PI-in-a-Box: a knowledge-based system for space science experimentation

Richard Frainier•, Nicolas Groleau•, Lyman Hazelton••, Silvano Colombano+, Michael Compton++, Irving Statler+, Peter Szolovits••, Laurence Young••

• RECOM Technologies, Inc., •• Massachusetts Institute of Technology, + NASA Ames Research Center
++ Sterling Software (Federal Systems). Correspondence to: Artificial Intelligence Research Branch, NASA Ames Research Center, m/s 269-2, Moffett Field, CA 94035-1000, USA. E-mail: frainier@ptolemy.arc.nasa.gov

Abstract

The Principal Investigator (PI) in-a-Box knowledge-based system (KBS) helps astronauts perform science experiments in space. These experiments are typically costly to devise and build, and often difficult to perform. Further, the space laboratory environment is unique, ever-changing, hectic, and therefore stressful. The environment requires quick, correct reactions to events over a wide range of experiments and disciplines, including ones distant from an astronaut's main science specialty. This suggests the use of advanced techniques for data collection, analysis, and decision-making to maximize the value of the research performed. PI-in-a-Box aids astronauts with "quick look" data collection, reduction and analysis, and also with equipment diagnosis and troubleshooting, procedural reminders, and suggestions for high-value departures from the pre-planned experiment protocol. The astronauts have direct access to the system, which is hosted on a portable computer in the Spacelab module. The system is in use on the ground for mission training, and has been delivered to NASA for in-flight use on the Space Life Sciences (SLS) 2 Shuttle mission scheduled for August, 1993.

Introduction

The critical resource in astronaut-tended flight experiments is time. The lack of time affects both pre-flight training for, and in-flight operation of, the experiment. This is true currently with the U.S. Space Shuttle program, and will persist with the advent of Space Station Freedom operations. Another key factor in space experimentation is the use of fixed experiment protocols. This major constraint severely limits the ability of an earth-bound scientist to change the course of an experiment even when the data and current situation clearly indicate that it would be scientifically more valuable to do so.

The PI-in-a-Box KBS helps scientist-astronauts do better science in space, given fairly severe time constraints and the need to work in areas outside their main specialty. The goal is to help the astronaut become a scientific collaborator with the ground-based Principal Investigator (PI) who has designed the experiment. The system facilitates this by sharing with the astronaut observations about the quality and importance of the data as it is being collected in-flight. This system has the potential to fundamentally change the way crewmembers interact with ground-based investigators in the Space Station era.

In this paper, we present a logical overview of the system, continue with a description of our first area of application, explain the technical details of the current implementation, and finally share some development philosophy used to manage this multi-year project. This system continues previous work described in (Young et al. 1989), (Haymann-Haber et al. 1989) and (Frainier et al. 1990).

Functional overview

The PI-in-a-Box system has several modules (figure 1). Together they allow the diagnosis of data-collection problems, hypothesis monitoring and formulation (limited to an analysis of "interestingness" in the initial system), determination and scheduling of the experiment's steps, and general-purpose help for the astronaut-user.

The Data Acquisition Module (DAM) and Data Quality Monitor (DQM) acquire data from the experiment (displayed in real-time), extract parameters from the data and interpret them. The DQM also analyzes the data to determine quality with respect to the experimental apparatus and provides results to the Diagnosis and Troubleshooting Module.

The Diagnosis and Troubleshooting Module (DTM) helps the astronaut isolate and recover from experiment data-collection problems. It suggests tests to isolate equipment faults. It also presents recommendations based on a computation of problem severity and possible recovery strategies with respect to remaining experiment session time (i.e., the system can actually recommend that troubleshooting not be performed.).

The Interesting Data Filter (IDF) module monitors data from the experiment passed to it by the DAM. The IDF analyzes the data to determine its fit with pre-flight hypotheses. The fit can be either statistical or heuristic. Deviations are reported as "interesting". These deviations are defined as "needing confirmation", even if not part of the original fixed protocol. Once confirmed, they cease to be interesting.

The Protocol Manager (PM) module generates the best possible experimental protocol for use at any given time in the experiment. It also displays information to, and accepts information from, the user-astronaut (user). Corresponding to these two major functions, the PM has two logical components: the scheduling component called the Protocol Suggester (PS) and the human-computer interface (HCI) component called the Session Manager (SM).

The PS creates a new experimental protocol upon request. A request from the user is likely when:
- there is a predicted shortage of time - possible need to cut steps.
- there is a predicted excess of time - possible need, or opportunity, to add steps.
- the experiment is giving interesting data - possible need to substitute steps that will collect more information about the interesting data.

The SM displays the current state of the experiment including progress against the protocol and elapsed times, and the history of other sessions occurring earlier in the mission. The SM also displays procedural step-by-step checklists of experimental steps to be performed within the experiment by the user. The SM updates the current protocol and elapsed times automatically and in response to user editing. The SM also offers a scratch-pad to allow users to record their observations. Users can perform the following actions using the SM:
- Display the status of the current session. This includes a list of completed steps, the current step, and all pending steps. It also includes temporal information about the session and the current step.
- Display alternative protocols.
- Display the history of other sessions occurring earlier in the mission. This history is a list of all completed steps, including the experimental conditions used for each step.
- Display experiment checklists for a given experiment step.
- Edit the current protocol and all temporal information known, and used, by the system.
- Replace the current protocol with any of the other available protocols.
- Order a new set of protocols for consideration (by calling the PS).
- Initiate an equipment troubleshooting session.

Finally, an Executive module controls module activation and focus-of-attention. It is also used to augment the operating system environment, if necessary, for a particular host CPU.

One further module is planned for future versions of the system: an Experiment Suggester (ES). The ES will work in conjunction with the IDF. Given a new hypothesis from the IDF, the ES will generate a set of tests that can be used to investigate the new hypothesis.

The first domain - vestibular physiology

The system has first been used in conjunction with a life sciences experiment in vestibular physiology known as "The Rotating Dome Experiment". It was devised by one of the co-authors, Prof. Laurence Young, who is the director of the Man-Vehicle Laboratory at the Massachusetts Institute of Technology. The experiment is conducted by one crew member while another one acts as subject. The purpose of the experiment is to understand the human occulo-vestibular system, and its relationship to the phenomenon of space motion-sickness. During the experiment, the subject's visual field is filled by a dome. The dome, which contains a constellation of dots, is rotated at various speeds and directions. This induces "vection", or the sensation that the subject is rotating instead of the dots. Voluntary and involuntary reactions to the vection are measured. There are typically two or three in-flight sessions each involving two to four crew members. An experiment session consists of equipment setup, equipment

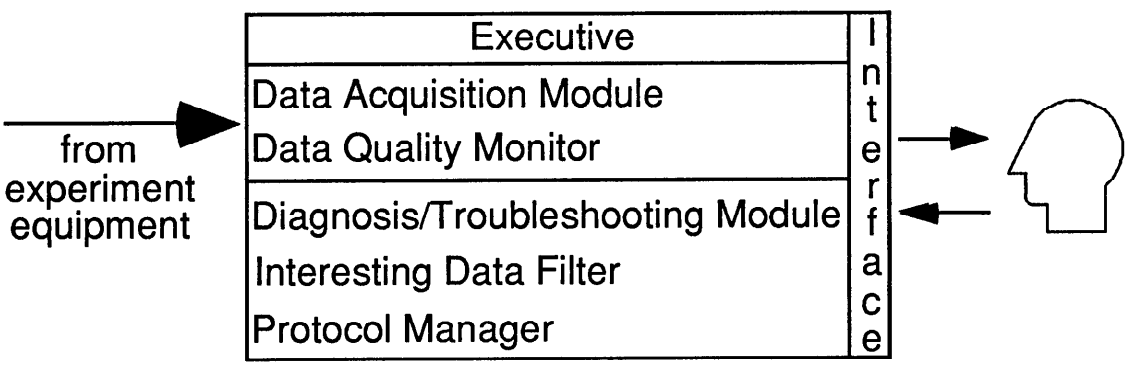

Figure 1: major modules

electrical checkout, several experiment runs on the first subject, introduction of follow-on subjects, and equipment shutdown and stowage. The experiment has been flown in space three times in the past[1] , and will fly again this year (1993). When performed in space, this experiment generates five analog data channels. While the astronauts could monitor two of the five channels in real-time on a small oscilloscope, they are not as expert as the ground-based investigator at validating, or reacting to, the data. Investigators can monitor all of the channels, but the experiment data are subject to delays and outages as they are passed from the shuttle to relay satellites to and through the ground-based telecommunications system. Further, the investigators are quite limited in their ability to change the course of the experiment during an hour-long session. In fact, they are limited in their ability to change the course of the experiment during any remaining in-flight experiment sessions.

PI-in-a-Box has direct access to all five data channels and performs "quick-look" validation and analysis in real-time. The analysis is driven by heuristics compiled from the investigators and the results are communicated to the astronaut performing the experiment. Thus, the system provides the astronaut advice on how to best use the precious time allocated to the experiment. This advice is based on the pre-flight plan, modified by all of the events that have occurred in flight, including a record of the experiment apparatus' performance, the list of crew members that have already performed the experiment, and what the analyzed data indicated about each of these subjects. Specific advice includes:

- recommendations on accepting a degradation of the experiment's data collection, or on spending time to repair a problem. If repair is elected, there is a step-by-step diagnosis/repair plan offered to the user.
- advice on the order in which to test subjects, and the order of individual test steps for a given subject.
- alerts about analyzed data that appear to be of particularly high value ("interesting" data).

There are other features that allow review of previously completed portions of the experiment, and that facilitate planning and/or replanning future experiment sessions. Finally, there are features providing reminders on setting up and using the experiment apparatus.

Some typical scenarios

Let us assume that it is now two days after lift-off. The first session involves two astronauts who will alternate as subject and operator. The system has been set up for the first session but there is a problem. An electrical connection at the junction of two cables is damaged. The problem affects one of the two electromyography data channels. The experiment setup is "on time", but the problem must be addressed. Further, there will be a voice and data outage (LOS) commencing in five minutes that will have a duration of 20 minutes. Without PI-in-a-Box, the crew would typically attempt to repair the apparatus for a while, and then ask the ground for advice if the effort was unsuccessful. If the LOS was in effect, the advice could not arrive until after the 20-minute blackout. With PI-in-a-Box, the crew could ask for a recommendation at any time. In this situation, even if the system had a repair procedure available it would recommend not spending time repairing the low-priority channel, but instead using that time to get data from the scheduled subjects.

Let us now assume that the astronauts declined the recommendation and spent 20 minutes at the repair. They are now part way through the experiment protocol and 15 minutes behind schedule. The astronauts realize that they are going to have to cut the experiment short. Without PI-in-a-Box, the crew would typically work as far along as they could and then cut the last steps of the protocol. In this case, the entire second subject would be eliminated. With PI-in-a-Box, the crew could ask again for a recommendation. Here, the system would recommend cutting the last experiment condition for the first subject, the first experiment condition for the second subject, and then continue with the rest of the experiment. This recommendation takes into account the various setup times and scientific importance of the experiment steps, realizing that a lengthy setup was required for two low-priority steps. Eliminating both the setup and the steps saved 13 minutes and increased the "coverage"[2] of the first session.

Current implementation

As fielded, the system runs on a single Macintosh PowerBook 170, which hosts all six modules. There is one other piece of hardware, an external (GW Instruments) analog-to-digital converter connected to the PowerBook's Small Computer System Interface (SCSI) port. The PowerBook is fitted with 8 MB RAM (the maximum available on that model) and a 40 MB internal hard drive.[3]

---

[1] The Rotating Dome Experiment flew on Space Shuttle-hosted SpaceLab missions SL-1 (in 1983), D-1 (in 1985), and SLS-1 (in 1991).

[2] The coverage heuristic states that it is better to have at least some data on each subject than to have a full set on runs on one subject with no data on another.

[3] All materials and equipment used on and in the Space Shuttle require a "qualification" (analysis and test for safety, reliability, etc.). These tests need to be performed in advance of the mission's Critical Design Review. It was not possible to "flight qualify" more recent (and more capable) versions of the PowerBook for our target mission.

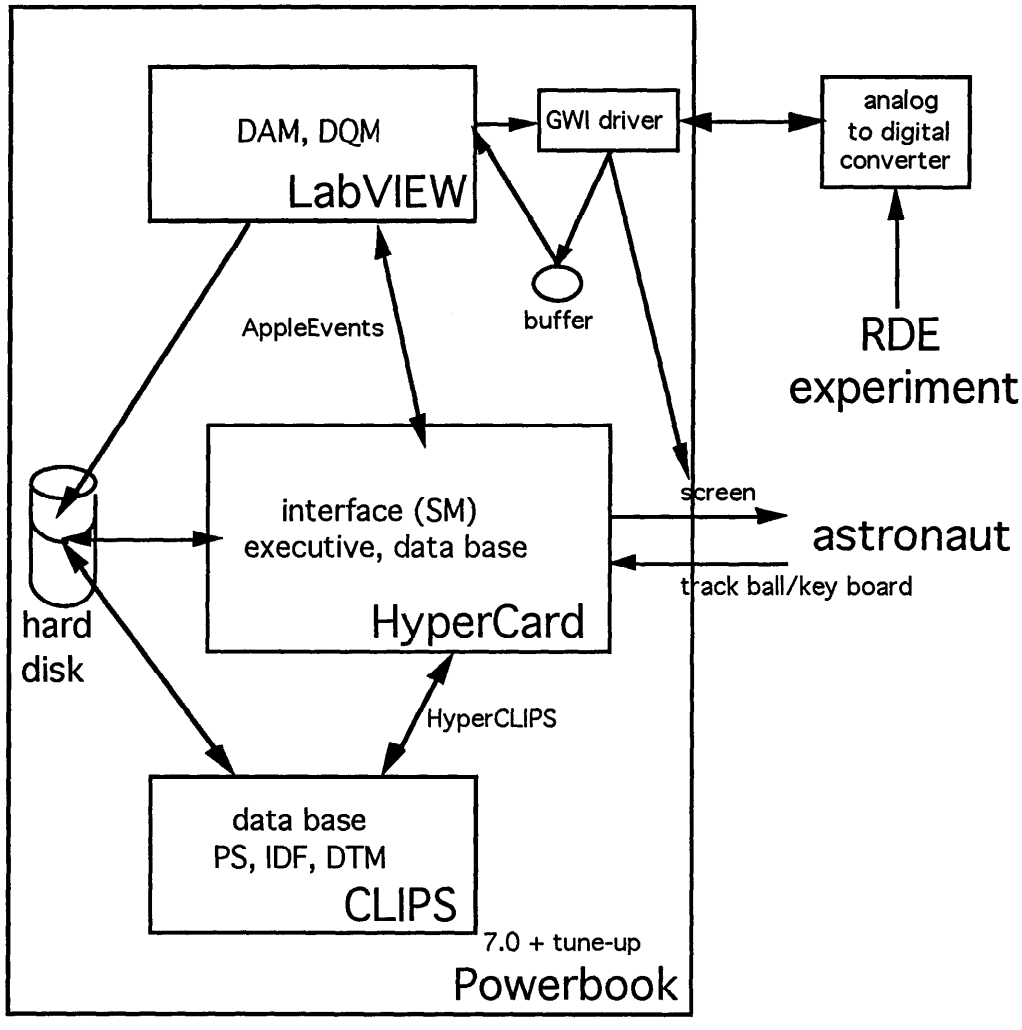

Figure 2: PI-in-a-Box physical-logical mapping

Three main software tools were used; CLIPS, LabVIEW, and HyperCard.

CLIPS, available from NASA/COSMIC, serves as the inference engine for the application. This OPS-style expert system shell is used by the system for schedule repair (PS), diagnosis (DTM), and symbolic analysis (IDF). Key factors indicating its use included: low cost, availability of source code (facilitating tool extension and customization), excellent support, continuous upgrades, a strong user group, and widespread use.

LabVIEW, available from National Instruments, controls data collection, reduction, validation and archival. It is a graphical data flow language ("software from pictures") with a mouse-oriented developer's interface and excellent browsers. These features facilitated rapid development and code reuse. Another key feature is its support for data acquisition and analysis in a single package. Other key factors indicating its use included: excellent support, continuous upgrades, availability of runtime version, a strong user group, and widespread use.

HyperCard (available from Apple) is used for the HCI, overall data management within the system, and module activation. Its procedural scripting language and part-whole object hierarchy facilitated the "rapid prototyping" style of HCI construction essential to our system's development. Other key factors indicating its use included: low cost, widespread use, and good technical support from Apple.

The three tools communicate with each other using AppleEvents and HyperCLIPS. HyperCLIPS, a set of two simple one-way drivers, was developed by our team for HyperCard-CLIPS interapplication communication. "AppleEvents" is an interapplication communication feature of the "System 7" version of the Macintosh Operating System. CLIPS and LabVIEW do not communicate directly with each other.

The system currently uses all 8 MB of RAM. Virtual memory was tested and rejected due to the associated performance penalty. HyperCard and LabVIEW are each allocated 2,500 KB of RAM, CLIPS is allocated 1,800 KB, and the remainder is used by the computer's Operating System. The system files[4] currently occupy about 11 MB of hard disk storage.

The mapping between hardware/software and each logical module of the system, as implemented for the Rotating Dome Experiment, is seen in figure 2. An obvious characteristic of our architecture is the necessity to make complicated technical tradeoffs when building systems which integrate different tools and solve real problems.

The HCI, called SM in our system, has been built as two HyperCard stacks. One stack contains 26 "cards". Fourteen of these serve as a persistent database (and are not viewed directly), and 12 are used for display. The second stack contains 16 "cards" that display pictures, line drawings, and real-time data traces.

The data base is divided between HyperCard and CLIPS. The HyperCard-resident portion consists of 14 cards. Most of these are used to store data from the experiment: one card is used for each step of the experiment that generates data. There is one card used to display summary results of previous experiment sessions. Two cards are used to store data used "globally" by several of the modules. The CLIPS-resident portion consists of about 120 base facts. This number increases as the mission progresses and experiment history is generated. During the typical operation of the PS, there are about 200 facts in the data base at any one time (out of 120 base facts and 400 generated facts).

An illustration of the system in use

Assume that the operator has just set up the laptop computer and analog-to-digital converter in anticipation of an experiment session. The system automatically loads upon power-up of the Macintosh. Two minutes and twenty seconds after power-on, startup is complete. The system

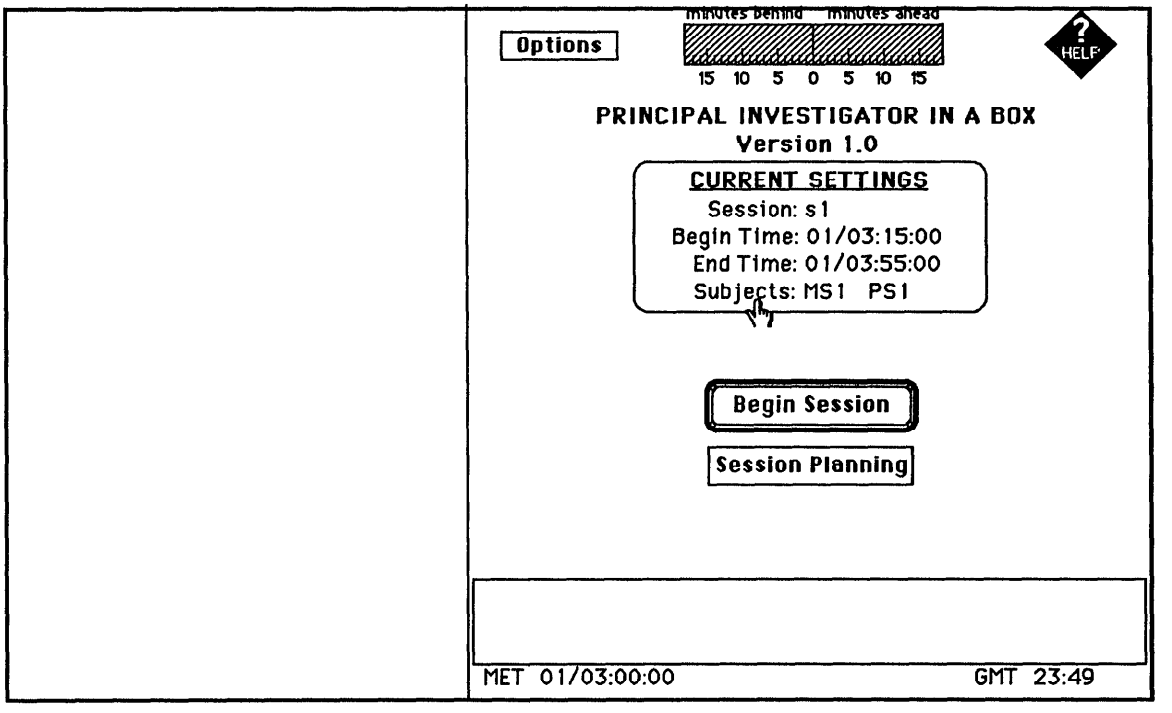

Figure 3: Session startup

[4] Includes documentation text files, rule files, permanent data files, other application files and tool image files.

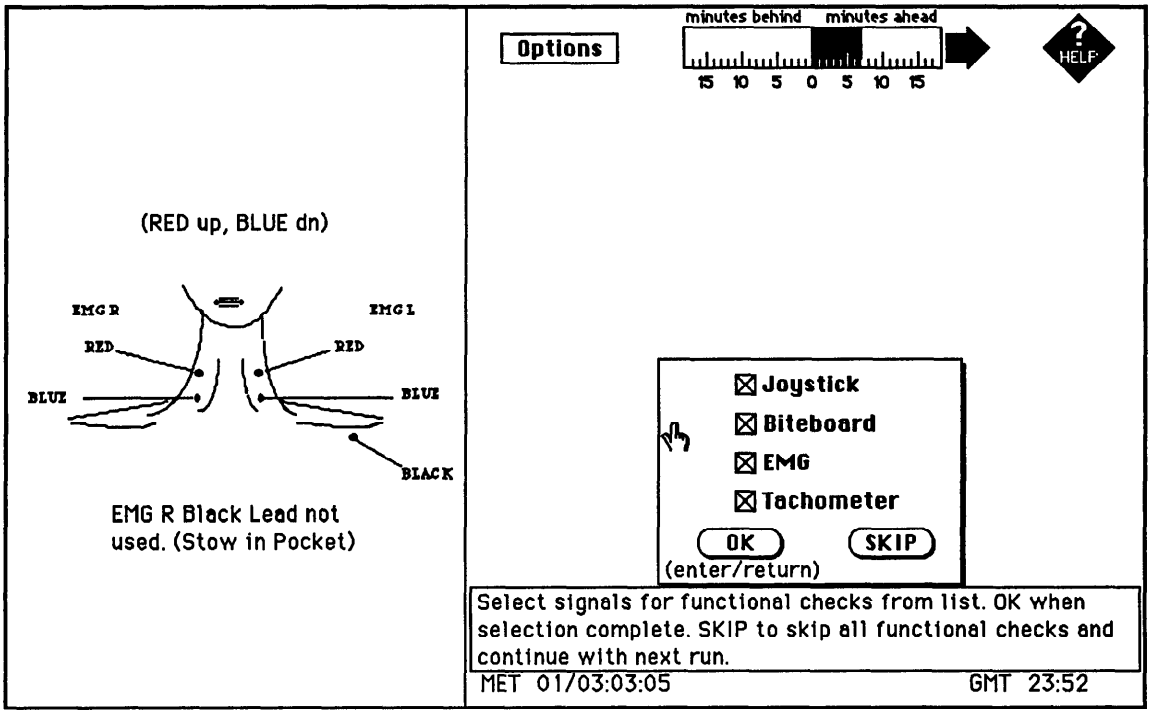

Figure 4: Functional check preparation

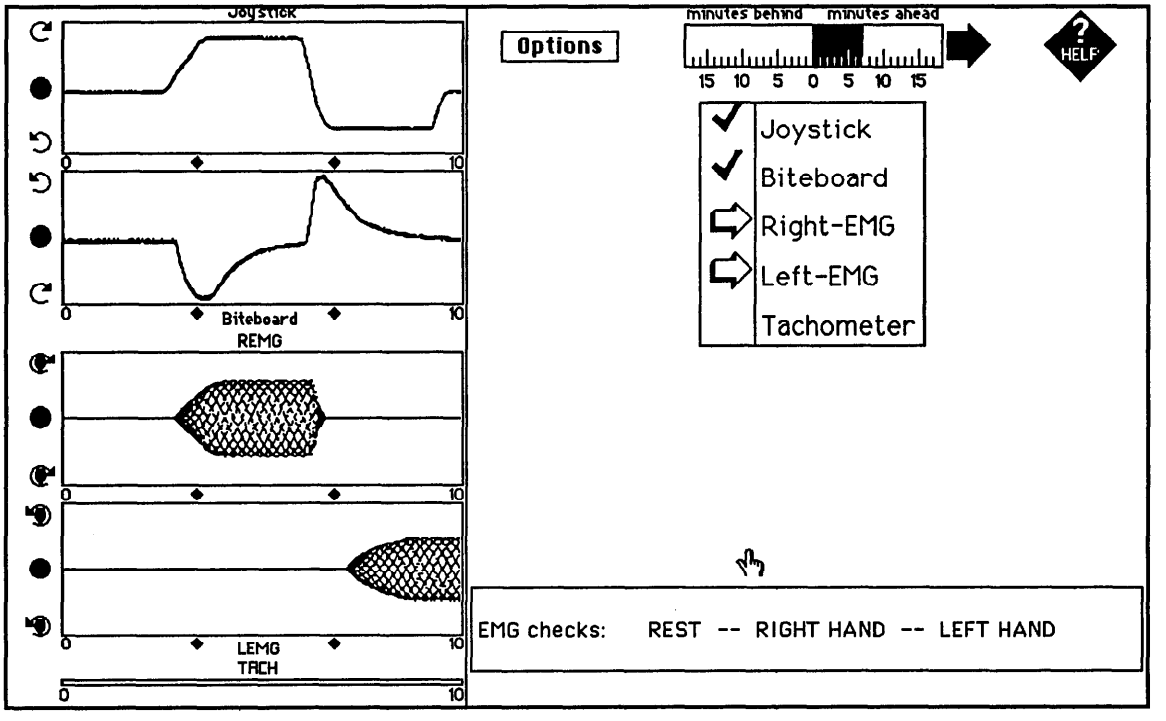

Figure 5: Functional check operation

has selected the next scheduled session, and presents overview information such as the scheduled start time, end time and subjects (figure 3). In this example, there are two subjects, Mission Specialist 1 (MS1) and Payload Specialist 1 (PS1). If the begin time or end time has recently changed, the user communicates the currently scheduled time to the system by clicking over the item to be updated. There is also the ability to similarly change/edit the subject list, as well as other options. When the current settings are completely correct, the user proceeds by selecting "Begin Session".

The system prepares for the next action, a functional checkout of the experiment's electrical outputs. Notice that since the "EMG" functional check should be performed, a graphic is displayed to help assure correct electrode placement (figure 4).

The LabVIEW-based DQM is used for this checkout and autocalibration of the experiment apparatus (These checks typically occur after equipment setup and before each new subject enters the experiment, although they can be optionally performed at any time during a session). The system displays a list of the signals to be checked, with an arrow pointing to the currently-checked signal. As each 10-second check is occurring, a real-time trace of that signal is displayed (figure 5). There are five signal traces that are displayed from top to bottom on the left side of the screen: Joystick, Biteboard, Right-EMG, Left-EMG, and Tachometer. The Tachometer is a "heartbeat" trace that is only 2 pixels high. Note that in figure 5, the Right-EMG and Left-EMG signals are displayed simultaneously as part of the single EMG check, and so there are two arrows indicating the current signal.

The DQM performs one of the more interesting, and challenging, tasks in our system. It was not obvious at first how to interpret and react to a 10-second slice of analog data. It was quickly realized during integration testing that the experiment hardware electrical specifications were not by themselves sufficient to determine if a signal channel was functioning correctly. We settled on partitioning the 10-second test into three segments, a rest-condition segment, a full-positive-deflection segment, and a full-negative-deflection segment. These segments are identified by applying mathematical filtering and differentiation operations. Specific parameters are then calculated for each segment. The analysis of these three segments is combined with known operational and faulty states of the hardware to finally determine the health of a given signal channel. When a signal is identified as operationally OK, then the check further serves as a calibration of the channel and is used by the DAM during its run-data analysis. Then there is a problem, DQM does not always indicate a unique fault. Resolution must then occur later in a troubleshooting session.

Upon completion of all requested checks, a summary of the results are displayed (figure 6). In this example, the experiment apparatus is functioning correctly. If a problem had been seen, then an automatic troubleshooting session (DTM) is initiated, and a recommendation is prepared for

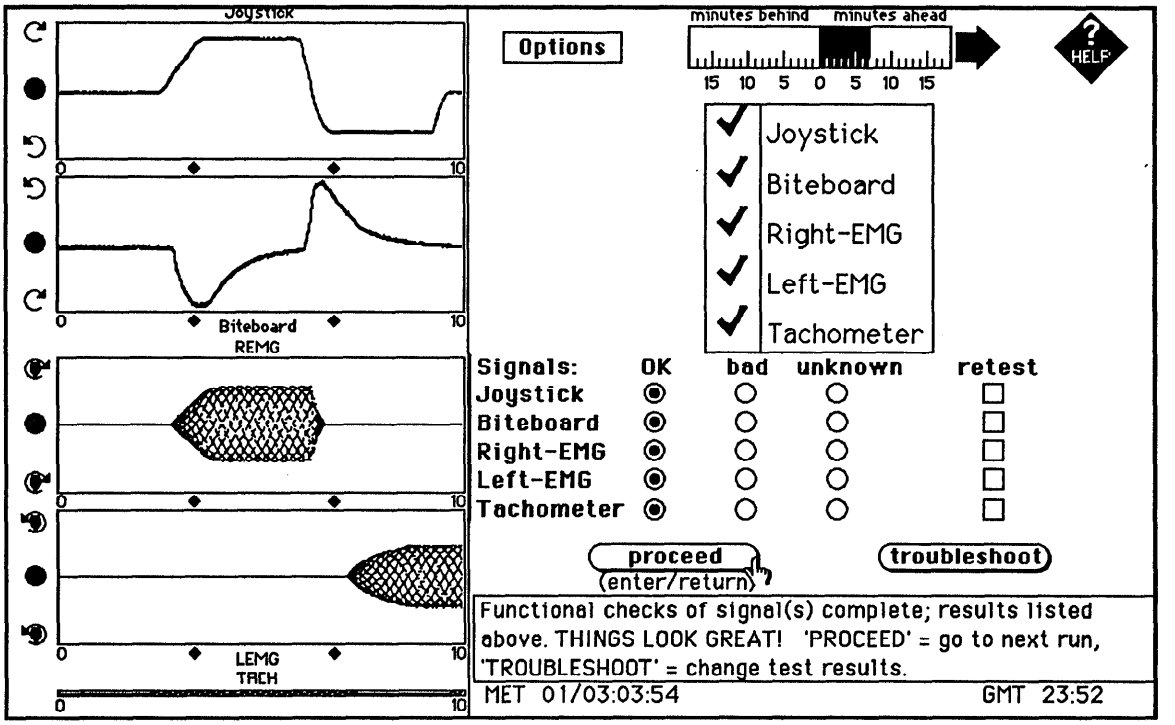

Figure 6: Functional check results

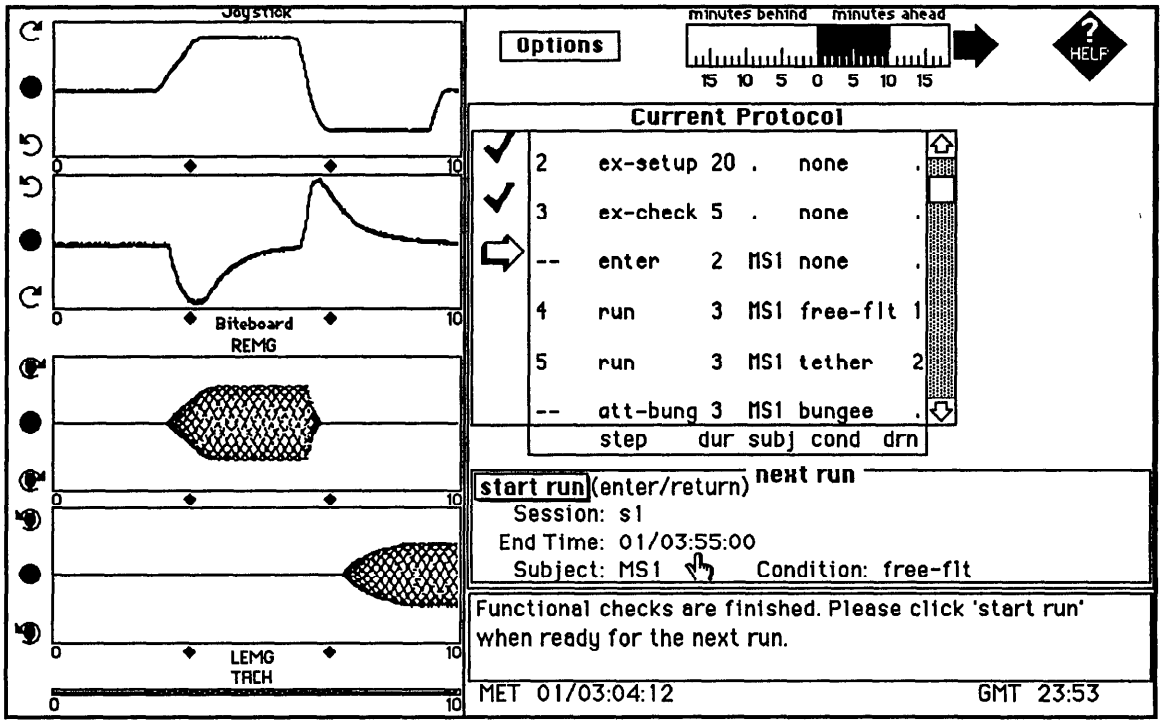

Figure 7: Main session display

the user. If troubleshooting is pursued, a series of interactions guides the repair effort.

Shown next is the main display screen (figure 7), with a summary of the current experiment protocol: (step) types, step (dur)ation in minutes, (subj)ect, experiment (cond)ition, etc. An arrow indicates the step currently being performed, check marks indicate completed steps and, finally, pending steps are listed below the current step. In this case, Mission Specialist 1 (MS1) is entering the dome to be tested in the free-float condition. This information is echoed in the "next run" display area just below the current protocol listing, and can be changed/edited by mousing over the item in need of update.

If the user wishes to check the procedures associated with the current, completed, or upcoming step, they merely click on that step in the current protocol window. A copy of the paper-based checklists and procedures is then displayed for review. These checklists exist as paper documents. They can be easily converted to "PICT" files which are then displayed by HyperCard. This facilitates maintenance of the procedures within the system as a result of Engineering Change Order (ECO) activity against the experiment apparatus.

Real-time data collection and analysis

PI-in-a-Box is a real-time KBS. It must receive, analyze, and then act on the data that is generated by the experiment quickly enough to be of use to the user. Each data-producing step in the vestibular physiology experiment consists of six 30-second data-producing trials (figure 8).

Each trial has 20 seconds of data gathering and real-time display followed by 10 seconds of rest (figure 9). The 20-second data collection/display period is controlled by a LabVIEW-compatible driver supplied by GW Instruments, the maker of the A/D converter. All five data channels are sampled at a rate of 225 Hz. During this time, the driver used for data acquisition monopolizes the PowerBook (even the "mouse" is not tracked). The following 10 seconds are shared by LabVIEW and HyperCard. The DAM performs data analysis, reduction, parameter extraction, and archival. It also communicates results to HyperCard for use in alerts and for post-run analysis (If DAM determines that a critical signal has malfunctioned during a trial then the run might be halted for troubleshooting).

The system performs several actions after a run of 6 trials. It first checks that at least five of six trials in a run have been successfully completed. If so, the run is labeled "nominal". It then checks the data for agreement with the current hypotheses (IDF). The IDF module presently consists of about three dozen CLIPS rules. Heuristics presently used to determine interestingness focus on the presence, onset latency, and intensity of "vection". These domain-specific heuristics include the following: the onset of vection is interesting if it is consistently less than two

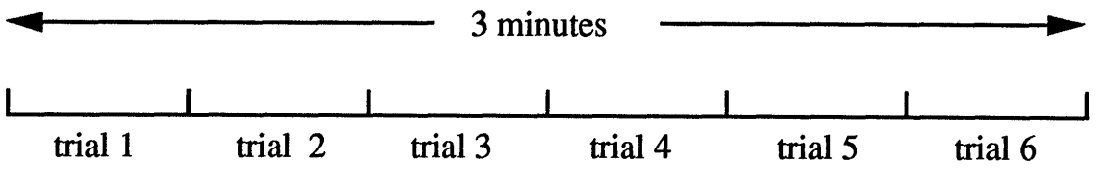

Figure 8: A Rotating Dome Experiment run

seconds; early in the flight, maximum vection is interesting if it is consistently greater than 90%; the number of dropouts experienced by a subject is interesting if it is consistently low (0) under tactile conditions; it is interesting if maximum vection under tactile conditions is consistently greater than maximum vection under non-tactile conditions; etc.

After the IDF module runs, the system prepares for the next run. Summary information is displayed for the user to review. A possible post-run display is seen in figure 10. In this case, the run was normal and the data was "interesting". An explanation of the interestingness is available for review if desired. In this example, the run was interesting for two reasons. The sensation of vection began quickly but with a low maximum compared to that predicted by the results of that same subject's earlier runs (see figure 11).

Between-run options

The user can invoke a variety of options (available from the "Options" pull-down menu) at this time. One option is to exploit the observed interestingness by asking the system for a better plan for session completion (experiment protocol). These protocols take into account the time remaining in the current session, as well as which subjects were tested, and with which results, from previous in-flight sessions. One of the two resulting suggestions is the "Proposed Protocol". This protocol observes session time constraints. The other suggestion is the "Optimal Protocol". This protocol relaxes the time constraint slightly to offer a focused plan with minimal negative impact on the mission time line. The time needed to suggest these new protocols is usually less than 30 seconds. The PS module controlling this consists of about 200 rules.

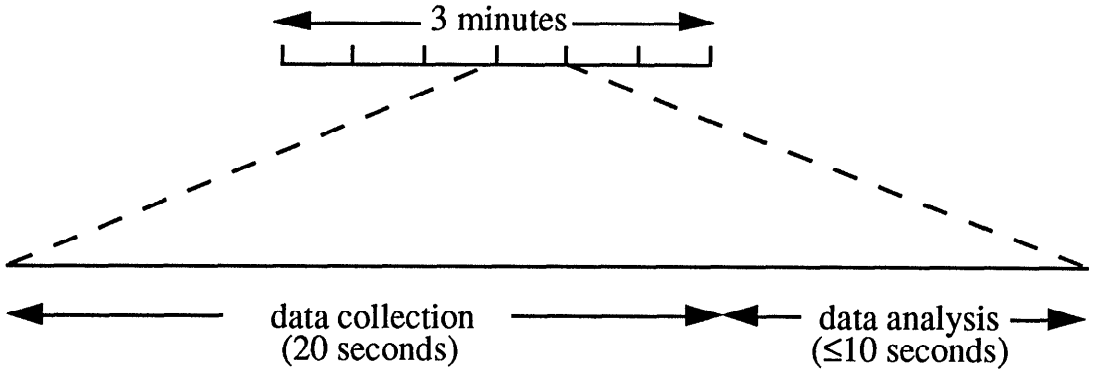

Figure 9: A Rotating Dome Experiment trial

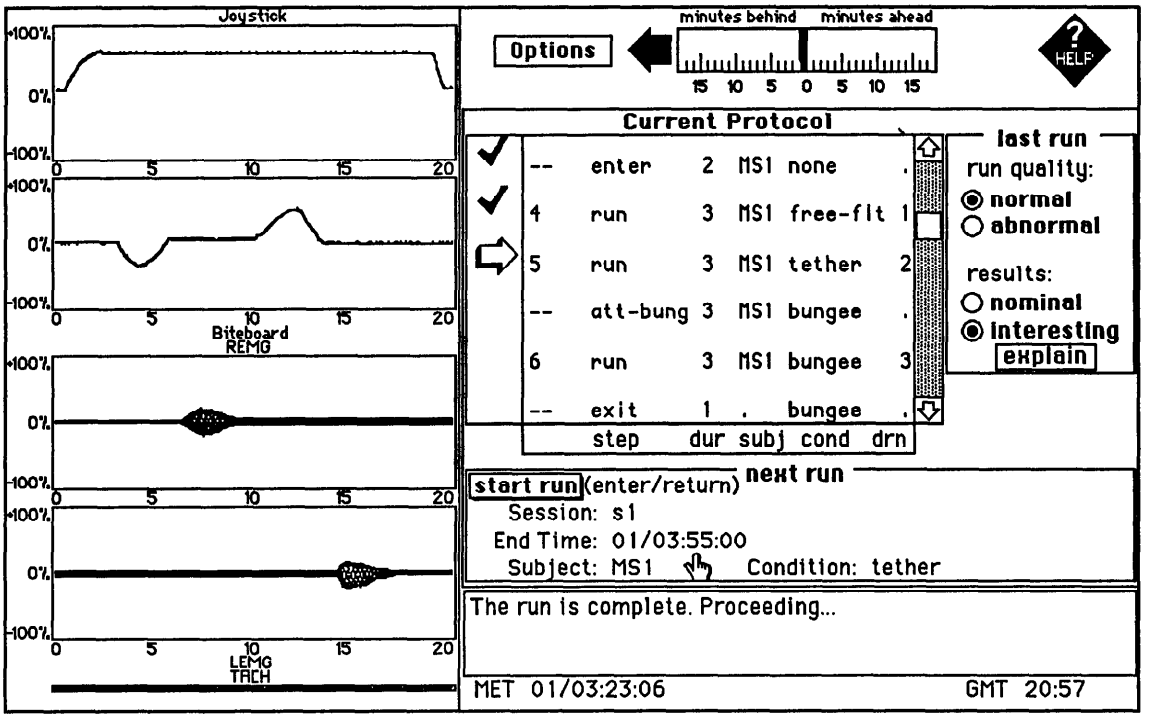

Figure 10: Post-run display

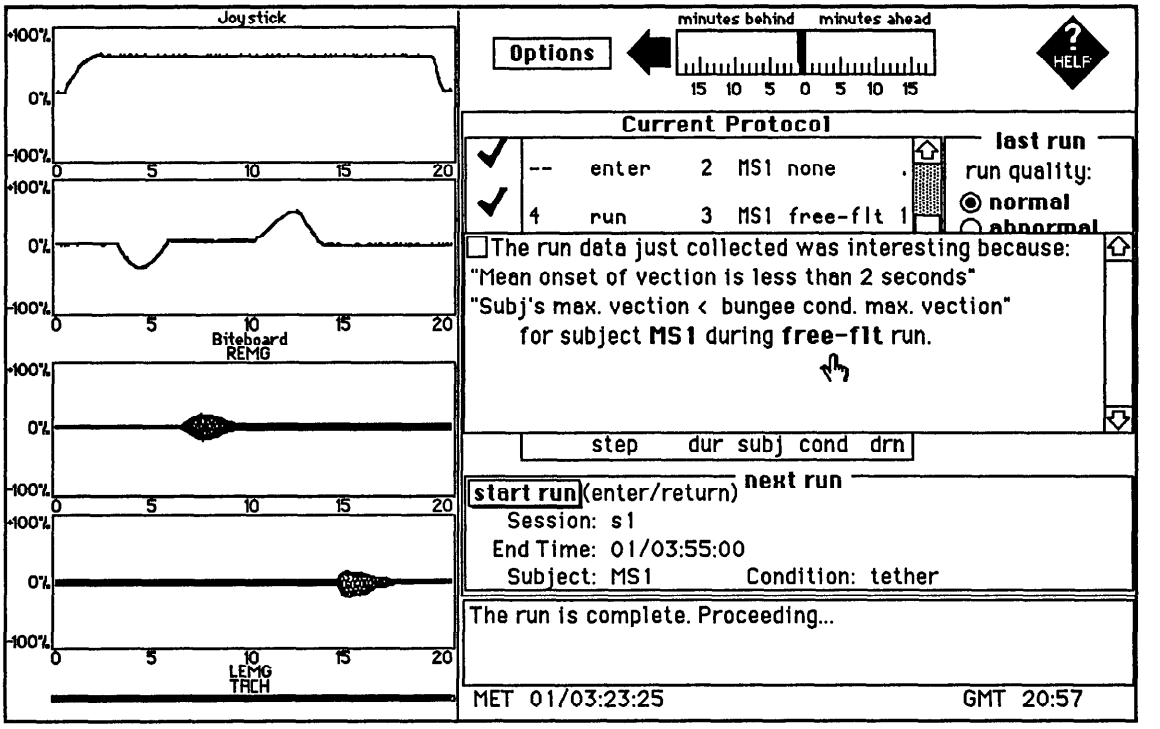

Figure 11: A brief explanation of interesting data

The experiment protocol is conceptualized as follows. There are a number of experiment sessions in a mission. Each session is conducted in accordance with its protocol. The protocol consists of a number of blocks. There is one block for each subject, a block for experiment setup, and a block for experiment stowage. Each block has a number of steps. The setup and store blocks are straightforward, but the optimum subject block order (and run ordering within a given subject block) depends upon a complex interpretation of previous mission history. The first task is to determine which experiment steps should be performed and to assign a (science) priority to each. The block ordering is determined next. After that, step ordering within a block is determined. Following the step ordering, minor setup tasks are inserted. Finally, the result is checked against current time constraints.

Heuristics presently used to determine new protocols include:
- Get at least some data on every subject;
- Complete the data collection using a full set of experiment conditions on at least one subject;
- After some data has been obtained, prefer missing experiment conditions;
- Do not schedule bungee runs if the Joystick signal is bad;
- Schedule runs with similar experiment conditions together;
- If the bungee is setup, schedule bungee runs first; if the bungee is not setup, schedule runs that do not need it first;
- After some data has been obtained, if there are "interesting" data to be confirmed on a subject, then prefer that subject;
- If a subject is currently setup in the test apparatus, schedule the remaining steps for that subject first;
- If no subject is setup, give first preference to a subject with unfinished tests from an earlier session and give next preference to a subject who was giving "interesting" data;
- If there was "interesting" data on a subject and experiment condition from the current session, and the subject and condition has not been repeated, then rerun the subject and condition;
- If there was "interesting" data on a subject and experiment condition from the current session, and that condition was again run today, but was not again found to be "interesting" then run that subject in that condition one more time;
- If there was "interesting" data on a subject and experiment condition from the current session, and that subject and condition was again run today, but the condition was not run on another subject, then run that condition on another subject;
- If there was "interesting" data to be confirmed on a subject and experiment condition from a previous session, and that condition was not run today, then run that subject in that condition;
- If there was "interesting" data to be confirmed on a subject and experiment condition from a previous session, and that condition was run today, but was not again found to be "interesting" then run that subject in that condition one more time;
- If more time is needed than is currently allowed, cut the least desirable (from science standpoint) step; etc.

As the previous paragraph suggests, it is difficult for an astronaut (or even an investigator) under time pressure to keep these heuristics together with their relative priority in short-term memory and to apply them correctly when rescheduling a session protocol.

Troubleshooting

Another option is to invoke a manual troubleshooting session (DTM). The user has a chance to indicate a variety of observed problems as seem in figure 12. In this case, the user has indicated that the "ECDS" display associated with an another onboard data-gathering computer is garbled. The system responds by recommending reinitializing the ECDS (figure 13), and displays the relevant portion of the control panel to aid recall (figure 14). A further example is seen in figures 15-16. Here, a bad EMG functional check lead to investigation of the EMG connectors, and later, to replacement of the EMG amplifier batteries. Additional functional checks are occasionally part of the troubleshooting process.

The DTM module consists of about 200 CLIPS rules. The CLIPS rules work with procedural code in HyperCard to guide the user through troubleshooting and repair. The overall flow involves the formulation of a repair recommendation based on the current experimental situation (The recommendation could entail forgoing a lengthy repair, for example). If troubleshooting continues, the system traverses a graph step-by-step until either the problem is repaired, the user terminates the session, or the system has no further advice to offer. Steps include the display of pictures (figure 14), line drawings (figure 15), repair instructions (figure 16), conducting functional checks, and making observations for the system (figure 15).

There are several other between-run options that can be invoked from the "options" pull-down menu. These include the use of a notepad, the review of the history of previously-completed sessions, and a manual editing facility for the current protocol. Finally, as mentioned earlier, the procedures associated with any step can be brought up for review.

System construction philosophy

The system makes maximum use of Commercial off-the-Shelf Software to leverage programming effort and avoid

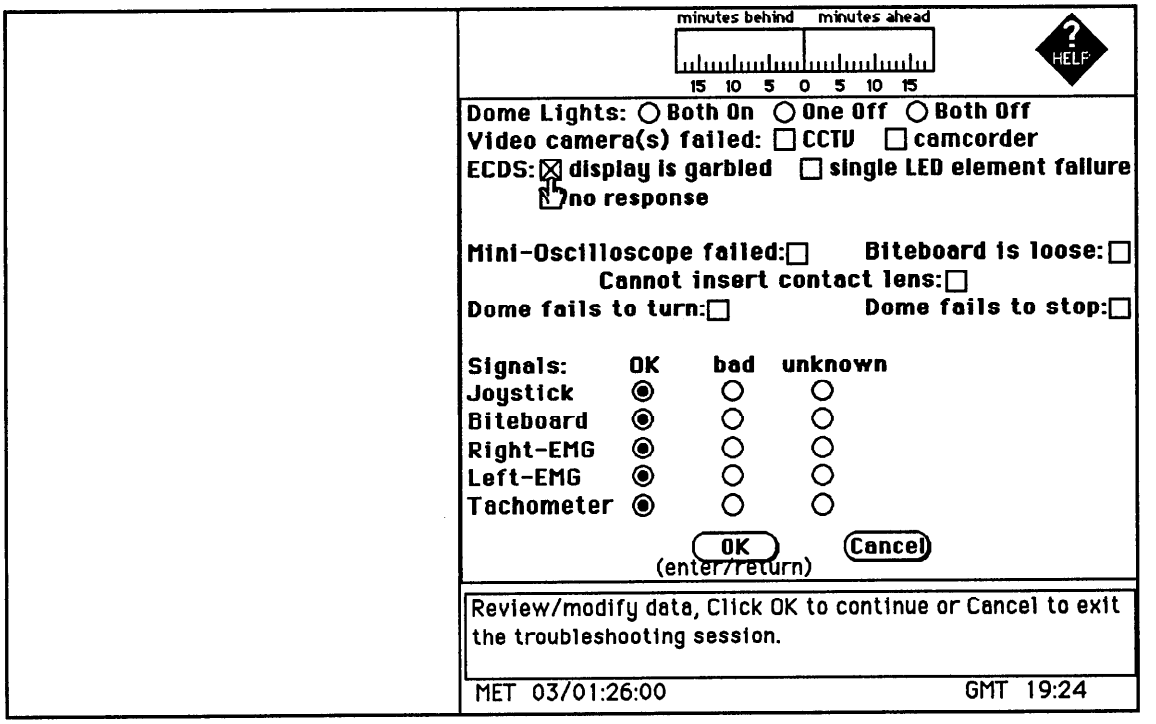

Figure 12: Selecting manual troubleshooting

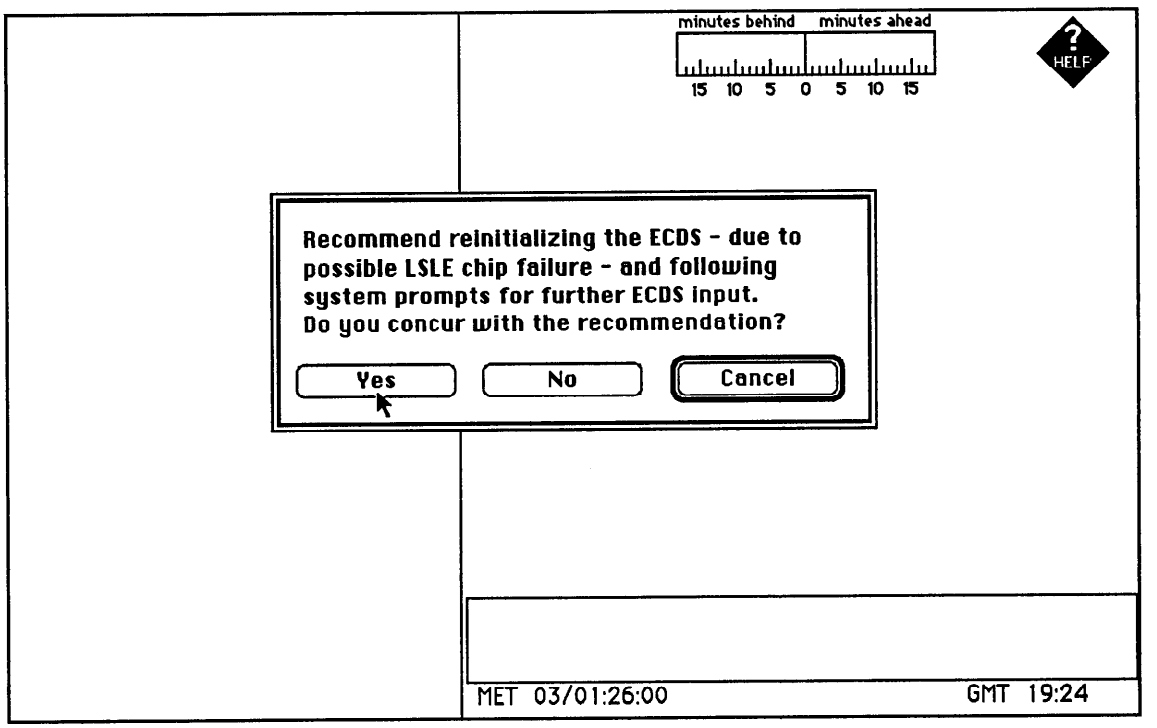

Figure 13: DTM recommendation

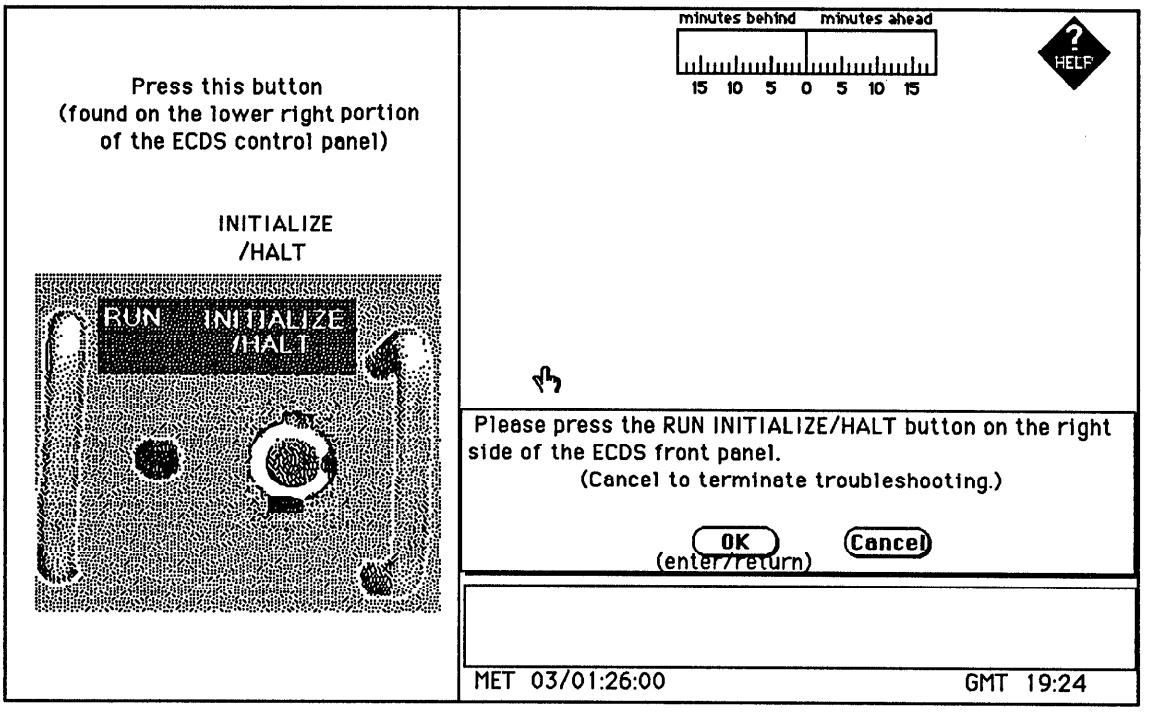

Figure 14: DTM instruction

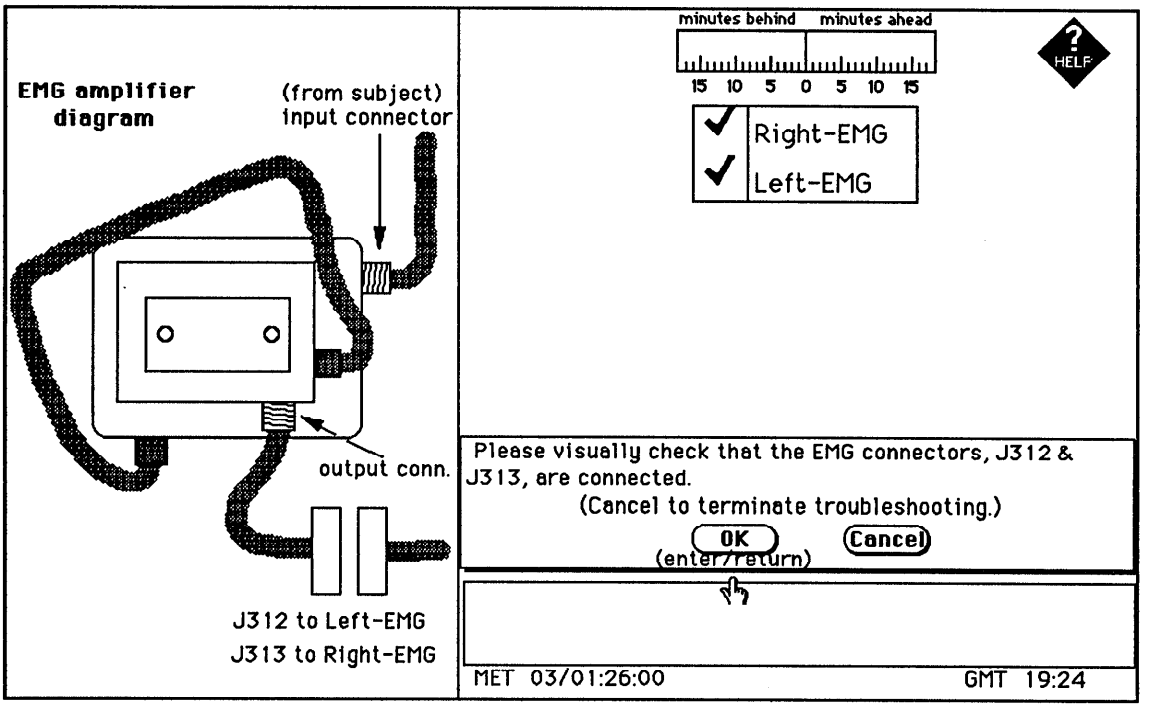

Figure 15: DTM request for information.

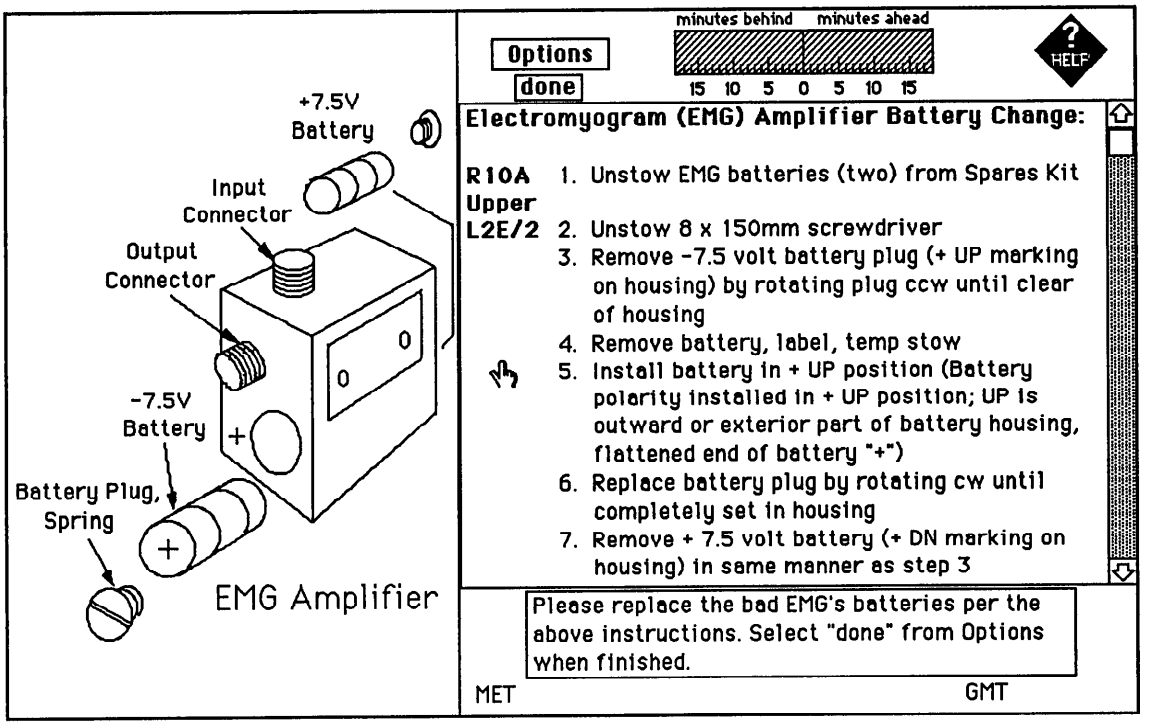

Figure 16: DTM repair instruction.

"reinventing the wheel". Some custom coding was done when necessary. There are clear advantages here: development started against a target machine that was not yet available, continued on a first-generation offering, and was delivered on a second-generation machine. Thus, users have the high performance of (and "excitement" of using) current hardware, and developers are able to take advantage of the increased hardware and software capabilities of the host machine. This is important because application software, and the programming paradigms it is based on, change more slowly than Operating Systems, which in turn change more slowly than CPUs. [Consider that CLIPS has been around since 1986 (or earlier), HyperCard and LabVIEW were first released in 1988, Macintosh System 7 was released in 1991, and PowerBooks became available in quantity in 1992.][5] Perhaps just as importantly, this approach has allowed better maintainability and a cheap upgrade path by leveraging off of the efforts of the technical staffs at Apple Computer, Claris, National Instruments, and the Johnson Space Center Software Technology Branch. For example, if we had to incorporate a complete interapplication communication facility between CLIPS, HyperCard, and LabVIEW, it would have been more expensive, less robust, and would have probably locked us out of routine software upgrades. Instead, we experimented with incremental enhancements that we could discard or retain as more capable software became available, or as the system's operational requirements changed. Another maintenance win occurs when team members occasionally, and inevitably, leave: there is a quick ramp-up for the new member who is already familiar with widely-used products like CLIPS, LabVIEW, and HyperCard.

A modified version of the spiral model of software development was followed for most of the major modules. This is a "requirements discovery" style where a rough specification is used to guide knowledge engineering and to rapidly construct a prototype. The prototype is then demonstrated internally (development team and end users) and externally (colleagues and upper management). Comments are then fed back into the specification. This leads to a follow-on version of the module after another iteration of knowledge engineering and rapid-prototyping. We believe this approach to be superior to the waterfall model of software development for KBSs for two chief reasons. First, there is no reasonable way to determine a specification that is detailed enough to guide a multi-year effort. It was only after coding, demonstrating, and using prototypes that differences between the way the task was

[5] The pace of improvement of hardware-based computing power over the last 20 years has been astonishing. Looking ahead to operations on a space station with a projected life of 40 years or more, it is critical not to start with hardware systems that will be obsolete before they are launched.

described and how the task was performed were resolved.[6] Second, the "knowledge" in the KBS is not static. As a result of the Space Life Sciences 1 Shuttle mission in June, 1991, there were significant changes not only in the experiment and its approach, but also in our conception of what the PI-in-a-Box system could best do. Our approach is not perfect. One potential problem is determining when to stop the development cycle. In our case, there were firm milestones associated with the flight schedule that provided the necessary constraints. Another potential problem is maintenance of requirements and test documents. As each development cycle modifies the system's requirements, these changes must be captured and reflected in the test plan and other documents. We found these issues to be minor frustrations compared to the benefits of our software development scheme.

An interesting approach to validating a module is "cross-prototyping". This was used on one of the modules. One team member built a prototype of the module based on the Official Production System (OPS) style of representation. When that team member left the project, maintenance and extension of the module was given to a new member. The new member was initially much more familiar with object/frame representations than with the OPS paradigm. After understanding the purpose and current requirements of the module, a new prototype was quickly built in IntelliCorp's KEE, and then translated to the Parmenides frame tool and FRuleKit OPS-style rule system (both from Carnegie Mellon University) running over Macintosh Allegro Common LISP. This provided performance comparisons between the two implementations and led to the discovery and elimination of several subtle bugs in the module.

Benefits of the knowledge-based system

The main benefit of this system is to maximize (or certainly increase) the scientific quality of data from experiments performed by humans in space. This in turn increases the value of the research performed. The increased value comes from increased crew productivity. This increased productivity has two dimensions. First, time is not spent on unproductive tasks after equipment failures. Second, reactions to the scientific consequences of already-gathered data are improved. Although caution should be exercised in generating dollar figures, we estimate savings of $6,000 per astronaut science hour (based on 20% crew productivity increase from operational use and a conservative figure of $30,000/hour of crew time on the space station).

[6] These differences included both the ground-based scientists and the astronauts. In both cases, actual task performance style was more conservative than the idealized version articulated for the system developers. We feel that the key to a really useful and valuable system lies in aiding actual task performance.

Future directions

Use of the system in support of Space Shuttle mission SLS-2 will continue through this year. The team is also working to identify follow-on experiments in future missions to support. After one or two experiments, we hope to know enough to create a general-purpose tool to aid science experiments in space, or indeed in any situation where quick-look analysis can be used to guide the focus of attention for the remainder of a limited scientific observation period. Examples under consideration include other life science experiments, materials science experiments, atmospheric studies, and plasma physics. While the system yields maximum benefit when applied to a particular experiment, there is value in adding just the DAM, DQM, and DTM modules to major equipment/facilities on Space Station (e.g., centrifuge and gas-grain simulation facility). This would allow a general-purpose monitoring, diagnosis, and repair system to be used over the life (20+ years) of the orbiting equipment. In this case a repair recommendation is weakened to the extent that experiment-specific data-collection heuristics and history are not available.

Alternative approaches

There are alternatives to the in-situ knowledge-based approach described in this paper. A more traditional approach would be to add more ground support people at one of the existing sites with voice and data channel links. They would do the analysis and present results and recommendations to the scientist. The traditional AI approach would be to add LISPM (in place of people) in the back room of Shuttle operations with a data-link to the scientist. Both of these approaches are critically hindered by the key problem of any ground-based solution: delays and outages in receiving experiment data and transmitting solutions to the crew. We feel that space experimentation requires careful consideration of the tradeoffs between those approaches and on-board intelligence.

Conceptually related work is associated with a Space Shuttle-based cryogenic experiment, SHOOT (Superfluid Helium On-Orbit Transfer). Two systems, AFDeX and CMS, facilitate the conduct of SHOOT (Raymond 1989), and (Shapiro and Robinson 1989). The AFDeX rule-based system is designed to provide intelligent process control, diagnosis, and error recovery. This system is hosted on a 80386-based GRiD laptop and will be sited in the Space Shuttle's Aft Flight Deck. The AFDeX system software is a combination of CLIPS and C code. AFDeX is capable on autonomous (closed-loop) control of the experiment, but is planned for use under astronaut control. The Command and Monitoring System (CMS) is designed to provide near real-time monitoring and control of the SHOOT experiment from the Earth by the investigator. The CMS is hosted on a Macintosh-II Computer. The system software is written

in Apple Computer's MPW(C). It is anticipated that most (roughly 80%) of the time the SHOOT experiment will be under ground control, while the remainder is under astronaut control.

Conclusion

PI-in-a-Box is a unique KBS for aiding scientific experimentation in space. We have used AI (symbolic reasoning), formerly-AI (advanced object-oriented HCI) and non-AI (data acquisition and analysis) techniques to build a useful system. Our framework will be expanded and generalized into a tool to aid the investigations that will occur on Space Station Freedom in the latter part of this decade.

Acknowledgements

We would like to thank the other members of the PI-in-a-Box team, past and present, for their efforts, especially Jeffery Shapiro, Guido Haymann-Haber, Chih Chao Lam, and Donald Kaiser. Thanks as well to the technical assistance provided from the NASA Johnson Space Center Software Technology Branch, National Instruments, and Apple Computer, especially Gary Reily, Chris Culbert, Rob Dye, John Hanks, John Fowler, Don Powers, Dave Nagel, and Tom Mayer. Thanks also to our reviewers for their helpful comments. Last but not least, thanks to NASA, especially Peter Friedland, Mark Gersh, Henry Lum, and Gregg Sweitek for management support.

This work was co-funded by the NASA Space Station Freedom Advanced Development Program and the Office of Aeronautics, Exploration, and Technology AI program to whom we are very grateful.

References

Frainier, R.; Groleau, N.; Bhatnagar, R.; Lam, C.; Compton, M.; Colombano, S.; Lai, S.; Szolovits, P.; Manahan, M.; Statler, I.; Young, L. 1990. A comparison of CLIPS- and LISP- based approaches to the development of a real-time expert system. In Proceedings of the First CLIPS Conference, 320-333. Houston, TX.: NASA Conference Publication 10049

Haymann-Haber, G.; Colombano, S.P.; Groleau, N.; Rosenthal, D.; Szolovits, P.; Young, L.R. 1989. An Expert System to Advise Astronauts During Experiments: The Protocol Manager Module, In Proceedings to the Conference on Space Automation and Robotics.

Raymond, E. 1989. Knowledge-Based Process Control and Diagnostics for Orbital Cryogen Transfer. In Proceedings to the Computers in Aerospace VII Conference. Monterey, CA.: American Institute of Aeronautics and Astronautics.

Shapiro, J. and Robinson, F. 1989. Interactive Remote Control for an STS-Based Superfluid Helium Transfer Demonstration. In Proceedings to the Computers in Aerospace VII Conference. Monterey, CA.: American Institute of Aeronautics and Astronautics.

Young, L.R.; Colombano, S.P.; Haymann-Haber, G.; Groleau, N.; Szolovits, P.; Rosenthal, D. 1989. An Expert System to Advise Astronauts During Experiments. In Proceedings to the 40th Congress of the International Astronautical Federation. Paris, FR.: International Astronautical Federation.