N88-16443

APPLICATIONS OF EXPERT SYSTEMS FOR SATELLITE AUTONOMY

A. Ciarlo
ESA-ESTEC (WDI)
P.O. Box 299
2200AG Noordwijk
The Netherlands

P. Donzelli
LABEN SpA
SS Padana Superiore 290
20090 Vimodrone (Milano)
Italy

ABSTRACT

This paper discusses some aspects of the on-board application of Expert Systems (ES's) in artificial satellites. The ideas presented are mainly based on the experience gained during a study performed by LABEN, CRI, DORNIER and CRISA under European Space Agency (ESA) contract; the activities of the study, which include the implementation of two prototypes on a dedicated AI machine, are described. The more general implications of the experience are then discussed. These concern firstly, the interrelationship between the ES and the architecture of the satellite and its impact on the mission definition phase of the satellite lifecycle. Secondly, the main obstacles that need to be overcome before operational use of ES's on-board can take place, and namely the matters of testing, knowledge collection, and availability of computing resources. Finally, the activities that appear to be required in the near future to prepare the way for the full exploitation of this technology for satellite autonomy are briefly outlined, together with a brief description of an ongoing work studying the application of AI techniques for the management of the Cassini Titan probe; the probe will not have a telecomand link and will therefore have to manage autonomously its descent on Titan.

INTRODUCTION

Lately increased satellite autonomy has become a more pressing necessity; the reasons are various, but all have their roots in the fact that current technology allows very complicated missions to be implemented, the control of which, with the methods developed for satellites of previous generations, is extrememly costly. The recent availability of powerful development environments for Expert Systems (ES) has simplified the implementation of these to the point of making them viable tools for the achievement of increased spacecraft autonomy, and some activities have been initiated by the ESA to investigate the possibilities of applications of ES's to its own purposes.

ESA SATELLITE ARCHITECTURE

Over the last twenty odd years an architecture and a relatively fixed break-down into major functional subsystems have become generally accepted. The subsystems usually include Structure, Thermal, Power, Data Handling, Attitude and Orbit control, Telemetry/Telecommand, Antennas, and Payload(s). This break-down in functional subsystems represents the best compromise among often contrasting requirements, of which a significant one is the need to comply with geographical distribution i.e. assignment of tasks to companies in all participating member-states

In this architecture the Data Handling (OBDH) represents the core of the whole spacecraft from the point of view of control and monitoring; it performs

all the functions required to decode and distribute telecommands, gather and format spacecraft data for telemetry, and to provide a general purpose on-board data processing facility. The design of the OBDH is based on a distributed architecture of (intelligent) units connected by a serial bus; specialised units provide interfaces to satellite subsystems and payloads, while a Central Unit controls bus data traffic and general subsystem operation. The fact that this subsystem has access to all satellite data and that it includes a number of computing resources makes it the "natural" host of an on-board ES for autonomy.

STUDY OF EXPERT SYSTEM FOR SPACECRAFT MANAGEMENT

One of the studies initiated by ESA had as main purposes to verify the feasibility of an on-board ES for the management of an autonomous spacecraft, to identify the general requirements of such an ES, to assess the resulting on board complexity, and finally to identify areas for future research in the field. The study was carried out by a consortium of four companies: LABEN (Italy), CRI (Denmark), Dornier (W. Germany) and CRISA (Spain).

The first task was to identify a satellite which could be used as reference for the study. It was as decided to define a "hypothetical" satellite, rather than rely on a real design, because it was thought that this would allow it to be tailored to fit the purposes of the study; this in fact resembles the "toy problem" approach. The definition of the reference satellite served its purpose, because it provided a stable, logically well defined test case of reasonable complexity, but the whole procedure introduced some difficulties that will be explained later. The architecture of the reference satellite was based on the functional breakdown usual for ESA satellites, and the ES was supposed hosted in the OBDH.

Although the eventual goal of the study was to increase satellite autonomy by the exploitation of ES technology in an a priori unrestricted fashion, it was also clear it would be necessary to identify a limited application domain. The second task of the study was therefore a review of the possibilities within the general areas of mission, health and failure management; of these, the mission and health management were discarded because these tasks consist essentially of resource and activity scheduling, and therefore require reasoning about time. The remaining one, fault management, was chosen also because diagnosis is one of the historically successful applications of ES's. For what regards the object of the fault management activity, the Power and Data Handling subsystems were selected because the complexity of their design and operation matched well with the goals of the study, the relevant expertise is well established within the Consortium, and finally because their continuous operation is vital for the safety of the spacecraft.

The conventional method of knowledge acquisition through face to face consultations between the domain experts and the ES builder was difficult due to the geographical distribution of the personnel involved. Consequently, after an initial study of the problem area, a "knowledge specification formalism" was produced. This was then submitted to the domain experts, who wrote down the problem solving knowledge, producing "paper knowledge bases". These, together with the actual descriptions of the target systems (OBDH and Power) were the input to the coding of the ES. There was good correspondence between the knowledge specification formalism and the ES architecture.

Consequently, the actual coding of the ES's using the "paper knowledge bases" was fairly easy. However, it turned out that the problem solving strategy used by the ES deviated in many cases from that of the experts. These deficiencies had their roots in an insufficient problem analysis at the outset of the study, or at least in the failure to identify the implicit assumptions about diagnosis strategy within the specification formalism and to discuss them with the domain experts. The problem solving techniques should have been identified better before the knowledge specification formalism was constructed. Although the prototype systems seldom arrived at erroneous diagnoses, the failure in capturing correctly the experts' strategic problem-solving knowledge is serious. Obviously the solution adopted was too optimistic, and more code-evaluate-update cycles are required even before the formalism is fixed; afterwards it can be used by the experts to supply inputs to the knowledge base in a reasonably simple yet consistent form.

The architecture of the ES was based on a representation which models the fault propagation by means of a causal associational network. Such a network allows certain sets of decision strategies to be implemented retaining an explicit representation of the diagnostic knowledge. This knowledge is structured in three different layers: observation layer (symptoms and test), causal layer (failure states) and diagnostic layer (diagnosed states). The observation layer contains all the information that can be obtained by the subsystem organised in symptoms and tests: the first ones are used to activate the diagnostic process, the second ones are used by the inference engine, during the diagnostic process, to discriminate among failure hypotheses. The causal network and diagnostic layers are represented by a directed acyclic graph of nodes where each node identifies a state representing deviations from the normal behaviour. Three types of nodes are used:

- Failure states: nodes which conjecture the occurrence of a failure in a certain system component
- Diagnosed states: end nodes of the network containing the identification of the failed system component
- Starting nodes: pointed at by the associational arcs as result of the identification of a known symptom.

The nodes are connected by means of two types of arcs:

- Associational arcs that link the symptoms with starting nodes identifying the weight of the association;
- Causal arcs that link couple of states identifying the weight of the causality.

The development environment used in the study was the Intellicorp's KEE running on on a Texas Instruments Explorer workstation. The failure states (diagnosis hypotheses) were implemented as units (frames), and the determination of the causal pathways was effected through inferences over rule-sets. Tests and actions were associated with the units representing failure states as methods. The debugging facilities provided by the system enhanced the "visibility" of the diagnosis process considerably, especially compared to what would have been possible with a "conventional" software development environment.

Two prototype expert systems were constructed, one for the Power S/S, and one for the OBDH S/S. The user interfaces are similar in design for the two systems.The graphical facilities of the development environment are used to display information about knowledge bases, subsystem, and diagnosis process.

The user's communication with the prototypes is based on menu selection. He may specify the symptoms of a failure among those that are considered likely by the experts; when tne setting of the symptoms chosen for the test case is completed, the diagnosis process is started. No emulation of the subsystems has been implemented, so the user must supply the ES with the outcome of the actions it attempts to carry out, be these tests or activation of redundant units. When more than one failure can conceivably cause the same symptoms, the diagnosis process implemented in the ES will choose as first hypothesis the most critical one on the basis of a criticality value that is associated to each failure symptom.

IMPLICATIONS OF STUDY RESULTS

From the early stages of the study a difficulty arose in the definition of the corrective actions that were to be executed by the on-board ES: quite simply the choice was restricted to some redundancy switching. The difficulty was initially attributed to the lack of detail in the satellite that had been chosen as target; as more insight into the matter was acquired, it was recognized that the problem was more basic than it seemed and that it was also caused by the idea at the base of many a system design: to make each component as much a "black box" as possible. This was compounded by the fact that, to avoid the difficulties related to the representation of time and the reasoning about it, sequences of actions were considered only in atomic form. The bottom line is that simplification of the interfaces among units reduces integration and control problems, but severely limits the choice of action to correct a failure, to the point of making questionable the advantages of an ES compared to conventional algorithmic or table driven software. In the future the complexity of spacecrafts will increase and so will integration problems; in addition, the functional partition of the satellite in subsystems is closely mapped into areas of competence of european industries which are by now well established and not likely to be easily changed. Therefore, it is to be expected that the same type of problem will appear in the future.

Another conclusion that can be drawn from the previous paragraph is that it is difficult and not necessarily advantageous to use an ES in a satellite designed without this technology in mind. This point of view is however very much in contradiction with a ratner general attitude which is more or less expressed by "first mission requirements must be defined, and from these the need for an ES". In this respect it is interesting to note that even within this study the flow of activities effectively followed this pattern: first a reference satellite was defined on the basis of an architecture which was developed with completely different priorities, and then an ES was designed to take up the tasks that were meant to be carried out by simple on board HW or SW. One should of course accept that this represents an "initiation rite" for most new technologies; at the same time, one should be aware of its existence and of its limiting effects on the appreciation of the technology. Requirements are inevitably influenced by what is known to be feasible or available and therefore mission definition studies will take into account ES technology; however, it is also clear that to rely only on this mechanism will imply an unnecessary delay before a potentially extremely useful technology can be applied to its full capabilities.

OPEN ISSUES

A number of problems remain to be solved before ES's can be accepted for routine on board use. The first is the matter of validation, probably the biggest obstacle that needs to be overcome before on board ES applications become accepted; current on-board SW for unmanned satellites is orders of magnitudes less complex and "brute force" testing can achieve sufficient coverage. This is certainly not going to be possible with ES's, and in fact, although the problem has been identified and described, a satisfactory solution is not available; nevertheless, work is being carried out in this area, and one can feel reasonably confident that an acceptable solution will be found. However, much of the testing of ES's will continue to rely also on the assessment by human experts of the "reasoning" that generated a specific output. The reasoning is accessible only through SW facilities that are not justified in a satellite out of ground contact; therefore there is a definite risk of having to choose between two equally undesirable alternatives: installing on board a system which is overburdened by unnecessary facilities, or performing the tests without adequate support. A possible solution would be the development of a standard run time environment together with a tool for the automatic porting from the development system to the run time environment. In this way the testing and validation problem could be split in two parts, the first being the "conceptual" testing, to take place in the full ES development environment, and the second being done once and for all by the validation of a tool for the automatic translation of the ES to a "streamlined" run time version. The translation would consist essentially in the removal of the display, explanation, modification, etc. facilities and, when applicable, of a compilation.

The second problem is that space qualified HW which is currently available or in sight does not provide the resources that are needed to run in acceptable time ES's that have been developed with powerful shells. The previously described combination of run time environment and automatic translation tool could provide a meaningful contribution to the reduction of on-board resource requirements without performance penalties.

Finally, within the european space industry the problem of knowledge collection is compounded by the fact that experts are widely dispersed among different companies and countries; in this respect, procedures and tools to aid the process of knowledge collection and formalization would be particularly beneficial. The method used in this study can be seen only as a first step.

CONCLUSIONS

The two prototypes that have been implemented proved that an ES that performs a meaningful subset of the functions required for satellite autonomy is feasible in the european space industry with current technology. The line of activity has been extended to implement an ES to manage the descent of the Cassini probe on Titan. This mission has been chosen as target because no TC link will be available and because, given its relatively early stage of definition, it is hoped that the conceptual problems described earlier could be overcome. The same consortium will implement a prototype on an AI workstation and then transfer it to some HW more representative of on board resources; the purpose of the additional step is to acquire some dimensioning information on the related difficulty and to be able to run some performance tests.