N88-17237

VERIFYING SHUTTLE ONBOARD SOFTWARE USING EXPERT SYSTEMS

William B. Wingert
IBM
Federal Systems Division
3700 Bay Area Boulevard
Houston, Texas 77058-1199

ABSTRACT

The Space shuttle uses a complex set of software to guide, navigate, and control it through all phases of flight. Adding to the complexity is the fact that the software is "reconfigured" for each flight. That is, thousands of constants in the software are changed to reflect the unique properties of a given mission (e.g., launch weight, orbit inclination). In the last level of test, The software is "flown" through end-to-end nominal and abort scenarios taking the shuttle from liftoff to landing. The analysis of the results of the Level 8 testing is experience and labor intensive. A set of pass/fail criteria have been defined for each testcase and in parallel with the knowledge acquisition, tools were developed which allowed the automation of the knowledge being gathered on paper. A prototype of the Analysis Criteria Expert System (ACES) has been put into production in the verification of the reconfigured onboard flight software. The system currently uses 3 PL/I programs, the ESE/VM program product and two large host systems to accomplish the task. The total system has approximately 3000 rules. The knowledge acquisition has begun again to take the knowledge base beyond simple pass/fail to the ability to determine the source of the criteria failures.

I. BACKGROUND

A. Space Shuttle Software

Four IBM AP-101B flight computers host a set of highly critical and complex programs to guide, navigate and control the Space Shuttle through all phases of flight (see Figure 1). The flight software, developed by IBM under contract to the National Aeronautics and Space Administration (NASA), also drives a set of instruments and graphic displays, accepts keyboard and other inputs from the astronauts, and interfaces with various hardware sensors and effectors. The flight computer operating system ensures that all active computers operate simultaneously, each performing identical functions. A failed computer is detected automatically and removed from the set of redundant computers. Due to the complexity of the avionics and data processing systems, size of the software and reliability requirements, independent verification has been employed in all phases of the software life-cycle to increase product quality. The goal is flight software which is "error-free."

B. Flight Software Testing

A software test plan provides for a structured process to identify and facilitate correction of software implementation or requirements errors, leading to demonstration that the software satisfies all design requirements. Testing is divided into two main phases: development tests, which are concurrent with the software development and performed by the software development organization, and verification tests, which are carried out by the independent test organization. Figure 2 describes the elements of the test program.

C. Level 8 Testing

The first seven levels of testing are performed on the basic set of programs whose software logic can support many flights. Thousands of constants in the software are changed to reflect the unique properties of a given mission. Level 8 testing consists of testing the software under simulated flight conditions and stresses, with the flight software configured for a particular shuttle flight. The tests are conducted through flight simulations exercising the onboard software and computers in a simulated flight environment provided by the Software Production Facility. The volume of simulation data required to adequately analyze the performance of the flight software is impressive. Each test generates over two million plotted data points and 30,000 lines of printed output. There are 15 - 25 tests run for each shuttle mission.

The analysis consists of evaluation of simulation variables at various events in the shuttle trajectory, analysis of flight sequences, and analysis of plots and cockpit displays. Execution and analysis of the suite of tests is both labor and skill intensive and requires up to six weeks for completion.

The goal of the Analysis Criteria Expert System (ACES) is to automate the analysis of the logged data. The benefits are numerous and include reducing labor costs, improving the quality and consistency in interpreting the data, and reducing the total time currently required to manually analyze the simulations.

II. KNOWLEDGE ACQUISITION AND ENGINEERING

The knowledge acquisition and engineering for ACES began in early 1985 when "pass/fail" criteria were created for a subset of our verification testcases. That effort was later extended to include more criteria and more testcases. In parallel with the knowledge acquisition, we began looking into different ways to automate the evaluation of the criteria. By the Spring of 1986, we had decided on a method for criteria automation and completed the first phase of knowledge acquisition. The knowledge base consisted of about 250 rules for each of the 14 testcases.

After using the criteria for about a year, we felt it was a good idea to take these criteria a step further and document how criteria violations were analyzed. In this process, we decided to reorganize the criteria. This reorganiztion meant duplicating some of the previous efforts. However, we felt that the expected size of the comprehensive criteria demanded the reorganization. The final set of the written criteria were in a form that could be picked up and independently implemented into an expert system. This process has been completed for about a third of our verification testcases.

III. IMPLEMENTATION

Implementing the pass/fail criteria proved to be much more difficult and complex than we anticipated. We encountered problems with automated data transfer from the simulation to the expert system, volume of data, rule base creation, and results presentation.

The problem of data volume turned out to be the biggest challenge. Each simulation produces over three million data points and 30,000 lines of printed output. The expert system shell we were using was the Expert System Environment (ESE) and was the only internally available mainframe expert system software. Due to ESE's lack of speed when handling large amounts of data, some sort of pre-processor would be necessary. We wrote three PL/I programs to handle the type of data we analyze.

AutoProg is used to process plotted data. However, it cannot handle textual data such as the simulation chronology (the online) and the textual representations of the Shuttle's onboard displays (the DEU images). To handle the Online message criteria we produced the Online Event Extractor (OLEVE). This program looks at the online output for missing, out of order, and unexpected messages in the chronology of the simulation. Each simulation logs images of the Shuttle's displays and later stores them in a file for processing by the Display Electronics Unit Criteria Evaluator (DEUCE). This PL/I program searches for a particular DEU based on some criteria such as time or within a certain amount of time of an event. This program can look for text strings in a group of DEUs, perform math operations, and print DEU images and captions for use in testcase reports. See figure 1 illustrating the flow of these programs

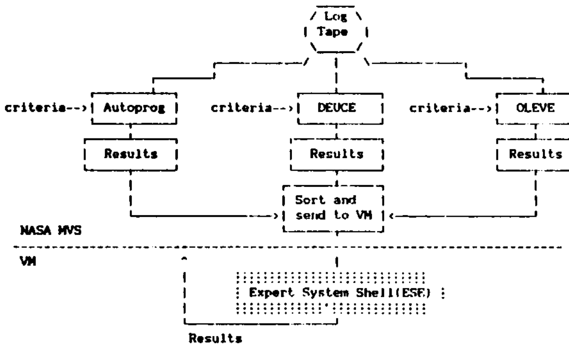

Figure 1 - ACES Flow

The entire process is automated from start to finish and requires no human intervention other than to submit the simulation job. ESE/VM can handle the smaller amounts of data produced by the PL/I programs within a reasonable timeframe. However, since most of our criteria reside in the PL/I program the ESE knowledge base is relatively small with about 20 rules for each testcase compared to more than 250 PL/I criteria. We are currently working on prototypes using other expert systems in the hopes of enlarging the knowledge base and producing an integrated expert system.

IV. VERIFICATION OF ACES

The verification of the expert system was divided into two distinct parts: tool verification and knowledge base verification. Tool verification concentrated on verifying AutoProg, DEUCE, and OLEVE. Knowledge Base verification centered on testing the rules supplied to the above tools as well as the knowledge base contained in the expert system.

A standard test approach was used for the tool verification. All of the tools went through a requirements, design, and code review process. Prior to code review, all of the tools were put through a set of unit tests designed to exercise all of the capabilities of the tool. Known valid inputs were fed to each of the tools and the developer analyzed the output for expected results. This review process, well known on the onboard Shuttle project, was easily implemented with very few tool problems.

The knowledge base verification is a different story altogether. Since the criteria were designed to work on any shuttle mission (of which there are essentially limitless combinations), we decided each knowledge base should be applied to at least three different shuttle flights. This was an arbitrary number, but we felt three different flights would give us good coverage over a range of shuttle trajectories. Wherever possible we had someone other than the developer of the knowledge base perform the testing. The analysts first performed manual analysis of the testcase using the written criteria as a guide. All problems and violations of the criteria were noted. The analyst then ran the expert system against the testcase and compared the results to that of the manual analysis. Differences were noted and probable source of the difference was noted, for example, knowledge base deficiency, knowledge base error, or tool error. All tool errors resulted in a Discrepancy Report or Change Request against the tool to bring it into compliance. Knowledge base errors were returned to the knowledge base developer for resolution.

As new versions of the tools become available, they are unit tested and executed with a small subset of the actual rules to insure identical results are produced. As knowledge is added it is tested as a standalone entity before being merged with the existing data. In production, if a tool or knowledge base problem is discovered, a Discrepancy Report is written to document the problem.

V. CURRENT USE AND FUTURE PLANS

The Flight Software Certification organization uses ACES today in their verification of the Shuttle onboard flight software. After a simulation is made, a job is submitted which automatically executes ACES and puts the output in a dataset. Some manual analysis is still performed on those criteria which are not yet covered by ACES.

Our main focus for the future centers on the expert system tools that we are evaluating. The expert system shell we are evaluating is ES PL/I, a PL/I-based imbeddable expert system shell. PL/I has proven to be faster, but all development is done using a standard text editor. We have built a prototype of our knowledge bases using ES PL/I. We have found it to be faster than ESE/VM. The main drawback has been documenting the rules. Rules are documented with ES PL/I by putting PL/I comments in line to the knowledge base. However, if the knowledge is well organized and catalogued, this poses no major technical problem. Development using ES PL/I requires a good understanding of PL/1 and the number of lines generated is similar to ES PL/I.

In general, we have found it useful to build a prototype of our knowledge base in order to evaluate its relative merits. We try to scope the size of the prototype such that it can be completed in a two to four week period. On the other hand, we try to include a good mix of rule types such that we can get a good feel for the amount of time it takes to implement the various types of rules. With the prototype, we can analyze the relative merits of an expert system, provide demonstrations for our customer, and tailor our paper knowledge to meet the requirements of a particular expert system shell, if necessary.

Expert systems can be used successfully to verify critical software. The time and resources required can be reduced and the quality of the verification can be maintained or improved by applying expert systems technology to an existing software verification effort. However, the transition takes time to perform the proper knowledge engineering and acquisition. In addition, with proper planning, it is possible to insert expert systems technology into an existing production environment.