations. The observations can be erroneous because of (1) wrong data reading by sensors, (2) modeling error, and (3) loss of model validity. We developed a robust diagnostic system to handle these possible errors. The diagnostic system uses the physical and temporal constraints imposed on observations in a dynamic system to identify the presence of erroneous observations. The robust diagnostic system was integrated with an operations automation test bed, and its ability to handle sensor failures was demonstrated.

The second part of the work deals with minimizing the costs associated with sensors without sacrificing the diagnosability of the system. We have developed a diagnosability analysis tool (DAT) that facilitates the analysis of the diagnosability of a system in terms of the sensors in the system. We defined three metrics: (1) *detectability*, which gives the longest time that is needed to detect a failure; (2) *predictability*; which gives the shortest time between the warning and the actual occurrence of a failure; and (3) *distinguishability*, which describes the size of the ambiguity sets given a time limit for the observation.

Using these metrics, DAT provides two kinds of analyses. In evaluation mode, the characteristics of a design

with a predefined sensor allocation are calculated. In advice mode, an arbitrary set of requirements can be defined for the characteristics, and the tool generates a satisfactory sensor placement.

Contact: Ray Carnes

Other Projects

Here, we discuss other areas that ACG is involved with. These areas cover a wide range of projects that didn't fit well in any of the previous topics.

Design Knowledge Capture

Design knowledge capture is a broad area that ACG has explored several times in the recent past. Experience with major engineering tasks has proven that design specifications alone are often inadequate to guide future design efforts. Consequently, we feel design rationale should also be captured for use in both future design changes and new development efforts. These design rationales include the lessons learned, the reasons underlying engineering decisions, and the reasons alternative specifications are rejected. Two projects in this area are AUTOSHRED and DART.

AUTOSHRED automates the process of requirement shredding of massive volumes of requirement documentation. DART is based on MACQUINAS, a Boeing proprietary knowledge-acquisition tool developed by Boeing Computer Services in Seattle (Boose and Bradshaw 1987). The purpose of the DART tool is to elicit knowledge from experts in various engineering domains for the purpose of (among other things) performing clustering and implication analysis.

Contact: Amber Olson

Improving Logic-Programming Languages

We developed a technique for indexing PROLOG clauses such that all clauses, no matter what their structure, are distinguished. Each fact or head of a clause that is retrieved is guaranteed to unify with the subgoal that triggered its retrieval. The key idea is to store subgoal schemata in addition to the heads of clauses. All clause heads that match each schema are associated with this schema. At execution time, the number of schemata to which each subgoal must be matched to retrieve all clause heads that will unify with the subgoal is no greater than $2^{(k/2)} - 1$, where k is the number of parentheses in the subgoal. An unlimited number of clause heads, guaranteed to unify with the given subgoal, can be retrieved together at essentially the same time. For example, hundreds or thousands of clause heads known to match a particular schema can be stored together in secondary store and retrieved as a group.

Retrieving and then matching *clause heads* (rule heads and facts) to subgoals is a central task of the interpreters of logic-programming languages such as PROLOG. The overall efficiency of such interpreters is largely dependent on this task. Various indexing techniques are currently used to increase the efficiency of the clause-head matching process. These techniques include using specialized hardware (Colomb 1991); indexing on the first functor of selected arguments of each clause head, as in QUINTUS PROLOG (Quintus 1990); encoding clause heads as binary code words (Colomb 1991); and indexing using a string for each clause head to represent the constants and part of the structure of this clause head.

All these indexing techniques, except representing clauses as binary code words, permit some clause heads to be retrieved that subsequently fail to unify with the given subgoal. The problem is that clause heads of various types, such as those with repeated variables or those with PROLOG structures as terms, are not distinguished. In contrast to other current techniques, Colomb's method requires specialized hardware support and is limited to indexing procedures with at most a few tens of thousands of clauses.

Current indexing techniques retrieve each clause head for unification with the given subgoal as a separate step. Our indexing technique enables the retrieval of a large number of matching clause heads at once.

An example of a subgoal schema is $c1(\$X1, c2, \$T)$. A given subgoal matches this schema if and only if it has a variable as its first argument, a constant as its second argument, and any structure as its third argument, where the constant argument is not the same as the functor. We call this type of schema a *short schema*. In contrast, a *long schema* has a short schema as its first section. The second section of a long schema specifies which syntactic components of the given subgoal must unify with one another. The third section specifies which constants in the given subgoal permit a match between the subgoal and the schema.

Various methods of encoding the subgoal schemata of a PROLOG program permit the selection of corresponding trade-offs between the space required to store the PROLOG programs and the time required to retrieve clauses. Figure 3 reflects one such method.

In the worst hypothetical case, the fastest algorithms for matching and retrieving clauses based on our technique require a large amount of memory to store thousands of schemata. However, by combining an appropriate algorithm based on our technique with conventional clause-retrieval techniques, program execution can be accelerated, and available memory is used. Prior to program execution, the schemata of all possible subgoals that unify with each clause head of the program are constructed. Figure 3 shows the set of 10 schemata that are constructed for the clause head $r(\$Y1,a)$. These 10 schemata are also shown in a tree. Each path from the root to a leaf represents one long schemata. No middle section of the long schemata is required in this case because the syntactic variable $\$T$ occurs no more than once in any schema.

If the clause head $r(\$Y1,a)$ were stored directly, three syntactic components would be stored, five if the two parentheses are included. The tree in figure 3 requires the storage of the short schemata, which can be shared with other clause heads, and the 19 constants below the dotted line. In this example, about 15 addi-

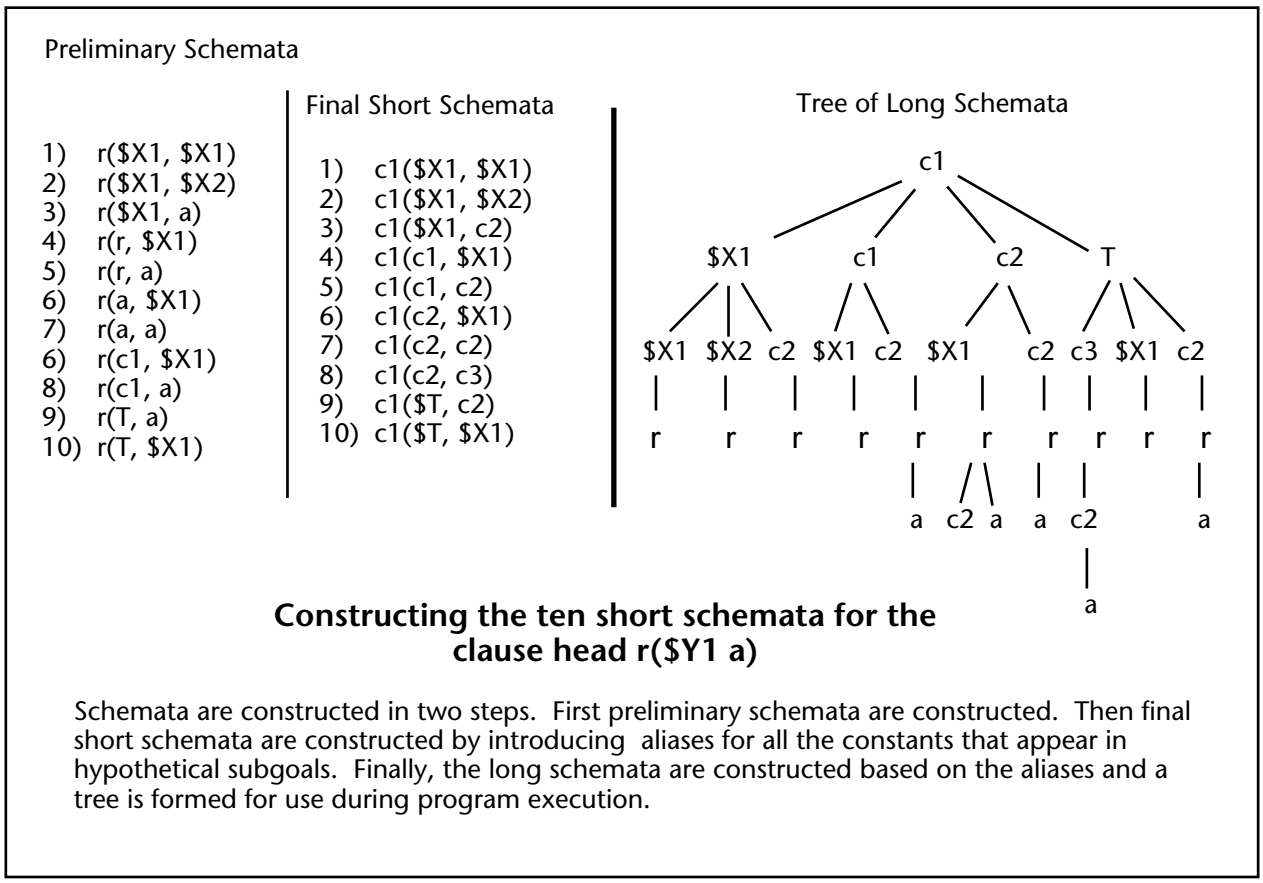

Figure 3. Constructing the Schema.

tional constants and the pointers to them must be stored. Any subgoal that unifies with *r*($*Y*1,*a*) will match one and only one path from the root to a node just above the dotted line. At most, 2*C* leaves below the node must be visited to find all clause heads, including *r*($*Y*1,*a*), that will unify with the given subgoal, where *C* is the number of constants in the subgoal. Because the subgoal is matched to a path below the dotted line, each constant in the subgoal can match only a node labeled with the constant or a node labeled by a lexical variable of the form *Ci*, where *i* is an integer. Consequently, in the worst case, a binary tree with 2*C* leaves must be traversed.

We are currently investigating the storage requirements of existing PROLOG programs. We expect real programs to fall well short of the theoretical upper bound and permit significant acceleration of program execution.

Contact: Jeff Jackson

Integrating Lisp and PROLOG to Support Natural Language Generation

A logic-programming language was implemented in Common Lisp to facilitate the automatic generation of English sentences. To implement the code that generates English statements, we wanted a PROLOG-like programming language. We also wanted logic programs to interface easily with Lisp on a single platform so that the programmer can use whichever programming paradigm best fits the particular problem at hand.

An advantage of integrating logic and Lisp programming is that the Lisp functions can return instances of logical queries or test the truth of statements. This integration permits the use of the logic-programming apparatus for fact retrieval and for the handling of logical rules to answer the question asked at a branch statement.

Our implementation permits Lisp functions to play the role of built-in predicates. We also implemented a delayed evaluation capability and other mechanisms that permit logic programming to be "more declarative" than traditional PROLOG programs. This capability permits the programmer to place the subgoals in the body of a rule without regard to their order.

Contact: Jeff Jackson

Conclusions

ACG has grown over the past several years by maintaining that AI can be useful and practical in today's corporation. We have made it so by combining basic research with applied projects. This mix allows us to both keep our customers satisfied and our researchers happy. It's not easy and takes a delicate mix of flexibility, credibility, salesmanship, and lots

and lots of hard work. Please feel free to contact ACG if you want further information on any of the projects listed here or about other work we are currently pursuing.

References

Allen, J. 1983. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM* 26(11): 832–843.

Boose, J., and Bradshaw J. 1987. Expertise Transfer and Complex Problems: Using AQUINAS as a Knowledge-Acquisition Workbench for Knowledge-Based Systems. *International Journal for Man-Machine Studies* 26(1): 3–28.

Carnes, R., and Fisher, D. 1992. Inductive Learning Approaches to Fault Isolation and Sensor Placement. Presented at the Third International Workshop on Principles of Diagnosis, Rosario, Washington, 12–14 October.

Carnes, R., and Underbrink, A. 1991. Task Planning for Autonomous Control Systems. In Proceedings of the IEEE International Symposium on Intelligent Control, 347–352. Washington, D.C.: IEEE Computer Society.

Colomb, R. 1991. Enhancing Unification in PROLOG through Clause Indexing. *Journal of Logic Programming* 10(1): 23–44.

Craig, F.; Cutts, D.; Fennel, R.; Case, C.; and Palmer, J. 1990. A Knowledge-Based Approach to Configuration Layout, Justification, and Documentation. In Proceedings of the Fifth Conference on Artificial Intelligence for Space Applications, 95–101. Huntsville, Ala.: National Aeronautics and Space Administration.

Daughtrey, R. 1987. Intelligent Man-Machine Interfaces on the Space Station. In Proceedings of the Second Conference on Artificial Intelligence for Space Applications, 321–325. Huntsville, Ala.: National Aeronautics and Space Administration.

Daughtrey, R.; Fennel T.; and Schwaab, D. 1991. PACKMAN: A Simulated Annealing Approach to 3-D Packing with Multiple Constraints, Boeing Internal Report HSV-ACG-ALEPS-001, Boeing Computer Services, Hunstville, Alabama.

Davis, W. 1990. Agent-Independent Task Planning. In Proceedings of the Fifth Conference on Artificial Intelligence for Space Applications, 1–10. Huntsville, Ala.: National Aeronautics and Space Administration.

Davis, W., and Carnes, C. 1991. Clustering Temporal Intervals to Generate Reference Hierarchies. In Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning, 111–117. San Mateo, Calif.: Morgan Kaufmann.

Dewberry, B. 1990a. Automation of the Environmental Control and Life Support System, Technical Report, National Aeronautics and Space Administration, Marshall Space Flight Center, Huntsville, Alabama.

Dewberry, B. 1990b. Space Station Freedom ECLSS—A Step toward Autonomous Regenerative Life Support Systems. In Proceedings of the Fifth Conference on Artificial Intelligence for Space Applications, 193–201. Huntsville, Ala.: National Aeronautics and Space Administration.

Dewberry, B. 1989. The Environmental Control and Life Support System Advanced Automation Project—Phase I Application Analysis. In Proceedings of the Space Operations, Automation, and Robotics Conference (SOAR-89). Houston, Tex.: National Aeronautics and Space Administration.

Dewberry, B., and Carnes, R. 1990. The ECLSS Advanced Automation Project Evolution and Technology Assessment. In Proceedings of the Space Station Evolution Workshop, 543–566. Huntsville, Ala.: National Aeronautics and Space Administration.

Fisher, D. 1987. Knowledge Acquisition via Incremental Conceptual Clustering. *Machine Learning* 2:139–172.

Fouche, M.; Nordwall, J.; Rao, N.; Tillotson, B.; and Woodcock, G. 1991. Neural Networks–Based Guidance for Aerobraking Applications (Part 1: Atmosphere Prediction). In Proceedings of the Second Workshop on Neural Networks, 529–541. San Diego, Calif.: Society for Computer Simulation International.

Johnson Research Center, the University of Alabama at Huntsville. 1990. ECLSS Advanced Automation Preliminary Requirements: Final Report, National Aeronautics and Space Administration, Marshall Space Flight Center.

Karsai, G.; Sztipanovits, J.; Padalkar, S.; Biegl, C.; Miyasaka, N.; and Okuda, K. 1992. Model-Based Intelligent Process Control for Cogenerator Plants. *Journal of Parallel and Distributed Computing* 15(2): 90–102.

Misra, A.; Sztipanovits, J.; Carnes, J.; Underbrink, A.; and Purves, R. 1992. Diagnosability Analysis and Robust Diagnostics with Multiple-Aspect Models. Paper presented at the NASA Workshop on Monitoring and Diagnosis, Pasadena, California, 15–17 January.

Padalkar S.; Karsai, G.; Biegl, C.; Sztipanovits, J.; Okuda, K.; and Nivasaka, N. 1991. Real-Time Fault Diagnostics. *IEEE Expert* 6(3): 75–85.

Quinlan, J. 1987. Simplifying Decision Trees. *International Journal of Man-Machine Studies* 27(3): 221–234.

Quintus Computer Systems. 1990. QUINTUS PROLOG Users Guide. Quintus Computer Systems, Mountain View, California.

Scarl, E., and Marcotte, R. 1991. The LOX Expert System: Final Report, Technical Report MTR11041, MITRE Corporation, MacLean, Virginia.

Scarl, E.; Jamieson, J.; and DeLaune, C. 1987. Diagnosis and Sensor Validation through Knowledge of Structure and Function. *IEEE Transactions on Systems, Man, and Cybernetics* SMC-17(3): 360–368.

Scarl, E.; Jamieson, J.; and DeLaune, C. 1985. Monitoring and Fault Location at Kennedy Space Center. *Newsletter of the ACM Special Interest Group on Artificial Intelligence* 93:38–44.

Tanner, S., and Fennel, R. 1991. The Placement of Equipment in the Space Station Freedom Using Constraint-Based Reasoning. In *Expert Systems World Congress Proceedings*, 1385–1398. New York: Pergamon.

Tanner S., and Graves, S. 1990. Artificial Intelligence Techniques for Modeling Database User Behavior. In Proceedings of the Fifth Conference on Artificial Intelligence for Space Applications, 525–534. Huntsville, Ala.: National Aeronautics and Space Administration.

Steve Tanner currently directs the Advanced Computing Group's (ACG) Virtual Reality and Visualization Laboratory. He was the founding member of ACG in 1985. He subsequently left to run the AI laboratory at General Research Corporation but returned to Boeing in 1990. His research interests focus primarily on human factors, intelligent interfaces, and behavioral modeling systems. He is pursuing a Ph.D. at the University of Alabama at Huntsville.

Ray Carnes is a principle scientist in the Advanced Computing Group at Boeing Defense and Space. His current research interests include diagnosis and sensor placement using machine-learning techniques. He is a Ph.D. candidate in electrical engineering at Vanderbilt University.