S7-63

p-14

N96-12920

A PC Based Fault Diagnosis Expert System

Christopher A. Marsh
The MITRE Corporation
1120 NASA Road One
Houston, Texas 77089

Abstract

The Integrated Status Assessment (ISA) prototype expert system performs system level fault diagnosis using rules and models created by the user. The ISA evolved from concepts to a stand-alone demonstration prototype using OPS5 on a LISP Machine. The LISP based prototype was rewritten in C and the C Language Integrated Production System (CLIPS) to run on a Personal Computer (PC) and a graphics workstation. The ISA prototype has been used to demonstrate fault diagnosis functions of Space Station Freedom's Operation Management System (OMS). This paper describes the development of the ISA prototype from early concepts to the current PC/workstation version used today and describes future areas of development for the prototype.

Introduction

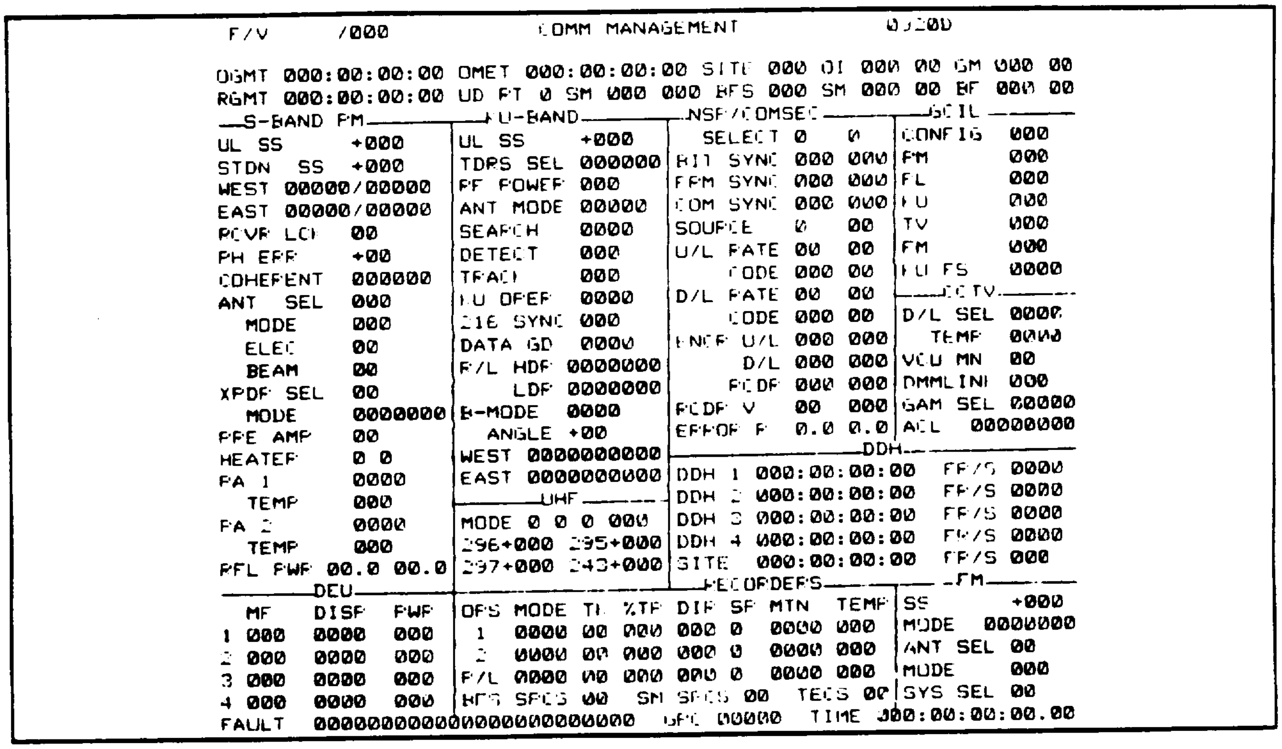
The Integrated Status Assessment (ISA) expert system is a fault diagnosis system that has moved from a concept to the integration phase of development. It started out as a demonstration prototype to help develop Operations Management Application (OMA) requirements for Space Station Freedom and not as a delivery product. The ISA has gone beyond its early demonstration prototype to an integrated field prototype to help answer operations and integration issues. The ISA will continue to evolve as a research prototype and it will be used to influence the development of a delivery fault diagnosis system for Space Station Freedom.

Concepts

In 1985 the Mission Operations Directorate (MOD) asked the MITRE Corporation to help develop requirements for system management of Space Station Freedom. The MITRE task addressed Space Station systems' control and monitoring. It helped develop concepts and requirements for the management of Freedom's onboard systems. The focus was to define the interfaces among the integrated systems management functions and the interfaces between integrated systems management and the individual core systems [1]. This task lead to the development of requirements for the Operations Management System (OMS) that performs the integrated systems management function for Space Station Freedom.

In developing the System Management concepts, the example of Space Shuttle flight control was used as a model. The Space Shuttle is managed on the ground by a flight director responsible for the overall mission, several front room flight controllers each responsible for a different system, and many back room controllers who each support a front room controller. This approach has been used since the early days of spaceflight and is manpower intensive.

Prior to 1989, flight controllers spent much of their time watching screens full of changing numbers representing sensor readings onboard the Space Shuttle (refer to Figure 1 for a typical flight controller screen). When a fault was detected, the flight controller refers to the malfunction procedures and flight rule books to guide them through the isolation of the fault and the reconfiguration of the shuttle. Each of these books were hundreds of pages long and sat in book shelves behind each controller. In addition to the books on the ground the astronauts carried hundreds of pounds of material in the Flight Data File (FDF) on each flight to guide them on the operation of the Shuttle. Failure analysis on the Shuttle was very manpower intensive and automation could play a major role in supporting the flight controllers. Much has been learned from the way systems management and failure analysis was done on the Shuttle for the Space Station Freedom program.



```
 F/V       /000                 COMM MANAGEMENT                 0J20D

OGMT 000:00:00:00 OMET 000:00:00:00 SITE 000 OI 000 00 GM 000 00
RGMT 000:00:00:00 UD RT 0 SM 000 000 BFS 000 SM 000 00 BF 000 00
__S-BAND PM___________KU-BAND__________NSP/COMSEC___________GCIL______
UL SS        +000    |UL SS       +000  |  SELECT 0     0  |CONFIG     000
STDN  SS     +000    |TDRS SEL  000000  |BIT SYNC  000 000 |PM         000
WEST 00000/00000     |RF POWER  000     |FRM SYNC  000 000 |PL         000
EAST 00000/00000     |ANT MODE  00000   |COM SYNC  000 000 |KU         000
RCVR LCK     00      |SEARCH    0000    |SOURCE    0   00  |TV         000
PH ERR       +00     |DETECT    000     |U/L RATE  00  00  |FM         000
COHERENT     000000  |TRACK     000     |     CODE 000 00  |KU FS      0000
ANT  SEL     000     |KU OPER   0000    |D/L RATE  00  00  |____CCTV_______
  MODE       000     |216 SYNC  000     |     CODE 000 00  |D/L SEL   0000
  ELEC       00      |DATA GD   0000    |ENCR U/L  000 000 |  TEMP    0000
  BEAM       00      |R/L HDR 0000000   |      D/L 000 000 |VCU MN    00
XPDR SEL     00      |    LDR 0000000   |     RCDR 000 000 |DMMLINI   000
  MODE       0000000 |R-MODE   0000     |RCDR V    00  000 |GAM SEL 00000
PRE AMP      00      |  ANGLE  +00      |ERROR R   0.0 0.0 |ACL    00000000
HEATER       0 0     |WEST 0000000000   |____________DDH______________________
PA 1         0000    |EAST 0000000000   |DDH 1 000:00:00:00   FR/S 0000
  TEMP       000     |________UHF_______|DDH 2 000:00:00:00   FR/S 0000
PA 2         0000    |MODE 0 0 0 000    |DDH 3 000:00:00:00   FR/S 0000
  TEMP       000     |296+000  295+000  |DDH 4 000:00:00:00   FR/S 0000
RFL PWR 00.0 00.0    |297+000  243+000  |SITE  000:00:00:00   FR/S 000
_________DEU_________|__________________RECORDERS__________|_____FM_______
  MF   DISP   PWR    |OPS  MODE TK  %TP DIR SP MTN   TEMP  |SS         +000
1 000  0000   000    | 1   0000 00  000 000 0  0000  000   |MODE   0000000
2 000  0000   000    | 2   0000 00  000 000 0  0000  000   |ANT SEL  00
3 000  0000   000    |P/L  0000 00  000 000 0  0000  000   |MODE      000
4 000  0000   000    |BFS SPCS 00   SM SPCS 00   TECS 00   |SYS SEL  00
FAULT    0000000000000000000000000000   GPC 00000   TIME 000:00:00:00.00
```

Figure 1. Typical Flight Controller Screen

The concepts developed from studying the Space Shuttle systems management were for more automated systems management functions. A systems hierarchy similar to the personnel hierarchy used for Space Shuttle flight control was envisioned for Space Station Freedom. This systems management hierarchy for Space Station Freedom would have multiple levels. The top level would be Integrated System Management (ISM). ISM would perform systems management at the highest level across all systems. This level of management is similar to the management performed by the astronauts on the Shuttle and controllers in the Flight Control Room. Each system would have its own System Management Application (SMA). Each subsystem within a system would have its own System Operations Application (SOA). Management at the lower two levels is similar to what is performed by the controllers in the Multipurpose Support Rooms and the other support rooms. It was recognized that systems management relies upon knowledge of the systems and their operation. Figure 2 shows the system management hierarchy for Shuttle and Space Station Freedom. As experience with Space Station operations is gained, the applications performing systems management would be refined to incorporate the new knowledge.

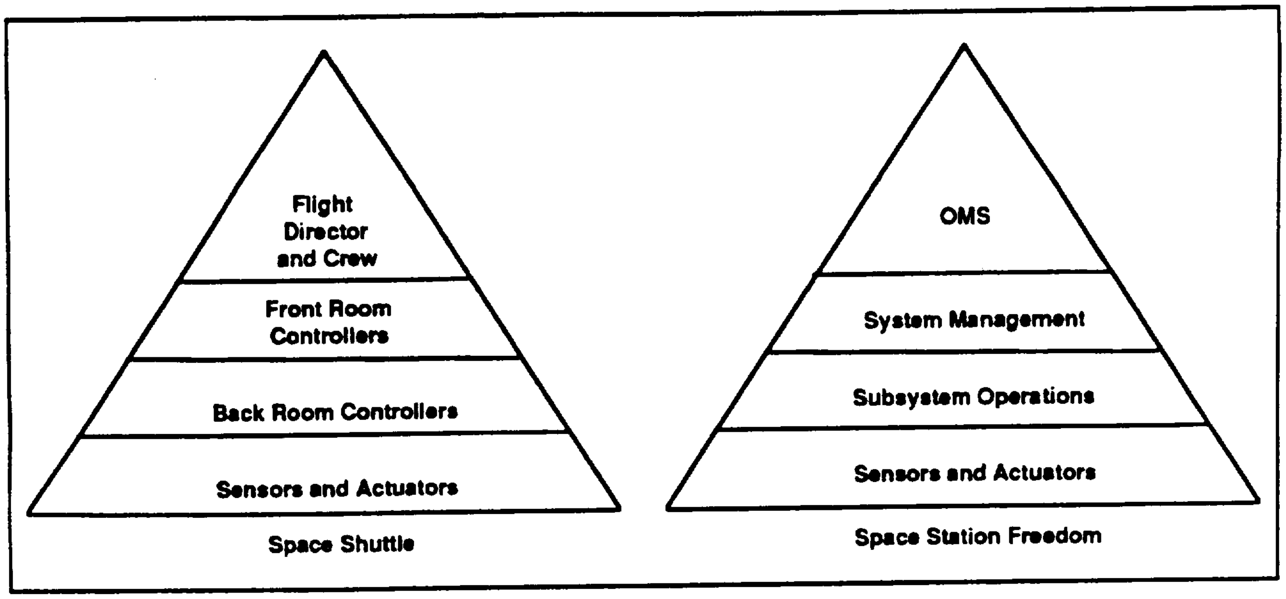


Figure 2. System Management Hierarchy

After the initial concepts were developed, the Operations Management System (OMS) working group was formed to define requirements for the top level of system management (originally called ISM) now called the OMS. Because of its position in the systems management hierarchy, the OMS would integrate operations across all the Station systems and elements. It would include onboard automation (the Operations Management Application (OMA)), ground based automation (the Operations Management Ground Application (OMGA)) , onboard manual operations (the flight crew), and ground based manual operations (the ground controllers). Figure 3 shows the division of the OMS. The OMS would perform high level functions including global planning, system/payload testing, command management, inventory management, maintenance, and training. The intention of the OMS was not to eliminate the work of humans, but to enhance it.

The OMS has been baselined as part of the Space Station Freedom program. Requirements of the OMS incorporated most of the SMA and SOA ideas as tier II and tier III system and subsystem managers.

Once the initial concepts and requirements were documented, the development of prototypes to further develop the concepts was started.

Demonstration Prototype

This effort involved the development of several prototypes of the systems management functions. These prototypes included the Integrated Status Assessment (ISA), Planning Support Environment (PSE), the Procedures Interpreter (PI), On-Orbit Maintenance (OOM), and Communications and Tracking System Management (C&T-SMA). These prototypes were used to demonstrate new concepts, educate the users about emerging expert system technology, and to further develop requirements for the OMA through comments and feedback from the user community. The prototypes were also used to assess proposed designs for OMS implementation. The ISA and PI prototypes were further developed to test OMA concepts in a test bed environment while the others were used only for gathering initial requirements. The following paragraphs describes the development of the ISA.

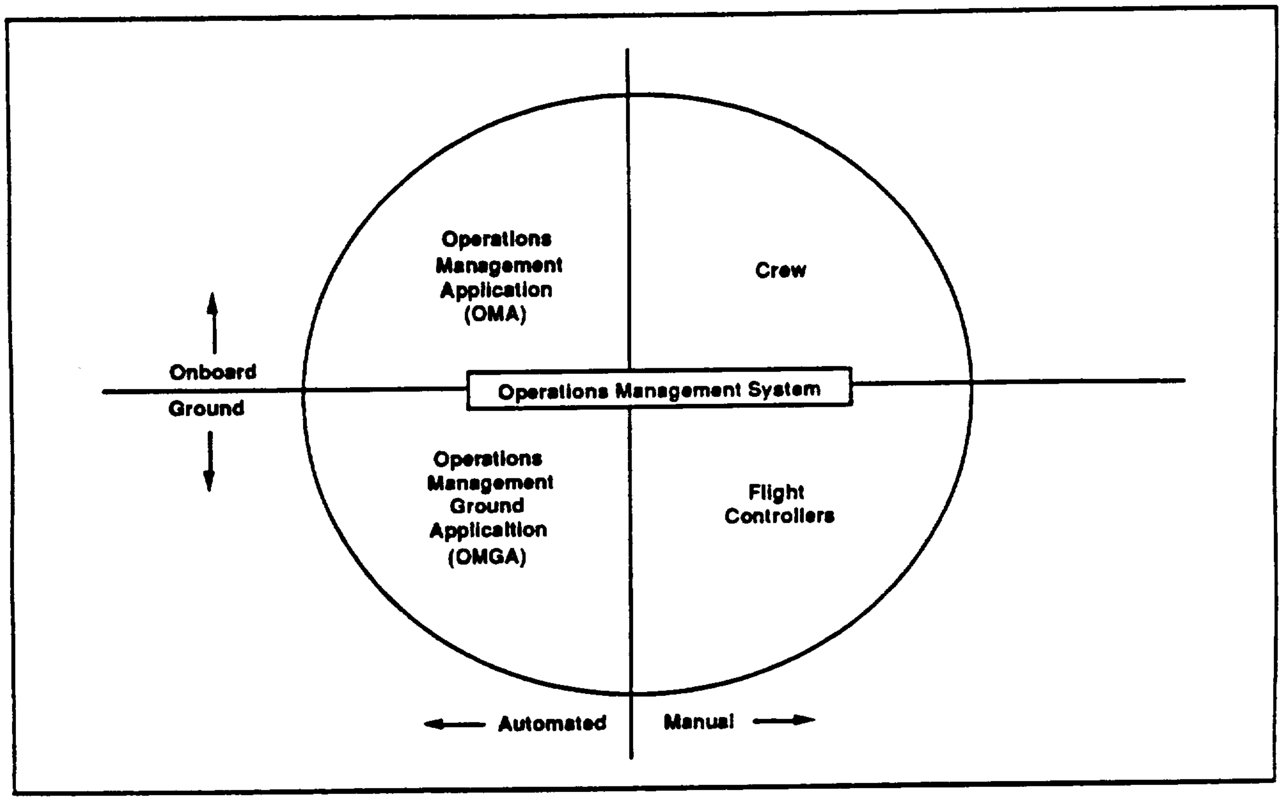

Operations Management Application (OMA)
Crew
Onboard
Ground
Operations Management System
Operations Management Ground Applicaltion (OMGA)
Flight Controllers
Automated
Manual

Figure 3. Operations Management System Components

The ISA project was a one person task that included attending working group meetings, requirements development, and briefings and demonstrations as well as prototype development. The main purpose of the ISA task was to introduce technology and educate the user community, develop requirements and not to do expert systems research.

The ISA prototype was developed to illustrate several functions. The ISA demonstrated the concept of gathering data from the various Space Station Freedom systems. It displayed the data in a coherent integrated manner with a user friendly graphical interface. In failure situations, the ISA showed how expert systems and advanced user interfaces could be used to determine the cause of the problem with a trace of its reasoning and possible recommendations [2].

Because the task of assessing the status of space vehicles is a complex job that requires "expert" knowledge to find heuristic solutions to problems, an expert system approach was chosen to prototype the ISA system. For the initial demonstration prototype, the ISA system was hosted on a Symbolics™ 3600 series computer and written in LISP and OPS5.

The knowledge engineering process took several months for the initial domain of the ISA prototype. The expert spent about a half day per week critiquing and suggesting changes to the system. More of the expert's time would have been useful but this was not possible. After most of the knowledge engineering process was completed, an elaborate user interface was added to the system. Next many other operations people were shown the prototype and their ideas and knowledge was later used to further refine the system. Requirements learned from this process were input to the OMS working group and later turned into Space Station Freedom requirements.

™Symbolics is a trademark of Symbolics Incorporated

The initial domain for the ISA prototype was the communications and tracking KU band system. This included the power busses, cooling loops, Tracking Data Relay Satellite System (TDRSS), and the interface to the DMS. This area was chosen because experts were available in this area and because it contained the intersections of several systems; a fault in one system could cause other systems to malfunction.

The ISA prototype is a rule- and model-based expert system that demonstrates Space Station Freedom fault detection and isolation. The ISA consists of a knowledge base, an inference engine, and a user interface. The knowledge base consists of facts and rules. The facts contain a high level qualitative model of Space Station Freedom. The rules consist of generic fault isolation knowledge and system specific knowledge to determine the source of faults. The inference engine controls how the knowledge base interacts with itself. The user interface gives the user overall control and allows the developer to examine and easily modify the knowledge base. The user interface gives the user overall control and allows the developer to examine and easily modify the knowledge base. Figure 4 contains a diagram of the system.

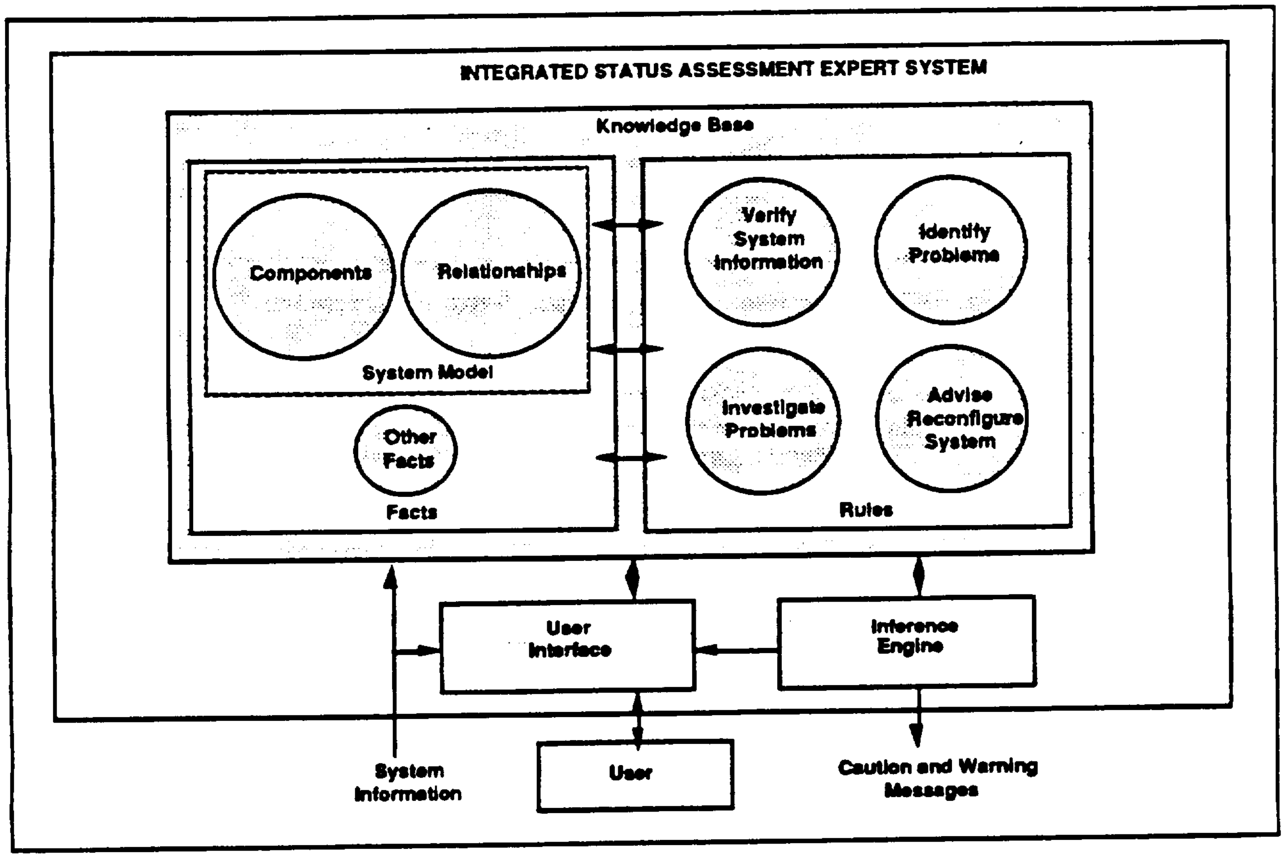


Figure 4. ISA Prototype Components

The ISA operates in an update, run, and react cycle. New operational data from a simulation file is input into the ISA and these values are used to update the model. If some of the operational data is off nominal, them the ISA rules are run on the model to isolate the source of the fault [3]. Once the fault is isolated, additional rules may react to safe Space Station Freedom. After the rules are through firing, the ISA can accept new data from the simulation file to update the model again.

At the end of the first demonstration phase of development, the ISA prototype was built with portions of the Communications and Tracking System, the Electrical Power System, the Data Management System, and the Environmental Control and Life Support System modeled. It has been demonstrated to many groups at NASA including the astronauts, flight controllers, engineers, and many different Space Station contractors. At each demonstration new ideas to improve the prototype and drive new requirements were solicited. These ideas and requirements were incorporated into NASA documentation and the OMS definition. The OMS definition document was incorporated into the Space Station Architecture Control Documents (ACDs).

Transition to a Test Bed Environment

The ISA prototype has been moved onto the Data Management System (DMS) Test Bed Through the OMS integration effort. The OMS integration effort is bringing many Space Station Freedom prototypes and simulations together to form a field environment for testing and developing OMS concepts. Moving applications in the OMS integration effort allows them to be run in as close to a real world environment as is possible. This is a necessary transition environment to test operations concepts with the applications. For the ISA prototype, this took place in two main phases: integration of the Symbolics LISP based prototype and integration of a rewritten C based prototype. Figure 5 shows the evolution path for the ISA prototype.

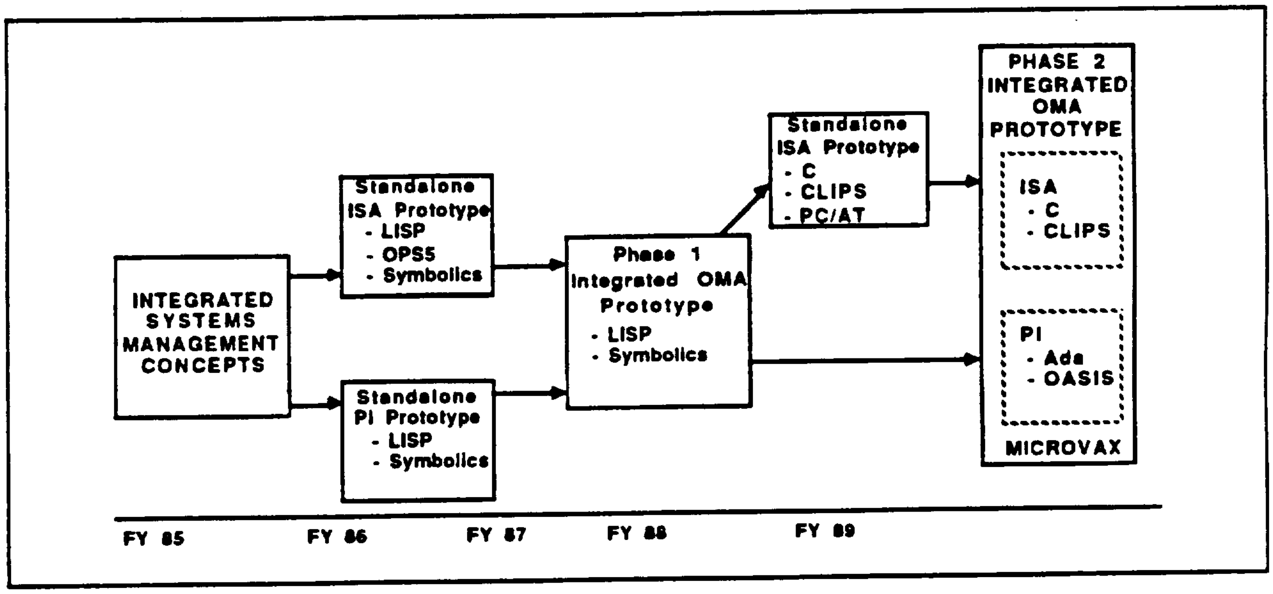

INTEGRATED SYSTEMS MANAGEMENT CONCEPTS
Standalone ISA Prototype - LISP - OPS5 - Symbolics
Standalone PI Prototype - LISP - Symbolics
Phase 1 Integrated OMA Prototype - LISP - Symbolics
Standalone ISA Prototype - C - CLIPS - PC/AT
PHASE 2 INTEGRATED OMA PROTOTYPE
ISA - C - CLIPS
PI - Ada - OASIS
MICROVAX
FY 85
FY 86
FY 87
FY 88
FY 89

Figure 5. Evolution Paths for ISA and PI

OMS Integration

There are many Space Station Freedom systems being modeled and simulated in test beds across the country. Many of these test beds are tied together by the Data Management System (DMS) Test Bed as part of the OMS integration effort. The OMS integration effort provides an environment to test and develop OMA concepts.

As part of the OMS integration effort, the OMA node is the focal point on the DMS network for demonstrating integrated operations of Space Station Freedom systems, with system simulations being represented on various nodes on the Test Bed. Phase One of this integration demonstrated the OMA (represented by the integrated ISA and PI prototypes) controlling and monitoring the Guidance, Navigation, and Control Emulation Laboratory in the execution of a reboost procedure [4]. Phase Two of the OMS integration effort adds four more simulations to the scenario: the Operations Management Ground Application; Generic Electric Power Distribution and Control; Communications and Tracking; and Thermal Control. Future plans include the incorporation of payload and life sciences nodes at the Johnson Space Center and the Marshall Space Flight Center and a data generating node from the European Space Agency. The DMS Test Bed is evolving to be closer to a real Space Station Freedom environment with real world scenarios. Figure 6 shows the DMS Test Bed configuration. In the future the DMS Test Bed will change to real Space Station Freedom like hardware using real DMS communications services.

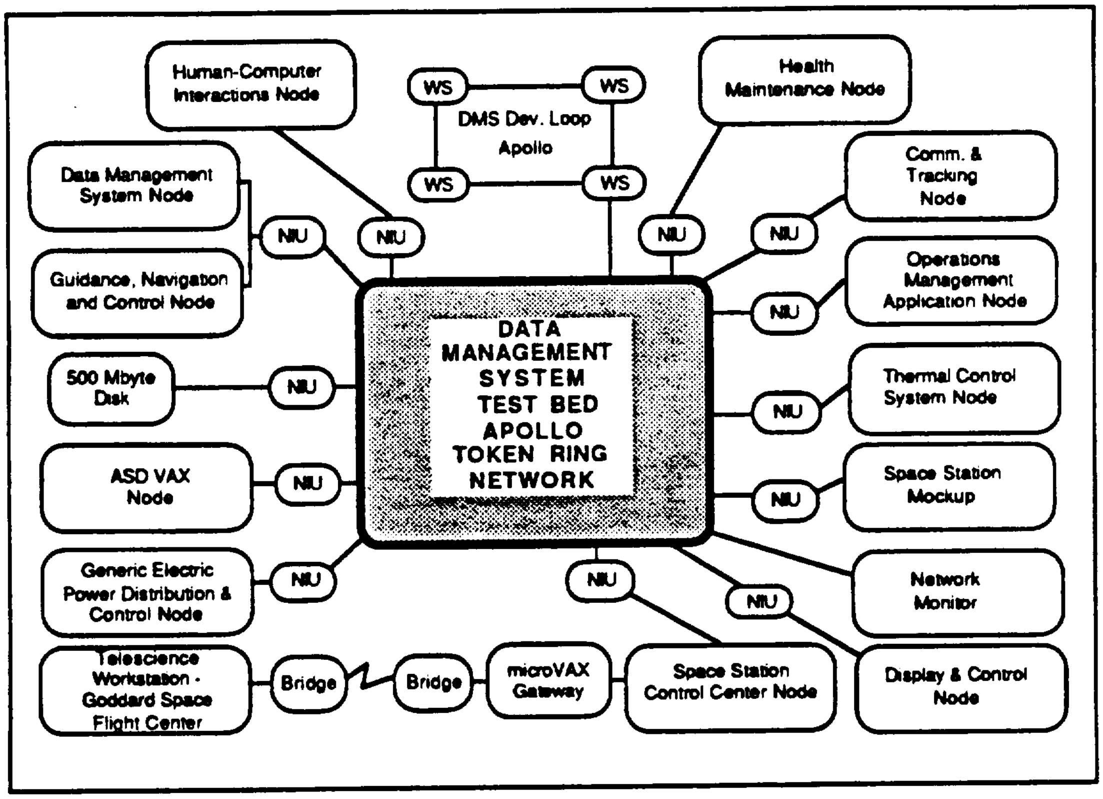
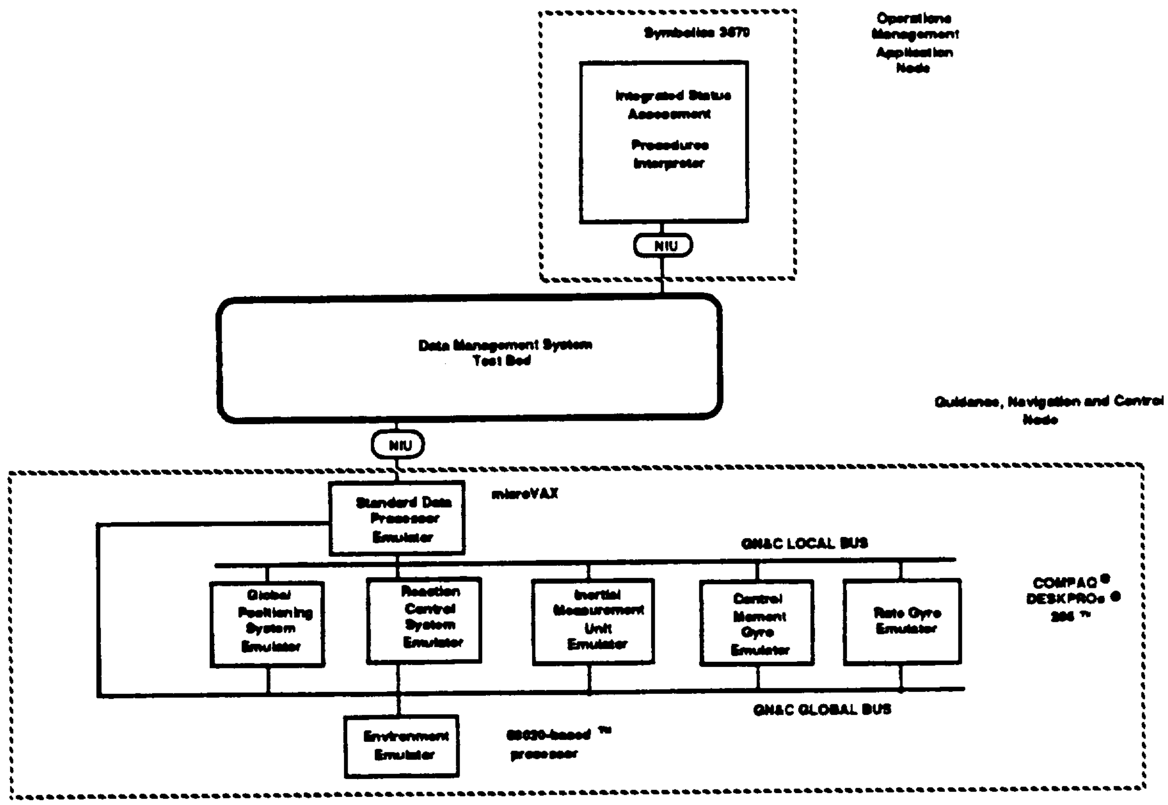

Human-Computer Interactions Node
WS
WS
DMS Dev. Loop Apollo
WS
WS
Health Maintenance Node
Data Management System Node
Comm. & Tracking Node
Guidance, Navigation and Control Node
NIU
Operations Management Application Node
DATA MANAGEMENT SYSTEM TEST BED APOLLO TOKEN RING NETWORK
500 Mbyte Disk
Thermal Control System Node
ASD VAX Node
Space Station Mockup
Generic Electric Power Distribution & Control Node
Network Monitor
Telescience Workstation - Goddard Space Flight Center
Bridge
Bridge
microVAX Gateway
Space Station Control Center Node
Display & Control Node

Figure 6. DMS Test Bed Configuration

Phase One OMA Integration

The first phase of the integration effort was to integrate the Symbolics based ISA that performed fault diagnosis with the Symbolics based Procedures Interpreter (PI) that executed and monitored procedures. This proved to be an easy task because of the nature of the object oriented Flavors system on the Symbolics. Messages were sent from one prototype to the other to pass information. When the PI needed expert system advice on a problem it would send ISA a "run" message. When the ISA rules fired and concluded that an action was needed by the PI, it would send PI the appropriate message.

Once these two prototypes were integrated to form the OMA prototype, the next step was to integrate with the DMS Test Bed and the Guidance Navigation and Control (GN&C) Integration Laboratory. Figure 7 shows the final configuration for Phase One of the OMA integration. The DMS Test Bed has three software communications services available to users: the Network Operating System (NOS) for low level communication; the Data Acquisition and Distribution Services (DADS) for requesting point to point cyclical data sets; and the Ancillary Data Service (ADS) for requesting a single predefined data set. The DADS and ADS are built on the NOS. Because the Symbolics only had the NOS software available to it, special code had to be written to communicate with the GN&C Laboratory.


Symbolics 3670
Operations Management Application Node
Integrated Status Assessment
Procedures Interpreter
NIU
Data Management System Test Bed
Guidance, Navigation and Control Node
NIU
Standard Data Processor Emulator
microVAX
GN&C LOCAL BUS
Global Positioning System Emulator
Reaction Control System Emulator
Inertial Measurement Unit Emulator
Control Moment Gyro Emulator
Rate Gyro Emulator
COMPAQ® DESKPROs® 386 ™
GN&C GLOBAL BUS
Environment Emulator
68020-based ™ processor

® COMPAQ and DESKPRO

™ 68020

Figure 7. Phase I OMA Integration

The Phase One demonstration showed the execution of a reboost procedure by the PI with faults inserted by the GN&C system. The ISA responded to the faults by advising that the reboost procedure be aborted when appropriate. While the phase one demonstration lacked the depth needed to completely test the ISA, it did integrate it with other systems.

Phase Two OMA Integration

Because the Symbolics hardware did not match well with the proposed Space Station Freedom architecture and because it did not have full support of all the software communication services on the DMS Test Bed, the

®COMPAQ and DESKPRO are registered trademarks of COMPAQ Computer Corporation

™68020 is a trademark of Motorola Incorporated

ISA and PI prototypes were ported to a MicroVAX® computer. The ISA was ported to C and the C Language Integrated Production System (CLIPS), an inference engine written in C. The PI was ported to Ada and the Operations And Science Instrument Support (OASIS) software written by the University of Colorado. The two prototypes communicate through interprocess communication on the MicroVAX. Because of the portability of C and CLIPS it was possible to first port the ISA on a PC based system. The only difference between the PC based ISA and the MicroVAX based ISA is the PC version has its own user interface written in C and the MicroVAX ISA uses OASIS for the user interface. The following sections describe the porting of the ISA system.

LISP to C

All the LISP code from the demonstration prototype was rewritten in C for Phase Two of OMA integration. For the procedural code, this was a straight forward process of rewriting LISP functions in C. For the Object Oriented code this involved creating C data structures and functions to replace the LISP Objects (Flavors and Methods). For the PC version, Graphic libraries and machine level calls to mouse driver routines were used to create a friendly user interface similar to the Symbolics version of the prototype. The models used by ISA are generated using the mouse and pop-up menus and create both graphical schematics and data structures describing the system. The model data structures are translated into CLIPS facts for the rules to reason about. The rules are written in CLIPS and can be controlled through the user interface. Special CLIPS functions were developed to allow the rules to modify the screen as well as the model.

Three difficulties were encountered in rewriting the ISA in C. First, the software tools for C were not as good as the LISP tools on the Symbolics. The Symbolics software development environment contained an integrated editor, debugger, flavor examiner, and compiler. These tools were as a big advantage in writing the various iterations of the original prototype. Other tools were used for the C version that were less capable. The impact of this was that a larger percent of time was spent debugging code. This was not a major problem, however, because most of the software design was completed with the LISP version and most of the coding in C was translation and not totally new code. Currently there are many new software development environments coming on the market for C and other languages and in the future the lack of tools should no longer be a problem for rehosting code from LISP to C.

The second difficulty in recoding the ISA was in building the new user interface for the stand-alone PC version. The Symbolics had its own unique object oriented windowing software, but there was no tool available with the same kind of functionality for the PC. All window functions had to be built from scratch; that took most of the coding time. The user interface was not an issue for the MicroVAX implementation of the ISA because it used OASIS for displays. In the future the use of windowing standards such as X.11 and software libraries to support such standards will reduce the difficulty of this problem.

The third difficulty of the rehosting process was substituting VAX Mailbox interprocess communication for the Flavor Message passing to communicate with the PI prototype. VAX Mailboxes are a system utility available on the MicroVAX to allow two programs to communicate to each other through a buffer in memory. VAX Mailboxes are not as easy to implement in C as messages are in LISP.

OPS5 to CLIPS

All the rules were recoded form OPS5 to use the CLIPS inference engine. CLIPS is a forward chaining production system similar to OPS5 that is written in C. Because the two rule systems are very similar in nature the recoding of the rules was an easy process [5]. All the structure of the rules remained the same

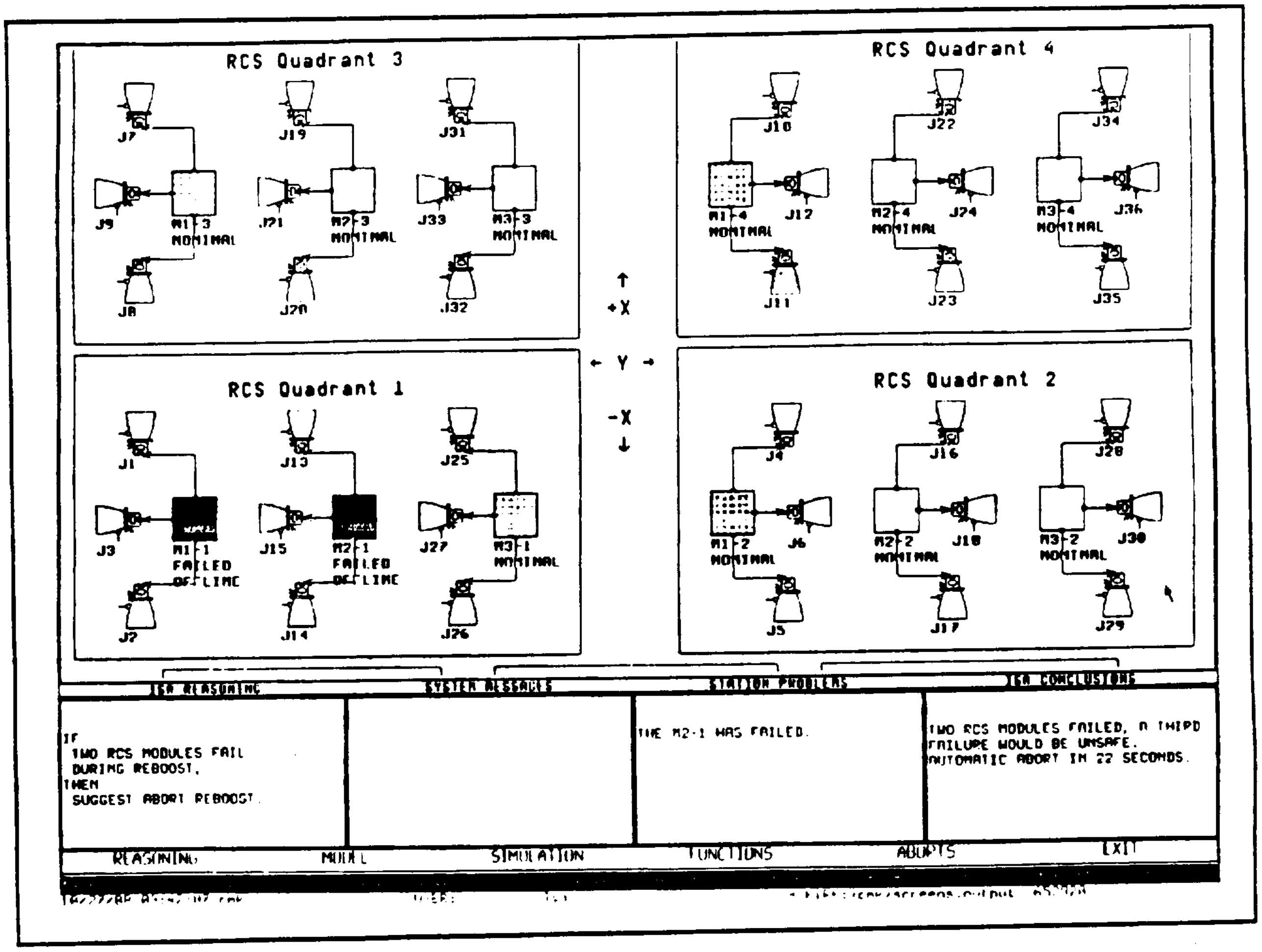
®MicroVAX is a registered trade mark of Digital Equipment Corporation with only minor syntax changes. The only difficulty was working around the lack of objects in CLIPS. This problem is currently being addressed by CLIPS developers for future releases of CLIPS.

Transferring System Models

A large portion of the knowledge in the ISA expert system is contained in the system models. These models contain all the static object knowledge used by the inference engine and all the graphic display information for schematics. Because the models are stored as text files, there was no change required to go from the LISP version of the ISA to the C version of the ISA.

Symbolics Graphics to PC Graphics

The large, high resolution black and white display of the Symbolics proved to be very useful to let the user view the status of systems. Figure 8 show the original Symbolics ISA screen. The PC based ISA has an Enhanced Graphics Adapter (EGA) which supports less resolution than the original so some detail was lost on the new display. The PC version also make use of multiple windows, panning, and zooming to show the user information. Figure 9 shows the ISA with EGA graphics.



IF
 TWO RCS MODULES FAIL
 DURING REBOOST,
THEN
 SUGGEST ABORT REBOOST.

THE M2-1 HAS FAILED.

TWO RCS MODULES FAILED, A THIRD
FAILURE WOULD BE UNSAFE.
AUTOMATIC ABORT IN 22 SECONDS.

Figure 8. Symbolics ISA Screen

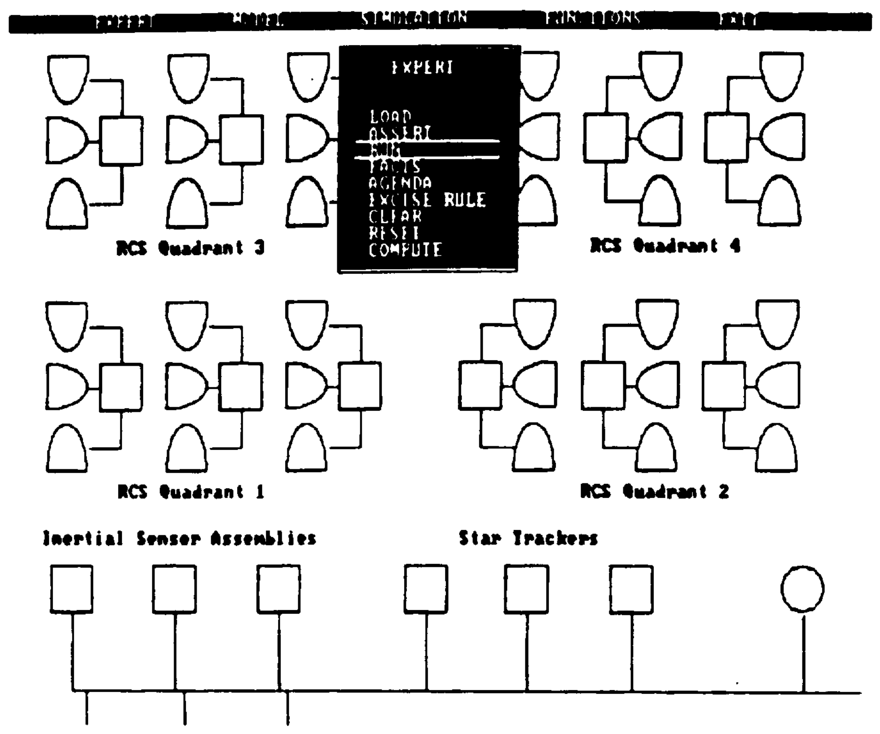


Figure 9. EGA ISA Screen

Phase Two Testing

Phase Two of the OMS integration effort adds four more simulations to the scenario: the Operations Management Ground Application; Generic Electric Power Distribution and Control; Communications and Tracking; and Thermal Control. Future plans include the incorporation of payload and life sciences nodes at the Johnson Space Center and the Marshall Space Flight Center and a data generating node from the European Space Agency. Phase Two will involve diagnosing failures in the Communications and Tracking System and in the Thermal Control System simulations.

While these additional simulations will add robustness to the OMS integration effort it will still lack the integrated closed loop simulation to back them up. What takes place in one simulation is not reflected in the others. The OMS integration effort is still just many separate simulations tied together on a communications link. To really test the ISA in its global fault detection capabilities, the simulations need to be more integrated. This is something that the OMS integration effort project needs to have, not only to test global fault detection capabilities but to test other OMA functions such as global resource management and station planning.

The Future of the ISA Prototype

The ISA system is still a prototype that will continue to evolve and remain a tool to help influence the design of the real OMA fault diagnosis function. The OMS integration effort has to have an integrated simulation capability. The remainder of the OMA functions need to be prototyped in an integrated environment. Deeper reasoning capabilities need to be built into the system. The following sections describe these future steps.

Better Simulation Capabilities

In order to test expert systems on the DMS Test Bed, there needs to be a Test Bed Simulation Coordination Node. This node will exist on the DMS Test Bed (probably as an additional process on existing hardware) and produce DADS data or an ADS data-set to coordinate the integrated demonstrations. Each node will have access to this data-set to get information about the current simulation being run. These are some of the features that such a node could provide: identification of the scenario being run; starting and stopping times of the current demonstration; current demonstration status; the ability to reflect changes on one node into another system; better coordination of integrated and multiple failures; and global resource levels. These features would help present a more coherent integrated simulation and move the Test Bed away from the awkward canned demonstrations we have today to a true testing environment.

Deeper Reasoning Capabilities

Rule based expert systems can perform good fault diagnosis for systems that fail where they are expected to fail. Unfortunately, systems don't always fail in a mode that was predicted. In the Apollo 13 mission to the Moon, two fuel cells were taken out in a mysterious explosion. There were no flight rules to handle such a failure. Because humans can reason much deeper that their written procedures they were able to bring the astronauts home safely. Model based expert systems that reason from first principals use the physical design instead of just rules to perform diagnosis [6]. These systems can diagnosis faults that were not planned for in a purely rule based expert system. The ISA currently uses high level models in its reasoning. To perform better fault detection and isolation, deeper models that contain behavior information will be used in the ISA.

C to Ada

All Space Station Freedom code will have to be written in Ada. Therefore the OMA fault diagnosis functions will have to be written in Ada. What limitations will this have on the system? Are the current generation of Ada compilers efficient enough to work within the hardware limitations and software requirements of Space Station Freedom and support expert systems? These are areas that need to be investigated.

Conclusions

The use of the OMA demonstration prototypes greatly helped illustrate the ideas developed in the concepts stage. These prototypes communicated these ideas to people much more effectively than the traditional concept and requirements documents alone did. Many useful comments and suggestions were gathered during the many demonstrations of the OMA prototypes. These comments and suggestions greatly enhanced the OMS requirements. The demonstrations also helped us gain many allies to support our ideas for a the implementation of the real system and also help give support to bring advanced automation and expert systems into the existing Space Shuttle program.

The rapid prototyping environment of the Symbolics proved to be very efficient for the stand-alone demonstration prototypes. Changes could be easily integrated into the ISA prototype in just a few minuets. Similar tools would have been very useful in later implementations of the ISA.

The use of C as the language for the integrated prototype was good because it allowed the ISA to be easily tied into CLIPS. The use of C and CLIPS allowed for the ISA to run on a multiple hardware platforms from a PC to a MicroVAX.

Making the software data driven from text files made the transition from LISP to C much easier and made it easy to introduce changes in both the knowledge base and the user interface. OPS5 and CLIPS both used text files for their rules and had a similar syntax. All the structure of the rules remained the same with only minor editing changes. The only difficulty was working around the lack of objects in CLIPS. A large portion of the knowledge in the ISA expert system is contained in the system models. These models contain all the static object knowledge used by the inference engine and all the graphic display information for schematics. Because the models are stored as text files, there was no change required to go from the LISP version of the ISA to the C version of the ISA.

LIST OF REFERENCES

[1] National Aeronautics and Space Administration (1986), *Space Station Systems Operations Concepts for Systems Management*, JSC-20792, Houston, TX: NASA Johnson Space Center.

[2] Marsh, C. (1988), *The ISA Expert System: A Prototype System for Failure Diagnosis on the Space Station*, The First International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems, June 1-3, 1988, ACM Press, (60 - 74).

[3] Brownston, L., R. Farrell, E. Kant, N. Martin (1985), *Programming Expert Systems in OPS5*, Addison Wesley.

[4] Kelly, Christine M. (1988), *Automated Space Station Procedure Execution*, AIAA 26th Aerospace Science Meeting, Reno, NV, January 1988.

[5] Giarratano, J. C. (1989), *CLIPS User's Guide*, Artificial Intelligence Section, Lyndon B. Johnson Space Center, May 1989

[6] Davis, R. (1984), *Diagnostic Reasoning Based on Structure and Behavior*, Artificial Intelligence volume 24 (ISSN: 0004-3702), Elsevier Science Publishers B.V., Amsterdam, The Netherlands.