N91-22794

METAMORPHOSES OF ONAV CONSOLE OPERATIONS: FROM PROTOTYPE TO REAL TIME APPLICATION

Malise Mills, Lui Wang
NASA Johnson Space Center
Houston, TX 77058

ABSTRACT

The ONAV (Onboard Navigation) Expert System is being developed as a real time console assistant to the ONAV flight controller for use in the Mission Control Center at the Johnson Space Center. Currently the entry and rendezvous systems are in verification, and the ascent is being prototyped. To arrive at this stage, from a prototype to real world application, the ONAV project has had to deal with not only AI issues but operating environment issues. The AI issues included the maturity of AI languages and the debugging tools, what is verification, and availability, stability and size of expert pool. The environmental issues included real time data acquisition, hardware suitability, and how to achieve acceptance by users and management.

INTRODUCTION AND PROBLEM STATEMENT

What follows is a description of the development of the ONAV expert system. This project could aptly be titled the Agony and the Ecstasy. The ecstasy being the privilege of working with a group of highly talented and dedicated people, who persevered through some very trying times. Of working on the state of the art project that would enhance the quality and reduce costs of mission support. The agony portion of the project consisted of attempting to develop an real-time operational system when the state of the art of the supporting technology was still in its infancy and when; for the most part, there existed institutional pressures not conducive to change and innovation. This paper will trace out the development of the ONAV system and hopefully highlight some of the pitfalls to avoid in development of such a system.

The idea of the ONAV Expert system was conceived in the summer of 1986 by the Mission Support Directorate (MSD); now Information Systems Directorate (ISD), at the Johnson Space Center. MSD's goal for this project was to develop and implement a real-time expert system in the Mission Control Center. The ONAV position was selected for this purpose because it provided a well defined and relatively small problem domain. Also several of the software engineers at MSD had previous ONAV experience. However, to develop this project MSD needed to elicit the support of the ONAV flight controllers, who work in the Mission Operations Directorate (MOD) . At the beginning there was some opposition from MOD since we had flown 25 flights previously without such a system, therefore we could and can fly future missions without such support. But as with many projects, it was through the perseverance of several people that the project stayed on course.

ONAV FUNCTION

The Onboard Navigator (ONAV) is a back room position that supports the Guidance and Procedures Officer (GPO) in the Mission Control Center. The ONAV's principal responsibilities are to monitor the health of the onboard state vector and navigational aids that are used to update the state vector and make recommendations to the GPO for actions to maintain the state vector. The state vector represents the orbiter's position and velocity at a given time and is used in various guidance and control functions on the orbiter. The state vector is propagated using the inertial measuring units (IMU's) and is updated by a variety of navigation sensors through a kalman filtering scheme. The sensors used are dependent on the phase of flight. ONAV supports three phases of flight; ascent, entry and rendezvous. During ascent only the IMU's are used; during entry the IMU's along with the tactical air command and navigation system (TACAN), air data transducer assembly (ADTA), a drag altitude model and the microwave scanning beam landing system (MSBLS) are used. Rendezvous uses the IMU's, star trackers and the rendezvous radar. Basically, the ONAV job entails monitoring several digital displays that provide information on the status of the various navigation sensors and the onboard state vector. Approximately 180 parameters are normally monitored. The ONAV, based on this information, makes recommendations ranging from incorporating the sensor data into the onboard filter to recommending a ground derived state vector be uplinked to the onboard system. To be able to perform this function an ONAV must have an understanding of how the onboard software operates, how the sensors function, crew procedures and knowledge of kalman filtering. In addition the ONAV must be able to rapidly determine what information on the displays is important, which varies as a function of time, what failures have occurred, and what sensors are available.

OBJECTIVES AND GOALS OF SYSTEM

The objective of the system is two fold; one is to use the system as a console assistant and the other is as a training tool. Currently each phase of the mission requires two ONAV's for support; a lead and an ONAV2. The lead's responsibilities are to coordinate all information from the console, communicate with the GPO and make all pertinent calls. ONAV2 logs information and provides backup to the lead. As the system matures and both the GPO's and ONAV's become more confident with it, the goal is to eliminate the ONAV2 position. Also the system should enhance the quality of mission support. This is being accomplished in a variety of ways. The first method is the automatic logging capability of events and recommendations with both an altitude and a time tag that the system generates. Previously the ONAV's would log these events by hand and if several critical problems occurred at once it became difficult to keep up. The system will eliminate that problem. Another enhancement is the incorporation of color graphics in the user interface. This provides an easier to read screen and significantly aids in problem recognition and resolution by providing trending analysis.

Probably the most critical enhancement provided by the system lies in its ability to monitor all of the sensors at all times. At certain periods, one sensor may be more critical than others. As a result, the ONAV's attention tends to focus on the status and the functions of that one sensor and loses track of the others. The expert system, however, has the ability to continuously monitor all of the available sensors while still providing increased focus on the specific sensor that is providing critical data. Over 600 parameters are monitored by the entry system and 425 parameters for the rendezvous system. We expect the ascent system will be required to monitor approximately 400 parameters. This function has already proven to be highly valuable during flight simulations conducted by MOD, when the system alerted controllers to a secondary problem.

The second objective for the system is to use it as a training tool. Currently, training an ONAV2 takes over a year. An additional nine months is then required for ONAV1 certification. A major impact to the length of time that the training process takes is the number of simulation hours available. A new trainee can expect, at most, four hours a week in simulation time. The expert system will lessen the trainee's dependance on simulations by providing the capability of running training cases in an office environment. The biggest advantage of the use of the expert system as a training tool, however, lies in the establishment of the rulebase and its knowledge capture feature. When running as a training tool, the expert system will be equipped with a knowledge browser and rulebase rationale. This will allow the trainee to assimilate the information at his own pace and as a result he will not be as dependent on having an experienced person available to answer questions. Finally, by establishing a single set of agreed to rules the certified rulebase will provide a source of consistent training.

IMPLEMENTATION

The development of the rules for the ONAV expert system was a joint effort between the domain experts of MOD and knowledge engineers from MSD. The ONAV domain knowledge was captured through a series of meetings where the ONAV experts presented information to the knowledge engineers in a classic classroom setting. Initial meetings focused on the system functional overview. The results of the meetings were converted to different design strategies and prototypes, which could be critiqued by the experts. Prototype development then, served as the feedback mechanism for the meetings. During the development phase, when the knowledge engineer understood the basic operations, they would generally develop the agenda for the meetings on a particular topic and the experts provided the detail knowledge. The meetings were scheduled once a week two to three hours for the first three months, and after were held less frequently. By the end of six months we had the first prototype with a minimum user interface. But the overall system design structure had been selected and tested.

It turned out that the ONAV normal task functions were fairly straight forward; however, detecting anomalies in the subsystems and component

failures recommending proper response required detailed expert knowledge. The overall system design was completed in the first year of the development phase. This structure resulted from the basic nature of the ONAV task and from the modularity guidelines of any good system engineering approach. Four functional components of the expert system can be identified: (1) Fact assertion, (2) monitoring, (3) analysis, and (4) output. In addition, there is a fifth non-expert system component that is a part of the overall ONAV system called "data preparation".

The data preparation (data prep) component receives information from the operational environment and performs three functions:

- Collects the information required by the expert system
- Performs any computations required on the data
- Filters and transforms that data into a form suitable for the expert system.

In short, the data prep component converts the numeric instantaneous data points to a set of symbols which reflect the environment. The fact assertion component simply takes the prepared data and puts that data required by the expert system into the fact base. The monitoring component generates intermediate conclusions and statuses of the individual subsystems that ONAV observed and manages. This component simply detects environment state changes and pass them on to the next component. The analysis component performs an overall assessment of the current status, (i.e., makes use of the data from the monitoring phase) confirms the subsystems status, and/or recommend actions to prevent further degradation of the state vector. The analysis phase is the heart of the knowledge base, because this component captured the ONAV expertise. Specific knowledge implementation such as letting TACAN take out the state error rather than doing an uplink to the orbiter, or deselecting the line replaceable unit (LRU) when the onboard data bus has a commfault, are programmatically captured, and kept. The output component controls the sending of the notices and/or recommendations to the ONAV expert system console.

As for the look and feel of the system, our philosophy was to keep it as simple and clean as possible. To achieve this we employed the pop and shoot method. This method consisted of a series of pop-up menus or toggle switches activated by the mouse. This provides a format that is easy to read and allows the operators to keep their eyes on the screen, a required feature in the highly dynamic environment of orbiter high speed operations. Messages appear in windows that can contain up to 20 messages and are scrollable. Status and quality lights take full advantage of color and also contain text to insure that the operator understands the meaning.

As a real time console assistant, the expert system requires real time telemetry and trajectory data from the Mission Operations Computer (MOC), using a data stream called Generalized Data Retrieval (GDR). The GDR data is actually retrieved by a program called GDR_driver, which reads the data from the MOC and writes it into a shared memory segment. This shared memory

interface is based on a model developed at JSC for use by real time applications (ref. Workstation Application Interface to Data Source Interface Agreement). This model is generic and modular and it provides a standard mechanism to access different data sources.

CERTIFICATION

The certification/validation scheme we devised for the system was established at three different levels. The first level consists of the component level; data acquisition (GDR), data prep, rulebase, plots and the interface. Its intent is to insure that the program is functionally correct, ie. that data is correctly gathered, that the rules fire if presented with appropriate actions and events, and that results are correctly displayed to the screen. The second level consists of integrated system testing and overall certification. The intent of this level of testing is to insure that the various constituent parts of the system function in an integrated manner and, more importantly, that the recommendations made by the system accurately model the expert's mental concepts of the state of the system and are the correct responses to environmental changes. The third level consists of an integrated workstation certification. This last level is accomplished because of the critical nature of the workstations and is to insure that this application will not interfere with any other workstation or mainframe applications.

For the GDR certification, we collected data during a sim at several time points. We took hardcopies of the data from the mission operations computer (MOC). For each time point we then compared the data from the MOC to the data we received at the workstation. Once we were certain that we were getting good data from GDR, we then went through the same process with data prep. The next step in component testing has been the verification and validation of the rulebase, which has been the most difficult of the tasks.

To attack the problem of verification, we first had to start by deciding what verified meant. The existing literature, at the time, on verification and validation we found to be lacking for a rulebase of this size, approximately 300 rules for entry and 500 rules for rendezvous. The final decision, after much debate, was the construction of an error matrix that contains 48 errors that a certified ONAV would have to be able to deal with to be certified. For the rulebase to be declared verified, the system must act correctly at least twice for each of the errors.

The method that we have been using to verify the rulebase has largely been dictated by the source of test data. The optimum method would have been to run a nominal case and then introduce a specific error in each of the following runs. Due to the complexities of the problem environment, the only source of data for the system is simulations or missions. However, the goal of simulations is to train flight controllers, therefore we are at the mercy of the training area as to what errors occur in a sim. Consequently the verification is somewhat of a hit or miss operation. We log and evaluate testcases when they are available, not necessarily in a controlled, selected fashion.

When a sim run is selected, it is first delogged. In this manner actual copies of the controllers screens are obtained from the run. Next, an expert will go through the delog and determine what occurred in the case and what calls and events the ONAV flight controller would have made and noted. The next step is for the expert to run the testcase through the expert system. At this time, any errors or discrepancies are noted. The expert then writes a log with the errors and discrepancies and what the correct action should have been. This is then passed to the software engineers, who in conjunction with the experts, work on correcting the problems. After the software engineers are done, then the corrected rulebase is passed back to the experts. The testcase is then replayed, and if all errors have been corrected and no new ones appear then the new rulebase is used in replaying all previous testcases. If everything checks out then the that testcase is signed off by all parties.

In the event any questions arise that concern procedures or requirements, the action is referred to the ONAV working group. The ONAV working group was formed, with representatives from all the functional ONAV and software engineering areas, to focus the collective knowledge of the ONAV community and provide an authoritative method of resolving requirement conflicts or procedural debates. In this fashion controllers were provided input into the generation of the rulebase and a feeling that they know and understand what corrections have been made and the methodology that is incorporated in the rulebase. In addition the working group minutes serve to document the decisions made and provide the knowledge engineers a paper trail that they can refer to when creating/modifying the rulebase.

Once all the various components have been verified, the system will be ready for full certification. At this point the system will be treated just as any flight controller for certification. To obtain final certification, the system will be run through a full set of entry sims with an GPO and ONAV1 monitoring. The GPO and ONAV1 will then evaluate and determine final certification. Also during this phase the system will be checked to verify that it will not interfere with any other workstation application.

Since not only expert systems but also workstations represent a new way of console operations, the system will be implemented in three phases. The first phase, which is going on during rulebase verification, is for a third controller to monitor the system behind the current operator. This allows us to run the system during actual sims and flights without conflicting with console operations. Once the rulebase is verified, the system will then be moved on console and be monitored by the ONAV2. This phase will allow us to do the final interface work and answer questions such as, is the system easy to read and does it convey all the information necessary for console operations. Once all operators become comfortable with the system then at that point it will be ready to replace the ONAV2.

PROBLEMS

From our experience on the project, time is the most critical element on the project. Time provides the luxury of careful scrutiny of the system but if time is not carefully allocated it can rob the project of its momentum. The following problems represent areas that forced the project off its path and consequently slowed the development of the project.

As the name expert system implies, an expert is an integral part of developing an expert system. When this project first began, there were two ascent/entry experts and no certified rendezvous experts. For the projected flight rate at the time, six ascent/entry controllers were needed and two rendezvous controllers. Due to the low numbers, the priorities of the available experts were flights, training, analysis and finally expert system development. So not only were there a limited amount of experts, their availability was extremely limited also. At one point the only expert working on the entry system quit, leaving roughly a year and a half where no expert was available. These factors stretched the development out over a considerable length of time. This led to problems in the software engineering side, in that the programmers became restless with the lack of progress on the system. It consequently contributed in the decision by several programmers to leave the project. The expert system, in fact, had fallen prey to the very thing the system was trying to protect against, that of a lack of a knowledge base. Due to this problem, the luxury of time for a quick development of a prototype, at the start, was denied the project, consequently robbing the project of its needed momentum.

Due to the drawn out development time and turnover problems several different groups of experts worked on the entry system over time. This meant that the rulebase represented an inconsistent set of methodology and at times without documentation a lack of understanding of why things were done the where they were. It was in response to this, that the ONAV working group was formed. Its function is to try to reach a consensus among current operators and document that methodology. The working group concept brought a sense of stability into the project because it as a organization remained constant and left a documented trail.

Another obstacle faced by the project was the uncertainty over the platform that would be used for the final product. One of the goals of the project is to be used as a console assistant. The system is to reside on a Mission Control Center Upgrade (MCCU) workstation. Unfortunately, our project and the MCCU project have been developed in parallel. During the evolution of MCCU the final platform went from MASSCOMP to SUN back to MASSCOMP, from a two MIP's to an eight MIP's machine and from a non-color to a color graphics terminals. Through it all we tried to keep the system as portable as possible and tried to get our requirements into the MCCU project. As a result of our philosophy of portability, the project did end up with some throw away code. The interface was done in four different versions; Curses, MC-Windows, SunView, and X-Windows versions. Even at this time, the MCCU's lack of robustness has caused the expert system setbacks in gaining controller

confident since 90% of our sim failures can be attributed to workstation problems. It is difficult to avoid this type of problem when working in a high tech environment, on what might be called the bleeding edge of technology. The best that we can do is be alert to changes and try to influence any changes to our environment. We believe the fact that the systems were worked in parallel helped since our system was ready to move on to the configured workstation. Consequently we were able to evaluate workstation performance. The time lost working workstation problems, though, stole valuable time from user performance time.

As for any system data is the life blood of the system. There existed two data problems for the ONAV expert system, availability and access. As previously stated the only acceptable data source, due to the size of the problem environment, is from simulations and flights. This proved to be a constraint on our development from the start. The main problem, however, was in getting access to the data required. The system requires telemetry data from the orbiter, for onboard systems and state vector evaluations, and data from the MOC for output of high speed ground filter data. Several source existed from which telemetry data could be obtained. GDR, however, was the only source for the ground filter. A longer delay in achieving access to GDR data than anticipated occurred due to MOC integrity concerns. This caused an even larger delay between prototype development and rulebase verification. Possibly if we had tried to get a better understanding of the system's data requirements closer to the start of development; we may have been able to piece together a set of nominal testcases from some analysis tools. This would have given us a leg up on verification but not solved the whole problem. Also from our experience with verification the system's data requirements can expand through this process. This one factor stole the most momentum from the project and was one of the most frustrating experiences to endure. The system was all dressed up with no place to go.

The C-language Integrated Production system (CLIPS) expert system shell was used in the development of the expert system. The ONAV project received one of first beta copy of CLIPS (release 3.0) for the prototype of the ONAV project. Even in the beta software, CLIPS inference engine was very robust. However, as the expert system development continued, more rules were added to the system. The rule based interactions became more complex. CLIPS at this time did not provide adequate debugging methods to deal with large knowledge bases, but in fact the state of the art for expert systems in general was lacking in this area. This was a particular problem because the expert system has to be embedded with other modules, such as the real time data acquisition, and user interface modules which added another layer of complexity. For a long time, tracking down a typographical error was very difficult. Since then, CLIPS development team has developed additional tools such as the cross-reference, style, and verification (CRSV) tool, CLIPS window interfaces environments and also added additional syntax to the pattern language to deal with some of the problems. Finally, much research is still focusing on finding the appropriate methodologies to perform verification and validation on expert systems. Certainly, CLIPS will also evolve along with the

technology. Those are some of the costs that one has to realize and eventually justify for its value of the project.

One area of particular concern was the acceptance of the system by users and management. One criticism from management was that the system would erode controllers skills, that a reliance on the system would eventually cause destruction of the ONAV knowledge base. However, our response was if the certification process for a controller is adequate then there should not be any erosion in skills. Users were also reluctant to use the system because the system did represent change and would require new training. Our attempt was to bring in the users at the beginning, keep them involved and place the interface design in their control. Also all rulebase changes had to be blessed by the expert. In these ways the user community became totally involved in development and responsibility for the project thus increasing acceptance by the users. The major factor in acceptance, however, is simply time. Time for the flight controllers to use the system and watch as the system's performance proves itself worthy of confidence. Time for management to accept that the system will not be death of flight controllers as we know them. Time for the system to become fully integrated into console operations.

CONCLUSIONS

To arrive at this point in time, where users are beginning to use the system, where it is actually being of benefit to operators, has been, as chronicled in the preceding passages, somewhat a tortuous path. In hindsight, a critical element in the development of the project was the length of time between rulebase development and data flow. The length of time for our project proved to be much longer then the ideal, causing us severe problems but, ultimately, not disastrous ones. What we should have done was have the assurance of the availability of experts and a first cut of data before starting rulebase development. Without this, a rapid development of a realistic prototype proved impossible, which prevented the users from actually using the system. Time on the system and with the system, by the users, as we have experienced is crucial for the development of the system. The working group concept should also have been used from the start of the project to facilitate documentation and provide stability for the system.

Time, it turned out was an enemy of this project. Too much was spent waiting; waiting for experts, waiting for workstations and waiting for data. As previously concluded, prototyping quickly, and keeping everyone involved is critical. Luckily, however, time will come to our rescue. We have spent a large amount of time on certification. This time has only increased the confidence of the controllers. And the more time spent in console operations, the greater the acceptance. Hopefully the opportunity will present itself that some of these lessons can be applied to other systems and in time, we can learn from the past and not just ignore what we have learned.

References:

[1] Giarratano, J.,*The CLIPS User's Guide.* NASA Document, June 1988

[2] Riley, G., *CLIPS Architecture Manual.* NASA Document, JSC-23047, June, 1988

[3] *Guidelines and System Requirement for the Onboard Navigation Console Expert /Trainer System.* NASA Document, JSC-22433, June 1986

[4] Liebowitz, Jay, *Expert Systems with Applications Volume 1* November 1990

[5] AAAI-90 Workshop on *Knowledge Based System Verification, Validation and Testing* Boston, Massachusetts July, 1990

[6] *Knowledge Requirements for the Onboard Navigation (ONAV) Console Expert/Trainer System: Entry Phase.* NASA Document, JSC-22657, September, 1988