A Prototype Expert System in OPS5 for Data Error Detection [1]

James Rash
NASA Goddard Space Flight Center
Telecommunication Systems Branch, Code 531
Greenbelt, Maryland

ABSTRACT

A prototype expert system has been developed in the OPS5 language to perform error checking on data which spacecraft builders/users supply to the NASA Goddard Space Flight Center for processing on the Communications Link Analysis and Simulation System (CLASS) computer. This prototype expert system, called Trajectory Preprocessing System (TRAPS), contains 49 rules and at present runs on an IBM PC in the OPS5+ software package from Artelligence, Inc. In its operational phase, TRAPS will run in the Oak Ridge Production Language (ORPL) on the CLASS computer (a Perkin-Elmer 3244 supermini). ORPL, an implementation of OPS5 by the Oak Ridge National Laboratory in MULTIFORTH on a Hewlett-Packard 9836 desktop computer, is now being ported to SS-FORTH on the CLASS computer. This paper discusses the expert system problem domain, development approach, tools, results and future plans stemming from the TRAPS project.

CLASS comprises several major software systems (written in FORTRAN) designed to analyze space and ground communications system performance. CLASS was developed by NASA primarily for the prediction of user spacecraft communications system performance through the Tracking and Data Relay Satellite System (TDRSS) (the term "user spacecraft" or "user vehicle" is used to refer to a spacecraft which is utilizing services provided by TDRSS).

The basic functions facilitated by CLASS are: (1) communications system design, (2) communications system performance analysis, evaluation, and prediction, (3) spacecraft mission planning, and (4) post-launch trouble shooting of communications problems.

The CLASS computer, located at NASA's Goddard Space Flight Center (GSFC) in Greenbelt, Maryland, is a Perkin-Elmer 3244 supermini with 16 megabytes of RAM, serving up to 32 simultaneous users locally or in distant cities.

A number of areas exist in CLASS for the application of expert systems: communications link fault diagnosis, communications system design, communications scenario

----

[1] Extracted from a paper presented at the 1986 Conference on AI Applications, NASA Goddard Space Flight Center, May 15, 1986.

optimization, and checking input data for certain kinds of errors. A prototype expert system for checking user-supplied input data for errors will be the focus of this paper.

It should be noted that the term "expert system" is used somewhat loosely in this paper to mean any rule-based system whether "expert" or not.

The basic constraints on the selection of expert system software for CLASS required that delivered applications be imbedded (or embeddable) within the CLASS environment, interface efficiently with existing CLASS programs, and run in a multiple-user mode. An additional consideration was that of a limited budget.

Various alternative approaches for expert system development, such as the acquisition of premium hardware and software systems (e.g., Symbolics and ART), or the contracted porting of some existing product (e.g., Rulemaster or KES II) to CLASS, were rejected as either technically unsuitable for the CLASS environment or too costly in time or money. It was finally decided to use the OPS5 language. Initially, OPS5+ from Artelligence, Inc. was to be used as a start-up tool on the IBM PC. In parallel, OPS5 was to be implemented on the CLASS mainframe by porting the Oak Ridge Production Language (ORPL), originally implemented in MultiForth on a Hewlett-Packard 9836, to SS-FORTH on CLASS.

Following consideration of several possible applications of expert systems for CLASS, it was decided that an expert system to preprocess user trajectory data would be constructed. This was a simple, straightforward problem which could be solved using either OPS5 or FORTRAN, thus permitting a comparison of the OPS5 language with FORTRAN. The main factor to be compared was program maintainability, where OPS5 was projected to provide a significant advantage. Program maintainability appeared to be particularly important in this problem domain because of the many possible variations in mission and spacecraft data requirements, which would make necessary the development and maintenance of many code modules.

CLASS analysis of a typical planetary mission may involve processing several hundred possible trajectories as specified by the CLASS user. Each trajectory will have 150 to 250 time points starting at the time of release of the user vehicle from the Space Shuttle and ending at the time communications with TDRSS are lost following the completion of the user vehicle's engine burn phase. Perhaps 2 to 3 hours total ground elapsed time is covered by the analysis for each such trajectory.

Analyses for orbital missions are essentially the same but may involve much longer mission time periods as well as special factors such as earth occultation and interference from earth multipath reflections.

CLASS users furnish input data in a standard format on magnetic tape. The input data must satisfy a number of specific rules. Some of these rules are mission-unique, and the rest are generic. However, in all cases the rules ("human rules") are derived from the logic of space communications rather than from the internal requirements of CLASS programs.

Preprocessing the user-supplied input data to determine whether it conforms to these "human rules" is the function of the Trajectory Preprocessing System (TRAPS), an expert system designed to protect CLASS from bad input by (1) recognizing bad data before the start of processing by the analysis and simulation programs, and (2) reporting the data errors to the CLASS analysts.

Each of the six initial requirements ("human rules") on which the TRAPS expert system prototype was based was translated into from one to eight OPS5 rules for processing the data records, plus from one to five additional OPS5 rules for generating the TRAPS output messages. With the rules for initialization and for reading the input data, the total number of OPS5 rules in the TRAPS prototype is 49.

At the present time, the expert system produces messages indicating any errors found. TRAPS does not (yet) have the capability of correcting or modifying the input data in any way.

The first version of TRAPS (based on only five of the human rules) read in all records from the input file before the data-checking rules in the knowledge base were allowed to begin firing. This version was very inefficient because as the number of data records increased, the size of OPS5 working memory rapidly increased and the processing speed rapidly decreased.

The second version of TRAPS (also based on only five of the human rules) used a record-by-record processing approach in which each record was processed against the data-checking rules before the next record was read. This record-by-record processing minimized the number of OPS5 working memory elements regardless of the number of records to be processed. This second version of TRAPS processed records approximately 14 times faster than the first, but had some inadequacies.

The third version of TRAPS was derived from the second version by adding a module of OPS5 rules corresponding to one additional human rule and by revising some of the rules to improve readability and to handle special cases. These enhancements caused the third version to run somewhat slower than the second version.

The following table summarizes the processing efficiency of the three versions on the IBM PC, based on processing a data file containing 163 data records using OPS5+ Version 2.0003. The values for processing speed are not highly significant since they are data dependent and sensitive to the structuring of the condition elements within the rules.

An important feature of OPS5 code is that literals (corresponding to variable names in other languages) can have virtually any length. This feature of OPS5 was utilized to satisfy the requirement that the code for TRAPS

TRAPS Processing Efficiency

Version	No. of Rules	No. of Rule Firings	Processing Speed	
			Records/second	Rules/second
1	41	397	0.11	0.26
2	40	849	1.6	8.1
3	49	862	1.3	6.8

be easily maintained by being easily read.

Future enhanced versions of TRAPS can be developed readily by simply including one or more additional code modules derived from new human rules. The ability of OPS5 to accommodate this modular approach further ensures maintainability.

To facilitate the goal of comparing OPS5 and FORTRAN, a FORTRAN program was written to perform the functions of the original five human rules. A comparison of this FORTRAN subroutine with the equivalent OPS5 code written for TRAPS shows that OPS5 can be made more readable than FORTRAN, as has been noted by others. This comparison also suggests that OPS5 is more terse than FORTRAN. However, this is often not the case since a number of functions, especially input/output functions and general numerical computations, are cumbersome in OPS5. The FORTRAN code developed for the TRAPS application was about 30% shorter than the OPS5 code because lengthy literals were deliberately used in the OPS5 program to enhance readability.

Based on the experience of developing three different versions of this prototype expert system as well as a more or less equivalent FORTRAN program, it is the author's opinion that OPS5 offers a considerably higher degree of maintainability than does FORTRAN.

However, it should be stated that developing OPS5 programs is not without pitfalls. Subtle interactions can occur between the rules in a production system program during interpretation by the inference engine. These interactions can be difficult to predict and tricky to debug. Thus, programming in OPS5 requires carefulness and skill, perhaps beyond that required for procedural languages such as FORTRAN.

From the TRAPS development effort, the development tool (OPS5+) received a positive evaluation overall. It was found to be an effective and essentially bug-free product with a favorable price-performance ratio.

The process of porting the ORPL (OPS5) package (developed by Oak Ridge National Laboratory), to SS-FORTH (developed by Sun Studs, Inc. of Roseburg, Oregon) on the CLASS mainframe is under way. After ORPL has been ported and validated, future efforts are expected to divide into two areas: (1) other CLASS expert system applications, and (2) enhancements to ORPL.