



The Space Congress® Proceedings

1988 (25th) Heritage - Dedication - Vision

Apr 1st, 8:00 AM

Ada and Knowledge-Based Systems: A Prototype Combining the Best of Both Worlds

David C. Brauer

Patrick P. Roach

Michael S. Frank

Richard P. Knackstedt

Follow this and additional works at: <https://commons.erau.edu/space-congress-proceedings>

Scholarly Commons Citation

Brauer, David C.; Roach, Patrick P.; Frank, Michael S.; and Knackstedt, Richard P., "Ada and Knowledge-Based Systems: A Prototype Combining the Best of Both Worlds" (1988). *The Space Congress® Proceedings*. 5.

<https://commons.erau.edu/space-congress-proceedings/proceedings-1988-25th/session-6/5>

This Event is brought to you for free and open access by the Conferences at Scholarly Commons. It has been accepted for inclusion in The Space Congress® Proceedings by an authorized administrator of Scholarly Commons. For more information, please contact commons@erau.edu.

EMBRY-RIDDLE
Aeronautical University™
SCHOLARLY COMMONS

MDC H1570A
AUGUST 1986

**ADA AND KNOWLEDGE-BASED SYSTEMS:
A PROTOTYPE COMBINING THE BEST OF BOTH WORLDS**

**DAVID C. BRAUER, PATRICK P. ROACH
MICHAEL S. FRANK, AND RICHARD P. KNACKSTEDT**


Presented by : Patrick Parker Roach
Artificial Intelligence Consultant
Network Solution Technology Center

digital

Digital Equipment
Centre Technique Europe
B.P. 29 - 06561 Valbonne Cedex
France - Tél. 92 95 51 11
Télex 461 837 F

MCDONNELL DOUGLAS ASTRONAUTICS COMPANY-HUNTINGTON BEACH

5301 Bolsa Avenue, Huntington Beach, California 92647 (714) 896-3311

MCDONNELL DOUGLAS 
CORPORATION

**ADA* AND KNOWLEDGE-BASED SYSTEMS:
A PROTOTYPE COMBINING THE BEST OF BOTH WORLDS**

David C. Brauer, Patrick P. Roach, Michael S. Frank,
and Richard P. Knackstedt**

**McDonnell Douglas Astronautics Company
5301 Bolsa Avenue
Huntington Beach, California 92647**

ABSTRACT

We describe a software architecture based on Ada tasking and packaging which facilitates the construction of distributed knowledge-based systems. We used this architecture to build the Knowledge-Based Maintenance Expert System (KNOMES) prototype for the Remote Manipulator System (RMS) of the NASA Space Station Mobile Service Center. Each module of the system contains Ada packages of standard systems services, which interface with an artificial intelligence/knowledge-based system (AI/KBS) language component that performs knowledge-based reasoning. By using Ada as the fundamental structure, we achieved a well-structured, maintainable program; by retaining the AI/KBS language component, we were able to capture the knowledge needed to solve ill-structured, dynamic, and/or nonalgorithmic problems.

1. INTRODUCTION

In general, the more complex a system is, the more difficult it is to maintain. As systems developed for NASA and the Department of Defense have become more complex, this problem has begun to increase life-cycle costs, especially for monitoring, fault detection and isolation, fault diagnosis, and repair. McDonnell Douglas began in 1985 an independent research and development (IRAD) project, titled Knowledge-Based Maintenance Systems (KBMS), whose primary objective was to apply knowledge-based (expert) systems techniques to these maintenance tasks.

The KBMS project has now expanded into three major efforts. The first is to establish requirements and designs for a generic Knowledge-Based Maintenance Systems Support Environment (KBSSE) tool, which will be used to construct and maintain knowledge-based maintenance systems in a variety of domains.

The other two efforts are under the heading of advanced development for the Space Station Definition and Preliminary Design, WP-02, contract. One is a requirements study to identify and classify the knowledge that should be collected to support the construction of knowledge-based maintenance systems when a system such as the Space Station is designed and developed. The study will recommend additional requirements on Space Station design data bases to facilitate this knowledge

collection. The remaining part of the KBMS project, and the one which will be covered in detail in this paper, is the development of a prototype knowledge-based maintenance system.

The purpose of the prototype, which we have named KNOMES for Knowledge-Based Maintenance Expert System, is to apply knowledge-based monitoring and fault management systems to Space Station subsystems. For our initial work, we selected the Remote Manipulator System (RMS) component of the Space Station Mobile Service Center (MSC), primarily because representative test case data in the form of shuttle telemetry tapes was available. However, the RMS KNOMES prototype is representative of the type of knowledge-based systems which will be needed for the Space Station. In addition, we expect our KNOMES prototype to support the choice of Ada as the primary language for organizing and implementing on-orbit expert systems. We believe the documentation and guidelines for our prototype will establish the basic technology for Space Station KNOMES applications.

2. CONCEPT

We feel that a KNOMES application will be part of the Space Station operational flight software. If it is, there are two directives of the Space Station Program that are particularly relevant. The first is the selection by NASA of Ada as the programming language for all flight software. The second is the congressional mandate that at least ten percent of the total budget for the Space Station will be spent on advanced automation and robotics, including artificial intelligence. Efforts on the KNOMES prototype have been guided by both of these directives.

2.1 Requirements

As a potential component of the Space Station computing environment, the KNOMES prototype must address certain requirements. The first follows from the Ada directive: the KNOMES prototype should be implemented to the greatest extent possible in Ada. It follows that the KNOMES prototype should be implemented on a standard processor supporting a verified Ada compiler. In addition, a KNOMES application will be performing monitoring and diagnosis functions in a time-critical, potentially life-threatening environment. For this reason, the system should function in real time or near real time and be fault tolerant. To meet these needs, the system should have built-in concurrency for eventual implementation on distributed and/or parallel computer architectures.

Consideration must also be given to integrating KNOMES applications into identified Space Station configuration items. As a software item aboard the Space Station, a KNOMES

*Ada is a trademark of the US Government.
(Ada Joint Program Office).

**Digital Equipment Corporation, Artificial Intelligence Consultant

application will draw on the computational resources of the Data Management System (DMS). The DMS as currently defined¹ will include a variety of hardware and software services such as:

- Local area network.
- Network interface units.
- Standard data processors.
- Interface devices.
- Mass storage.
- Display/control devices.
- Network operating system.
- Mass memory management.

Interfaces to each of these services can be defined as Ada packages. Then if the character of the services changes, only the affected package need be changed; the KNOMES application will remain unaltered. We will define these interface packages as the DMS services required by KNOMES are identified.

The other pertinent configuration item is the Operations Management System (OMS). The OMS is "intended to provide sufficient automation to remove the need for manned-intervention for conduct of operations except where it is required and/or necessary."² Requirements have been established for the OMS in three major areas:

- Operations scheduling and execution.
- Monitoring and reactive control.
- Sustaining operations support.

A KNOMES application within the OMS software configuration would fulfill requirements in the areas of monitoring and reactive control and sustaining operations support. In particular, KNOMES would play a central role in monitoring a system or subsystem and assisting in the correction of any detected or predicted failures. We describe the processes a KNOMES application would use to carry out these tasks next.

2.2 Processes

A complete KNOMES application carries out five processes within the OMS:

- Monitor system behavior.
- Isolate detected failures and/or anomalies.
- Diagnose systems to identify faults.
- Predict potential failures or functional loss.
- Perform health checks on newly replaced components.

KNOMES will monitor subsystem data from sensors, Built-In Test Equipment (BITE), and occasional human input. If an anomaly is detected in the incoming data, KNOMES will perform a rapid isolation to determine if any critical functions or missions are impaired and, if so, will notify an external caution and warning system. At the same time, KNOMES will diagnose the anomaly to determine the actual fault or failure. Diagnosis may at times require data from the isolation process, historical data, and detailed monitoring under checkout procedures. KNOMES will carry out prediction of faults or failures based upon trends in the subsystem data. It will also be called upon at times to perform health check procedures on newly replaced components.

In the early KNOMES prototypes, we focused on the monitor and diagnose processes. We will add the isolate, predict, and health check processes as future enhancements. We describe the Ada-based architecture and implementation of the current prototype version below.

3. KNOMES PROTOTYPE

Before we describe the architecture of the KNOMES prototype, it is worthwhile to mention other research efforts in artificial intelligence and Ada. The majority of these efforts have focused on implementing artificial intelligence algorithms in Ada. This research indicates that a significant portion of AI/KBS algorithms can be recoded in Ada, but that this recoding takes much time and effort.³ There are also skeptics who believe that true AI will not be possible in Ada without important changes to the Ada specification, such as the passing of procedures as parameters.⁴

Rather than debate whether or not true AI can be done in Ada, we intend to build upon those features of the language that are consistent with the AI/KBS philosophy. Object-oriented programming, data abstraction, process abstraction, and a programming language support environment are common to both Ada and AI.⁵ Our approach is to build upon this common ground, constructing the basic framework of our program in Ada. However, for portions of the KNOMES prototype it will be necessary to use AI/KBS algorithms that cannot be implemented in Ada or that cannot be implemented within the scope of this project. Our strategy for these cases is to create an Ada interface to AI/KBS algorithms coded in AI/KBS languages like Ops5, Lisp, or Prolog.

Our approach, then, represents a hybrid solution composed of state-of-the-art Ada and AI/KBS. Experienced developers of real-world AI/KBS systems have indicated that only twenty to thirty-five percent of their systems could be classified as AI/KBS algorithms.⁶ Thus, the use of Ada as a primary programming language is not ruled out. Interfacing Ada to AI/KBS languages, of course, gives Ada access to a variety of powerful knowledge-based systems. We will be able to experiment with these systems and identify the knowledge structures and inference engines most suitable to meeting reasoning requirements for KNOMES. Furthermore, as research on the implementation of AI/KBS algorithms in Ada comes to fruition, we can use the resulting packages in our solution. Promising work has already been done on a Lisp implementation in Ada⁶ and on an Ada-based inference engine.⁷

3.1. Analysis and Design Issues

3.1.1 Common Routines Upon analyzing the primary processes of KNOMES, it became apparent that they had common needs: all of them needed to communicate with at least one other process at some time; most needed access to data bases and/or the actual input data stream; and some needed the ability to interface with humans. We identified the common routines and mapped them into Ada packages which could then be combined into a Common_Routines package for inclusion in any KNOMES process.

3.1.2. Concurrent Routines Analysis also indicated that the primary KNOMES processes were not strictly sequential. The monitor process, for instance, must remain active and fully functional while diagnosis is carried out. The potential exists for several diagnosis processes to be active at the same time, and the isolation process will definitely overlap both monitoring and diagnosis. We looked at two possible solutions to this concurrency problem. The processes can be implemented (1) as separate tasks in the Ada sense, or (2) as separate, independent, communicating processes on one or more processors. Our

preference is to pursue the first option, coding our application within the Ada tasking mechanism. However, there are situations where a single processor is not sufficient. Currently we must use option two to take advantage of multiple processors. However, as multiprocessor architectures for Ada become available we will return to option one, concurrent tasks within Ada.

3.1.3 Knowledge-Intensive Activities The intent of KNOMES applications is to aid humans in performing the monitoring, diagnosis, isolation, prediction, and health check functions and eliminate rote redundant activities. To do this, it will be necessary to capture and apply the knowledge human experts normally use in performing these tasks. Where we can capture this knowledge in the form of an algorithm, we will code it in Ada. However, much of human knowledge cannot be expressed as a well-formed algorithm, and here we must rely on AI techniques to capture this knowledge.

The Ada interface with traditional AI/KBS languages is accomplished, as mentioned above, by constructing the interface as an Ada package. The knowledge-based component of a KNOMES application is then coded in the appropriate AI/KBS language. Procedures and packages communicating with the knowledge-based component do so strictly via this interface. We are currently using Vax* Ops5, a forward-chaining production rule system, for knowledge-based components. The interface between Vax Ops5 and Vax Ada is fully functional.

3.1.4 Flexibility and Modularity We intend to create KNOMES applications when Space Station systems and subsystems are designed and developed. This means that the targets of KNOMES applications will be dynamic and rapidly changing. KNOMES must be flexible enough to easily accommodate these changes and highly modular so that changes in one module do not affect others. Being both flexible and modular, a KNOMES application can grow with a subsystem design. In the same way, on-orbit growth can be accomplished.

We allow for flexibility and modularity by treating the major processes of KNOMES as discrete objects in Ada.⁸ Interfaces between Ada objects need only be defined once in the package specifications. The internal processing of an object is hidden from the rest of the system and can be altered as required. Because the package specification does not change, the overall behavior of the system is not affected. We have already been able to take advantage of this feature. When the first version of the KNOMES prototype was coded, the exact format of the telemetry data used to drive the system was not available. We coded the data input as a separate package. The package specification for this data input established how data would be sent to the rest of the KNOMES prototype. We have since been able to change the body of the input package to process telemetry data of the correct format. These changes have not altered the package specification or how the rest of the system uses the data input package.

3.2 Prototype Architecture

3.2.1 ACTORS. We have called the basic architecture that supports the KNOMES prototype ACTORS,** for Ada Cognitive Task Organization Scheme. The key element in this scheme is a standard framework for all of the functional components of the KNOMES architecture. This framework is called an Ada Cognitive Task Organization (Actor). An Actor is a discrete

software module that can operate either as an Ada task or as a detached process. All Actor modules have similar structures, but in a working application each Actor in the system plays a particular role. An Actor can be a process, a function, or a service, determined by the role-specific Ada procedures added to the basic structure and by the knowledge added to the knowledge-based component of the Actor.

Each Actor is composed of two discrete sections with an interface between the two (Figure 1). The first section is a collection of Ada packages and procedures which represent the basic capabilities of the Actor. The section includes the Common_Routines package (see Section 3.1.1 above) which provides communication and data access routines. This section may also contain role-specific procedures or packages which implement algorithms necessary for the role the Actor is playing.

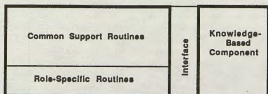


Figure 1. A Generic Actor.

The second section of the Actor is the knowledge-based component. Here the knowledge which enables an Actor to effectively perform its role is encoded. In the current prototype, this knowledge is written as rules. The Ops5 inference engine matches these rules against data describing the state of the Remote Manipulator System (RMS). This data is accessed via the interface and the Ada Common_Routines. The results of successful rule firings may also be communicated via this path.

The interface between the two sections is written as an Ada package. The advantage of defining the interface as a discrete object lies in the flexibility it provides. If it is discovered that a different AI/KBS language is appropriate for the knowledge-based component of an Actor, only the knowledge-base section and the interface need be changed; the Common_Routines and role-specific routines of the first section remain unchanged.

A KNOMES application using this architecture is simply a collection of Actors playing specific roles, which cooperate to accomplish the tasks assigned to the KNOMES application. Each Actor is a separate compilation unit which can function as an Ada task or as a detached process. These Actors can have different priority levels which can be dynamically adjusted so that when resources are constrained, the most important Actors receive resources first.

3.2.2 Community of Experts The cooperative framework of Actors which constitutes a KNOMES application can be viewed as a community of experts.⁹ Each modular Actor within this framework is in effect a semi-autonomous expert system with its own self-contained knowledge base and inference engine. The Actor is an expert at performing the role it was assigned. For example, a Manipulator Controller Interface Unit (MCIU) Diagnose Actor contains heuristic knowledge on diagnosing a failure in the MCIU of the RMS.

These modular experts are organized hierarchically, with some Actors having jurisdiction over lower level Actors, determining when they are used and examining their results. Actors within a level, such as diagnostics, may cooperate to achieve a desired goal (i.e., fault identification). Thus, a

*Vax is a trademark of Digital Equipment Corporation.

**McDonnell Douglas Astronautics proprietary development.

KNOMES application displays both of the general organizational schemes for distributed expert systems—hierarchical and cooperative.⁹ Communication within this framework is via pipelines between parents and children in the hierarchy. For cooperative problem solving, two or more Actors at the same level may use a common parent as an active "blackboard" for communicating hypotheses and results.

3.3 Prototype Implementation

The current KNOMES prototype is assigned the tasks of monitoring data from an RMS, detecting anomalies in the data, and diagnosing the cause of the anomaly. The Actors used to perform these tasks, their communication relationships and data access are shown in Figure 2.

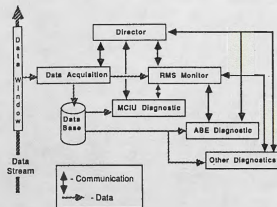


Figure 2. KNOMES Prototype Actors.

The Director is the first Actor invoked when the KNOMES prototype is started. It oversees the creation of the other Actors and keeps track of their status and relative priorities. The Director also has the capability to respond to resource conflicts and to modify the priority of other Actors. To make decisions on priority, the Director must have knowledge about which tasks are the most important in the current operational context. For example, if the RMS is being used for a critical operation, then monitoring the arm must receive highest priority. However, if the RMS is not being moved, a Diagnose Actor can be allocated a higher priority. The Director may also adjust the amount of data being sampled by Data Acquisition. Finally, when an active Diagnose Actor arrives at a diagnosis, the Director is responsible for comparing this result with other diagnoses, deciding if the diagnosis is valid and/or determining if the anomaly was a false alarm, and notifying the rest of the system.

The Data Acquisition Actor retrieves and formats data for the other Actors in the system and also puts time-stamped data into mass storage. This requires knowledge about what Actors require which data in the current operational context. The Data Acquisition Actor serves as the KNOMES window to the outside world. It uses the DMS services to acquire the data needed by the Monitor and Diagnose Actors. The intent is to have Data Acquisition keep up with the sensor and BITE data from the RMS which is transmitted over the DMS network in near real time.

The Monitor Actor receives data from Data Acquisition and watches for any data which may indicate an anomaly in the RMS. If it detects an anomaly, the Monitor Actor will notify

or activate the appropriate Diagnose Actor. This requires that the Monitor have knowledge of when data is indicative of an anomaly, and which Diagnose Actors handle which anomalies. It should be noted that the criteria for when data are anomalous change with the current operational mode and configuration of the RMS.

There are distinct Diagnose Actors for different diagnosable entities in the RMS. The current strategy is to assign a Diagnose Actor to each of the line replaceable units (LRU) in the system. The Director Actor will resolve conflicts that may arise from faults that propagate to more than one LRU. Each Diagnose Actor has the knowledge needed to determine if a fault or failure in its domain was responsible for observed anomalies. The current prototype has diagnostics for the MCIU and for Arm Based Electronics (ABE). Additional Diagnose Actors are being created. There are advantages to partitioning diagnosis knowledge among distinct Actors in this fashion. The knowledge base for each Diagnose Actor is kept relatively small, making it easier to verify the knowledge base. Also, several Diagnose Actors can be operating on parts of an observed anomaly in parallel, which should decrease the diagnosis time.

One final Actor of the current prototype is not shown in Figure 2. The Display Actor handles the interface between the KNOMES prototype and the human user. It contains very little knowledge at this point. However, the potential exists for adding knowledge about users and their respective display needs. Currently, the Display Actor can present a variety of displays to the user to aid understanding the system behavior. These displays include a simulated mission control console to display telemetry data, and windows which appear as an Actor is activated. The Actor windows display trace data showing how the particular Actor is functioning.

4. FUTURE DIRECTIONS

As the KNOMES prototype is improved, additional Actors will be added. First, more Diagnose Actors covering more of the LRUs in the system will be added. Isolate Actors will be added which are responsible for determining potential degradation of the RMS based on observed anomalies. This process is different from diagnosis because it proceeds on the assumption that the anomaly definitely indicates a failure. Using knowledge of how the components of the RMS function, the Isolate Actors will quickly determine if reconfiguration is needed to insure the safety of the crew or system.

We will also be adding Actors that can analyze trends in data and intermittent faults and predict potential failures in the system. These Predict Actors will use data on maintenance histories and mean time between failure (MTBF). They will use knowledge to correlate this data and predict failures. Users and/or other systems (schedulers) will be notified so that preventative maintenance can be performed.

The Explanation Actor is currently a high priority item to complete. It will be able to analyze and display reasoning chains used in any of the other Actors and answer user queries about the reasoning used. This capability is critical to the knowledge engineering process. If the reasoning process that led to a conclusion can be examined, mistakes can be identified and corrected. This will be particularly important in fine-tuning the Diagnose and Predict Actors.

We will also be improving current Actors. We are in the process of adding more knowledge to the Director so that task priority assignment and conflict resolution can be performed. We are adding more knowledge to the existing Diagnose Actors

so that they can diagnose a greater number of faults in their respective LRUs and do so with more accuracy. The amount of data handled by the Monitor is being increased. Finally, the graphics capabilities of the Display Actor are being enhanced.

Interfaces to other AI/KBS languages are another important capability being addressed. A CommonLisp interface will be constructed which should provide access to any CommonLisp based tool (e.g., Art,* KEE,** Knowledge Craft†). This interface will expand the variety of inference engines available for use, significantly increasing the ability of our prototype to address highly complex diagnostic tasks. If we find the need for computational logic capabilities, we may develop an interface to Prolog. Interfaces to other languages may be desirable: for example, we may interface to C to take advantage of the C version of Art or the NASA Clips inference engine.

Finally, we intend to adopt Ada-based inference engines as they become available, moving towards a full Ada application. Of particular interest is the proposed Ada-based versions of Art, Ops5, and Clips. We also foresee the possibility of building our own Ada inference engines as we identify those that are most effective for the KNOMES prototype.

5. CONCLUSIONS

Ada packages that support knowledge-based systems are not readily available yet. However, many features of Ada do support some of the requirements of knowledge-based systems. True concurrent processing seems more feasible for Ada because of its tasking mechanisms. Ada also supports object-oriented programming via packages, generics, and data/process abstraction.

Until AI/KBS packages are available for Ada, it is reasonable to work in both Ada and AI/KBS languages. We use Ada for the more traditional programming segments of our prototype. This decision is important because 30 to 70 percent of AI/KBS systems are standard algorithmic or procedural code.⁵ Ada's software engineering emphasizes well-structured and well-documented standard code. Retaining AI/KBS languages for the knowledge-based portion of the prototype has also proven worthwhile. Knowledge-based systems and the languages used to implement them are inherently easy to update. This flexibility is important for ill-structured problems such as those in our prototype application. Using the AI/KBS languages, we can experiment and try different combinations of knowledge until we find an acceptable solution.

We have found there are extra benefits to using the Actors architecture, especially in terms of code verification. Ada's

code structure is highly testable and verifiable. Because most of our prototype is written in Ada, it can be verified to a large extent using conventional methods. However, the Actors architecture also yields a more verifiable knowledge-based segment. An Actors knowledge base is highly modular. Knowledge is allocated across many Actors, each having a relatively small knowledge base. Small knowledge bases are inherently easier to trace and verify. Finally, as the knowledge-based component of an Actor is repeatedly exercised, algorithmic solutions to parts of the reasoning process sometimes become apparent. These algorithms can be readily migrated over to the Ada section of an Actor. Thus, the knowledge-based component of the prototype can be constantly minimized and the Ada portion maximized, the result being a more verifiable system.

REFERENCES

1. Space Station Definition and Preliminary Design (WP-02), Preliminary Analysis and Design Document (DR-02), Book 12, Data Management System (Section 4.9), Technical Report, McDonnell Douglas Astronautics Company, MDC H2028, December 1985.
2. Space Station Definition and Preliminary Design (WP-02), Preliminary Analysis and Design Document (DR-02), Book 18, Operations Management System Software (Section 4.14), Technical Report, McDonnell Douglas Astronautics Company, MDC H2028, December 1985.
3. R. L. Schwartz and P. M. Melliar-Smith, *On the Suitability of Ada for Artificial Intelligence Applications*, SRI International, July 1980.
4. R. F. Rosen, *Knowledge Representation in Ada*, MDC Internal Memorandum A3-370-AEDO-85-RFR-33, June 1985.
5. D. Naedel, "Ada and Embedded AI," *Defense Electronics*, April 1986.
6. D. C. Dietz, "AdaLISP: A Tool For Artificial Intelligence Implementation," in *Proceedings of the IEEE 1984 National Aerospace and Electronics Conference*, Vol 2, IEEE 1984.
7. D. B. LaVallee, "An Ada Inference Engine for Expert Systems," in *Proceedings of the First International Conference on Ada Programming Language Applications for the NASA Space Station*, Houston, June 1986.
8. G. Booch, *Software Engineering with Ada*, The Benjamin/Cummings Publishing Company, 1983.
9. M. S. Frank and R. P. Knackstedt, "Using Ada to Implement the Operations Management System as a Community of Experts," presented at *The First International Conference on Ada Programming Language Applications for the NASA Space Station*, Houston, June 1986.

*Art is a trademark of Inference Corporation.

**KEE is a trademark of Intellicorp.

†KnowledgeCraft is a trademark of Carnegie Group.