TELEMATICS *and* INFORMATICS Vol. 8, No. 4, pp. 425–437, 1991
Copyright © 1991 Pergamon Press plc. Printed in the USA
0736-5853/91 $3.00 + .00

EXPERT OPERATOR'S ASSOCIATE: A KNOWLEDGE-BASED SYSTEM FOR SPACECRAFT CONTROL

Mogens Nielsen
Klaus Grue
François Lecouat

Abstract – This paper presents the Expert Operator's Associate (EOA) project, which studies the applicability of Expert Systems to day-to-day space operations. A prototype Expert System is developed, which operates on-line with an existing spacecraft control system at the European Space Operations Centre, and functions as an "operator's assistant" in controlling satellites. The prototype is demonstrated using an existing real-time simulation model of the MARECS-B2 telecommunication satellite. By developing a prototype system, it is examined to what extent the reliability and effectiveness of operations can be enhanced by AI based support. In addition, the study examines the questions of acquisition and representation of the "knowledge" for such systems, and the feasibility of "migration" of some (currently) ground-based functions into future space-borne autonomous systems.

1. INTRODUCTION

Supervising a spacecraft, interpreting the telemetry received, deciding about the correct on-board operational conditions, reasoning about proper corrections, and executing the appropriate control procedures are complex tasks for modern spacecraft. During the launch phase of the spacecraft, specialists may be at hand, who know about the design of the various subsystems on-board, but in the subsequent operational phases they will usually not be available for immediate consultancy to the operators responsible for safe day-to-day monitoring and controlling of the spacecraft.

The complexity of modern spacecraft systems, and the resulting high demands on the personnel operating them, concerns about the potential for human errors, and the risk of inaccessibility of the people with appropriate expert knowledge, calls for improved methodologies and environments providing computerised support for monitoring and controlling spacecraft.

An essential element in the development of such support is the transfer of parts of the knowledge regarding the procedures for controlling the spacecraft and regarding the design of the spacecraft, from the experts who conceived them, to a system that can be used to assist in working with the spacecraft in the operational phases.

Mogens Nielsen and Klaus Grue are both members of the Space Support Systems group with Computer Resources International A/S, Bregnerødvej 144, DK-3460 Birkerød, Denmark. François Lecouat works with MATRA ESPACE, z.ℓ. du Palays, 31, nu des Cosmonautes, 31077, Toulouse, France.

The study is performed under Contract No. 7627/88/NL/DG to ESA for ESOC and ESTEC. It is carried out by a consortium composed by CRI and MATRA.

This has been the motivation for the Operations Center of the European Space Agency (ESOC) to define a 3-year-study project, called the Expert Operator's Associate (EOA) project, with the aim of developing a prototype expert system for assisting in the operation of satellites. The MARECS-B2 communication satellite has been chosen as an example case for demonstration of the prototype. The prototype will be demonstrated with the aid of ESOC's "high-fidelity" real-time MARECS-B2 spacecraft simulator (a software model), which operates in closed loop communication with the ground control system via simulated telemetry and telecommand links.

The project is structured such that the first phase concentrates on constructing the basic system. It is here demonstrated how the operator can be assisted in the selection and execution of Flight Control Procedures (FCP), covering the situations where the spacecraft operates within the limits prescribed by the specifications of the satellite.

In the second phase of the project, the scope is extended such that some of the situations where the spacecraft does not operate inside these limits are taken into account. It is demonstrated how to assist in the situations where the problem can be diagnosed immediately, and handled by predefined Contingency Recovery Procedures (CRP).

Finally, several experimental extensions of the system are investigated. One extension is operator assistance in the situations that cannot be immediately diagnosed. Another extension is machine learning, where new knowledge (e.g., CRPs) is developed as a result of a dialogue with a spacecraft expert, and stored in the knowledge base of the system. Furthermore, the question of to what extent control functions can be "migrated" from the ground to future spacecraft, and the question of how to "streamline" the transfer of knowledge from the spacecraft experts to the system, will be addressed.

The prototype developed is a workstation-based system, controlling the process of daily operations of the spacecraft. It works in a real-time environment communicating with the spacecraft operator and the spacecraft.[1] The user interaction is facilitated by a graphical user interface utilizing state of the art techniques such as mouse, multiple windows, and pop-up menus.

At the time of writing, the project is near its completion and the paper presents the overall results, concentrating on the knowledge representation used, and the system architecture. In Section 2, the general problem domain and the example case are further described. Then, in Section 3, the functionality and architecture of the prototype system are introduced. In Section 4, the representation and structuring of the knowledge used by the system, and the expert functions, are described. In Section 5, it is illustrated how the execution of Flight Procedures is implemented. Alarm processing is described in Section 6. Finally, Section 7 concludes and the continuation of the project is detailed further.

2. THE PROBLEM DOMAIN AND THE EXAMPLE CASE

The operating state of orbiting spacecraft is monitored and controlled on the ground at ESA's ESOC at Darmstadt, W. Germany, by specialist personnel supported by on-line spacecraft control computer systems. These computers receive telemetry data from each spacecraft, typically in near-real-time via ground stations in various parts of the world. Data from the telemetry are evaluated and displayed to the spacecraft controller, who in turn can initiate the uplink of telecommands (via the ground station) to the remote satellite, from his computer console.

[1]via the Multiple Satellite Support System (MSSS).

MARECS-B2 is a geosynchronous maritime communications satellite, and is an interesting example case for an on-line Expert System to support spacecraft control. MARECS-B2 poses requirements in day-to-day operation, which are typical for the current generation of telecommunications satellites. The downlinked housekeeping telemetry data, flowing 24 hours per day, provides a "snapshot" of the spacecraft state in a "format" of several hundred "parameters" (readings of on-board sensors) every 19.2 seconds.

The spacecraft control computer system for MARECS-B2 is the ESOC Multiple Satellite Support System (MSSS).

The spacecraft controller monitors only a few of the telemetry parameters at any time, but the MSSS performs automatic checks on many of the parameters in each new format when it has been received. If the checks discover that parameters are outside their normal operating range, or status, audible and visual alarms are raised, so that the spacecraft controller is aware of a possible problem and can decide what to do.

In regular day-to-day operation, which is the type of activity which can be most effectively supported by EOA, the actions of the spacecraft controller are, in principle, completely defined by a large manual of operations procedures known as the MARECS-B2 Flight Operations Plan (FOP). This comprises FCP covering nominal operations, and CRP, which describe the actions to be taken in the event of non-nominal cases. The existence of the FOP ensures that operations can be carried out with a high degree of efficiency (speed in effecting configuration changes which affect the end-user services provided by the satellite), and reliability. Both of these aspects are of prime importance in the provision of telecommunications services.

Nominal operations of the spacecraft are preplanned, and a schedule is defined a few days beforehand by a specialist, who selects the FCPs required, and defines the time at which they are to be performed. However, it occasionally happens that during operation of the preplanned schedule, the spacecraft exhibits some unexpected behaviour. The spacecraft controller then has the task of selecting the appropriate CRPs. For this, he may need the assistance of a specialist engineer, but the latter may be unavailable immediately (e.g., in the middle of the night).

It also affects the complexity of the task, that the spacecraft controller must sometimes take account of actions and knowledge about the spacecraft state in the past, that is, historical information.

One of the functions of the EOA is to assist the spacecraft controller in choosing the right CRPs in a given non-nominal situation. In many cases, this will be possible on the basis of straightforward matching of the situation to corresponding descriptions stored with each CRP. Thus the EOA will effectively speed up the selection process. However, the situation will sometimes arise where the choice of CRPs is uncertain. One of the study aims is to show how the EOA can assist in the selection of recovery action, also in such cases.

3. OVERVIEW OF THE EOA

3.1. Functions

The core functionality of the EOA system is to assist the spacecraft operator in nominal spacecraft operation. Support is given by:

- Receiving, interpreting, and displaying information regarding the state of the satellite;

- Proposing selected procedures based on the current operating state of the spacecraft and the user's indication of the desired state;
- Presenting the chosen procedure to the user in both textual and graphical form;
- Preparing the various spacecraft command sequences needed for the execution of the plan, and on acceptance from the user, sending them to the MSSS;
- Receiving and evaluating reports from the MSSS on commanding activity;
- Continuously verifying the validity of constraints posed by the procedure.

Non-nominal situations are identified by the reception of alarms (e.g., TC verification failure alarms or Out Of Limit alarms) or by trend analysis of telemetry parameters. At present the EOA system assists in processing alarms by:

- Investigating the cause of the alarm;
- Distinguishing alarms requiring action from nonrelevant or expected alarms;
- Invoking and executing CRPs.

Situations requiring complex diagnosis before execution of a CRP or requiring actions that are not implemented in the form of an existing CRP, are considered in the last phase of the project, and are therefore not yet demonstrated by the prototype.

FCPs and CRP's are both special cases of Flight Procedures. EOA supports the execution of procedures in general, and therefore the execution of CRP's is supported in the same way as execution of FCPs.

The EOA utilizes the knowledge of experts to perform procedures, and to reason about problems in much the same way as is done by the experts themselves. It has the ability to explain its reasoning, and incrementally acquire new knowledge.

Additionally, the EOA is more flexible than conventional software, for example, it responds opportunistically to incoming data, or situations, and modifies its behavior under varying conditions. As an example, procedures have to cope with the exigencies of the current situation, or cope with reconfiguration or modification of the units considered.

The EOA provides a number of expert functions integrated within the system, which can be organized along three axes:

- Procedure generation;
- Spacecraft state monitoring;
- Execution scheduling.

These types of function are described in more detail in Section 5, and can be summarised as follows: Concerning *procedure generation*, the functions range from interpretation and execution of already existing procedures to generation of new procedures. With *spacecraft state monitoring*, functions range from conventional verification of patterns of parameters to complex failure diagnosis. *Execution scheduling* functions are initially dedicated to controlling the timing of procedure execution. However, it happens that procedures compete for execution, and the system provides functions to arbitrate conflicts. Additionally, EOA has facilities for supporting the spacecraft engineer in editing and maintaining the different types of knowledge in the knowledge bases. In particular, a syntax driven Flight Procedure editor has been developed, in addition to standard knowledge maintenance facilities.

3.2. System architecture

The EOA system communicates with two external entities: the user and the MSSS. The EOA system runs on an independent SUN workstation and communicates with the MSSS system via an X.25 communication link. It is implemented in the programming language Common LISP using the expert system shell KEE. However, to ensure proper speed in performance, and to have proper access to the operating system, parts of the system taking care of communication with the MSSS and the user is implemented directly in the C programming language.

The architecture has been designed with special attention to the fact that the EOA is integrated in a real-time environment, and that it must always be able to respond to the MSSS (e.g., to process alarms). Furthermore, an aim of the architectural design has been to construct an open-ended and modular architecture, thereby supporting maintenance and future extensions.

The result is a multiprocess architecture, consisting of a number of interacting systems, communicating through a common protocol. The architecture is outlined in Figure 1. The **EOA MANAGER** takes care of the overall scheduling in the system, and the internal communication protocol. The **Dialogue System** is a monitor for the **User Interface**. This interface is described in the next section.

Receiving, buffering, and parsing of information from the MSSS, and formatting and sending messages to the MSSS is handled by the **External Systems Interface**. The **TM/TC System** is a monitor for the **External Systems Interface**.

All the system **Knowledge Bases** have a common interface, called the KB Methods, through which all accesses are made. The **Knowledge Base Management System** constitutes a monitor for the Knowledge Bases of the EOA, and contains functions for retrieving, inferring, and updating knowledge. The **Flight Procedure Execution System** is the central system supervising and controlling the execution of procedures (FCPs and CRPs), interpreting TMs, validating and verifying TCs, etc. The **Knowledge Maintenance System** monitors the execution of the EOA, the user, and MSSS interactions, and controls the consistency, completeness, and feasibility of the EOA knowledge. The system proposes updates of the knowledge and also evaluates updates proposed by the user.

The **Fast Response Recovery System** takes over control in case of a non-nominal situation and performs alarm processing and selection of CRP;s to invoke in cases where this can be done without complex diagnosis.

In other non-nominal situations, control may be transferred to an **Advanced Reasoning System** which, in close interaction with the user, performs complex diagnosis and generates new procedures if necessary. As indicated in Figure 1, this system is not implemented in the current prototype so far.

3.3. User interface

There are two main categories of user: *spacecraft operators*, who control the daily operations of the spacecraft; and *spacecraft engineers*, the experts who know about the design of the spacecraft. Each has a different pattern of communication with the system.

The User interface utilizes "state of the art" Man Machine Interface (MMI) techniques, including mouse, windowing, and pop-up menus in the user interaction. It has been designed using the powerful facilities of KEE.

Figure 2 shows the basic layout of the EOA screen used for daily operations. It is divided into two separate areas, the **System Area** and the **Procedures Area**.

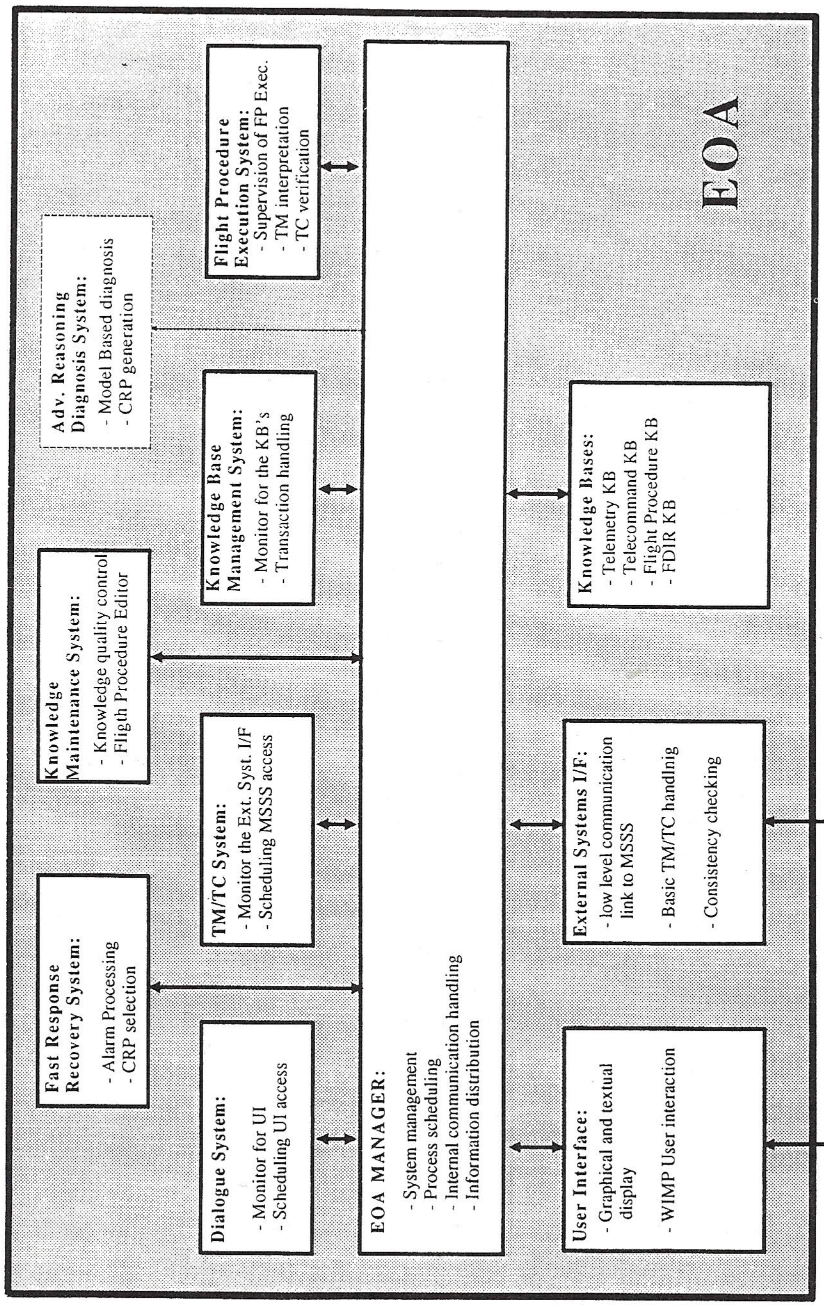
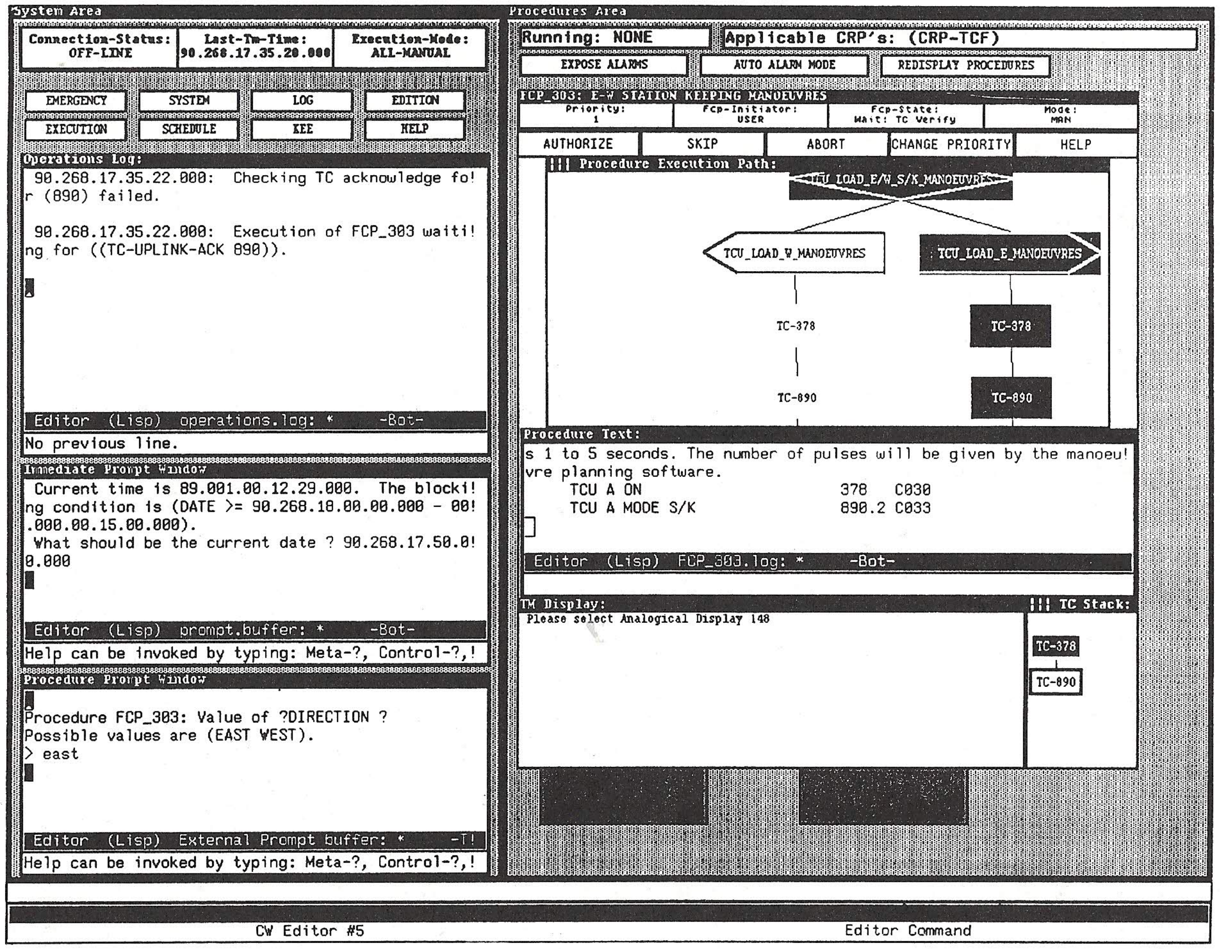


Figure 1. Architecture of the EOA system.



```
 90.268.17.35.22.000:  Checking TC acknowledge fo!
r (890) failed.

 90.268.17.35.22.000:  Execution of FCP_303 waiti!
ng for ((TC-UPLINK-ACK 890)).
```

```
Editor  (Lisp)  operations.log: *      -Bot-
No previous line.
```

```
 Current time is 89.001.00.12.29.000.  The blocki!
ng condition is (DATE >= 90.268.18.00.00.000 - 00!
.000.00.15.00.000).
 What should be the current date ? 90.268.17.50.0!
0.000
```

```
Editor  (Lisp)  prompt.buffer: *      -Bot-
Help can be invoked by typing: Meta-?, Control-?,!
```

```
Procedure FCP_303: Value of ?DIRECTION ?
Possible values are (EAST WEST).
> east
```

```
Editor  (Lisp)  External Prompt buffer: *      -T!
Help can be invoked by typing: Meta-?, Control-?,!
```

```
s 1 to 5 seconds. The number of pulses will be given by the manoeu!
vre planning software.
     TCU A ON                          378   C030
     TCU A MODE S/K                    890.2 C033
```

```
Editor  (Lisp)  FCP_303.log: *      -Bot-
```

```
Please select Analogical Display 148
```

```
CW Editor #5                                              Editor Command
```

Figure 2. EOA prototype system screen layout for daily operation.

In the **System Area** all system level information and EOA operational information is displayed, and most user dialogue takes place here. In the basic layout, windows for querying the user **Prompt Window**, for logging the operations and tests, are also placed in the system area. However, these can be moved around by the user. The system area contains an array of buttons used to select system functions.

In the **Procedures Area** the active procedures are displayed, and progress of the procedures is monitored. The area contains, for each active procedure, a group of windows for the execution and control of that procedure. The procedure is displayed both textually in the Text window, and graphically as a flow diagram in the **Procedure Execution Path** window, where the part of the procedure already executed appears in highlighted form. There are also windows displaying the information to and from the satellite (**TC Stack** and **TM Display**) window. For each procedure, there is an array of buttons for the control of the procedure (authorize an action, skip a step, abort procedure, etc.).

Spacecraft engineers can use another EOA screen[2] to display and edit knowledge bases. It includes a syntax-driven editor permitting procedures to be written.

4. KNOWLEDGE REPRESENTATION

The knowledge structure in the EOA has been organized so as to provide satisfactory solutions to the specific problems of procedure generation and execution.

Procedures can be structured as sequences of *steps*. Each step implements a piece of a procedure, and contains tests, actions (TC uplink, display message, etc.), conditional statements, go-to-statements, iterations, and transfers to other pieces of information (e.g., other procedures).

With conventional sequential programming techniques, the order of task execution within a program is entirely determined by the control structure of the code. Conventional programs are rather nonresponsive to unanticipated situations, and lack flexibility. The EOA approach is to describe each procedure or part of procedure as a set of schematic instructions (called *scripts*), which are expanded (or interpreted) in the context of the execution. Each schematic instruction describes a *goal* that the system will try to achieve in executing the procedure. An inference mechanism provides a means for directly using the knowledge in the system to reach the desired operational goals, through choices of applicable knowledge.

The declarative sematics, which is used together with the inference mechanism, provides good flexibility and allows for incremental changes to the system, as well as explanatory capabilities.

Another issue that comes with conventional programming techniques is that pieces of code (e.g., subroutines) are named or labelled with arbitrary names that have to be unique. The drawback of this approach is that the link between a piece of code and its functionality may be lost, hereby loosing software engineering and explanatory possibilities. In the EOA, each set of schematic instructions (or scripts) is attached to a name that specifies its goal. This goal is used by the inference mechanism so as to achieve the desired operational goals. Therefore, the EOA is *goal oriented*.

However, attaching a goal to a script is not sufficient. There may indeed be many ways to achieve a goal, each way being applicable in a specific context. A *context* is a set of facts that represent the state of the world, as it is affected by the procedure. With conventional programming techniques, only the control structure allows to call

[2]By "EOA screen" we mean a specific layout of the workstation display.

different subroutine conditionally. With the EOA, each script has attached not only a goal but also a context specifier, which specifies in which circumstances the script can be used to achieve the attached goal. During procedure execution, it is the inference engine that identifies, for an invoked goal, which script is applicable with respect to the current execution context.

A *context specifier* is defined as a list of variables with desired values. This implementation paradigm has been chosen so as to achieve a double purpose. As explained above, the aim is to provide a deterministic and unambiguous definition of the context where the script is applicable. The list of pairs (variables, desired values) represents the facts that have to be true in the context of the execution. The context specifier may also be needed to query supplementary information, as needed for the execution of the script. It is possible in the script to specify the context for a call to a goal, or to modify the current context. Each context instruction is lexically scoped. Therefore, the EOA is also *context oriented.*

The execution of a piece of a procedure described by a script calls for many other pieces of information that are explicitly or, at times, implicitly stated in a conventional procedure. These pieces of information have been formalized using the Theory of Plans (Wilensky, 1983). The selected types of information are:

- *Pre-execution* checks, which have to be true before continuing the execution of the piece of procedure;
- *Execution constraints*, which have to remain true throughout the execution of the piece of procedure;
- *The script itself*, which describes the checks or actions to be performed, and goals to be achieved with this piece of procedure;
- *Postexecution* checks, which have to be true immediately after the execution of the piece of procedure;
- *Postexecution* constraints, which have to remain true after the execution of the piece of procedure, until they are unset by another piece of information.

It appears then that many pieces of information are attached to a given goal. In the EOA, the chosen implementation paradigm is to group all this information into an object, whose facets will hold the various types of information (Figure 3).

As previously explained, several procedures may exist, allowing the achievement of a given goal in different contexts. These procedures may share common pieces of information. Therefore, they are grouped in a hierarchy, where the parent procedure holds the common piece of information, including in particular the goal specification. The common information is inherited down the hierarchy, until overwritten by local information specific of a procedure or subhierarchy of procedures. Procedures are thus implemented as a hierarchical library of objects.

The objective of the knowledge acquisition process is then to feed into each object complete expert knowledge, so that each object contains necessary and sufficient information for object selection and object execution. The sources of knowledge can be spacecraft design (e.g., for the various subsystems of the spacecraft), or operational experience (e.g., general technical expertise of mission controllers).

To conclude, the EOA is *object oriented*, to provide an efficient and convenient implementation for procedural expert knowledge. The EOA project shares common goals with PRS (Georgeff, 1985, 1986); namely it aims at building a system that explicitly represents and reasons about procedural knowledge. The EOA approach is less general in the sense that control knowledge is not represented as explicitly as in PRS (e.g., the fact that TC failure alarms have priority over COL alarms must be

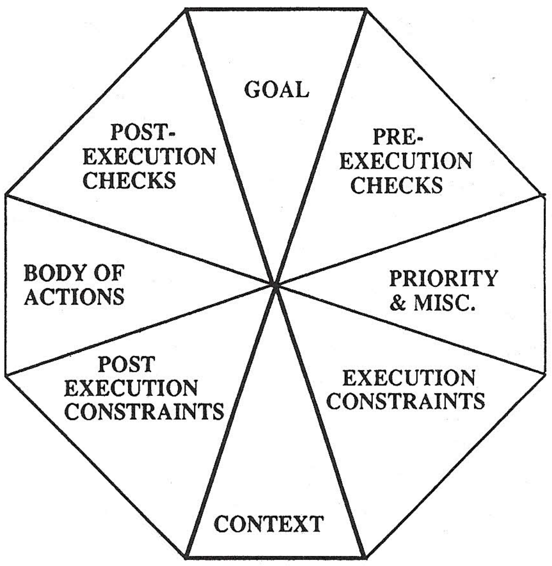


Figure 3. EOA prototype system: Object facets.

modified by the implementor). On the other hand, the EOA language is richer. It permits complex procedures with iteration and conditionals that look quite similar to real procedures. This facilitates manual validation of procedures.

5. EXECUTION OF FLIGHT PROCEDURES

This section illustrates how the knowledge implemented in the EOA is used for the execution of flight procedures.

5.1. Procedure generation

The system is used in a goal-oriented way. The user can enter a goal corresponding to a procedure or a step. The user is prompted if the specified goal does not match unambiguously one of the goals known by the system.

When one goal is unambiguously identified, its applicability with respect to the current context is examined by the system. To do so, the system collects the object or hierarchy of objects that are able to perform the specified goal. Then, using the context specifier of each object, it tries to find at least one applicable object. The introduction of required information is done through interaction with the user, whenever information is not available in the system. If no object is found applicable, the system leaves the procedure execution mode and enters a procedure generation mode (to be developed) where the system tries to construct a small procedure to set the world in a configuration compatible with the specified goal. This is typically a planning process, with chaining of "operators" from one state to another.

When a convenient object has been found to achieve the initial goal, its execution is initiated. The instructions described in Section 4 as pre-execution checks are initially performed. Execution constraints are asserted and checked periodically. The most important part of the execution is the interpretation of the script. As mentioned in Section 4, the script contains a set of instructions, such as actions (Telecommand uplink), checks (e.g., Telemetry Values), conditional statements, and calls to other goals. Finally, postexecution check instructions are performed and postexecution constraints are asserted.

In the process of cascading the initial goal into subgoals specified in the script, the inference engine will recursively try to achieve the subgoals. Each subgoal can be called within a specific context, by locally amending some aspects of the current context. The initial goal will be considered as achieved as soon as all goals in the cascade of subgoals are achieved. Achieving each subgoal is done sequentially, respecting the procedural order described in the script.

An interesting feature is that the inference engine used to perform script instructions and cascade into subgoals is based on an interpreter that is interrupt driven. Whenever the execution of instructions finds some reason to stop (e.g., wait for the right time or detection of anomaly), the returned code contains:

- The *context* of the current execution;
- The *continuation* of the execution, that is, all the instructions to be executed as soon as the execution resumes.

This is a very convenient mechanism for explaining what has been done and what remains to be done. When something goes wrong in the execution process, the system analyses the cause of the procedure interruption to assess which measures are needed to enable the resumption of the execution.

As an example, when the system is waiting for the right time to initiate some action, no special action has to be performed. In the same way, when some telecommand has just been uplinked, and the corresponding telemetry check fails, nothing has to be done since at least one telemetry flow has to come down before the telemetry actually appears as changed. On the other hand, if the telemetry check still fails after reception of several telemetry flows, the system reacts to this anomaly. This may result in uplinking again the Telecommand, or entering a complex diagnostic process. In the same way, when no object is found applicable for the achievement of a given goal, the system may ask the user to confirm the goal specification together with the call context, and later on, initiate the construction of an appropriate new procedure.

5.2. Spacecraft state monitoring

In a first phase, the control of the spacecraft state is done through the verification of the values of a large set of telemetries, generally grouped into Analogical Displays in the MSSS workstations. The EOA does not copy all the TM verification carried out by the MSSS, but focuses on verifying selected parameters in order to assess the progress of procedures. The basic reaction to an unexpected telemetry value is described in Section 6.

5.3. Execution scheduling

The execution of procedures may be time driven. For example, the performance of eclipse operations has to comply with a precise timing. The EOA provides functions for time monitoring.

Another type of scheduling problem exists when time constrained procedures compete for execution. Another example is a CRP being initiated while a FCP is being performed. The EOA could include functions to manage earliest start dates (ESD) and latest completion dates (LCD), and implements heuristics to prioritize a procedure against another.

When performing procedure generation, the EOA may also need to manage scheduling aspects in order to verify the feasibility of the generated procedures.

6. ALARM PROCESSING

A key skill for spacecraft control is the ability to react quickly to any kind of alarm from the MSSS. Reactions range from simple actions like calling an expert for help, to important decisions like ignoring an alarm or choosing and executing a contingency recovery procedure. For each possible alarm, the spacecraft controller can find in the FOP a sequence of actions that can be done. Since these actions have been written conservatively they often lead to a call for help.

One goal of the EOA is to provide assistance to increase reliability, speed, and scope of day alarm processing. The implemented prototype deals with alarm combinations that have been foreseen in advance and can be treated by existing emergency procedures.

In compliance with the philosophy of the project, it was not attempted to design new alarm mechanisms but instead to provide assistance to users of the existing control center. The EOA must deal with all the alarms of the MSSS. Nevertheless, it is also possible to implement new kinds of alarms. This can be done by one or several procedures running permanently, to perform a trend analysis, for instance. The MSSS generates two kinds of alarms:

1. *TC verification failures* when something goes wrong in a TC uplink, and
2. *Out Of Limit (OOL) alarms* when some parameters go outside some limits defined dynamically on the MSSS or have inconsistent values.

Thanks to the multiprocess capability of the EOA, alarms are processed as soon as they arrive from the MSSS without interrupting procedures. One important problem is to know which alarm to focus on, since an alarm rarely occurs alone. For this, the EOA uses different kinds of knowledge: priority number on OOLs, alarm prediction included in procedures, and mode definitions, which can explain that an alarm has been caused by another one. In any case, the user can focus on the alarm of his choice and can discard nonrelevant alarms.

Once an alarm has been selected, a set of rules are evaluated to generate a list of all the procedures that can be applied. Each procedure can have such a rule, written in the EOA language, to indicate whether it is appropriate to enter the procedure. If the rules have been well written, at most one procedure will be applicable. In the other case, the user has to arbitrate which procedure to start or has to call for help.

The alarm processing scheme described here can be improved in many ways, one of them being the interface with a diagnostics expert system such as DIAMS (Haziza, 1988), which incorporates spacecraft design knowledge for those situations that require a sophisticated diagnostics method that cannot be implemented conveniently as a procedure.

Another direction for improvement consists in developing a more sophisticated priority scheme. When alarms occur during the execution of the FCP, the EOA may have to start a CRP in parallel. The default rule is to give priorities to CRPs over FCPs. Clearly this can be improved, since a FCP may have crucial hanging constraints (e.g., turn a component off). One solution would be to acquire from experts priority numbers to be attached to each step of a procedure. A more ambitious scheme would be to infer these priorities from design and operational knowledge.

7. CONCLUSION

A large part of the functions described in this paper have already been implemented, leading to a prototype that covers assistance to spacecraft controllers for normal daily operations and simple non-nominal situations. Obviously, there is room for a lot of improvements and extensions, but the EOA has already shown interesting results.

Eight procedures that are quite representative from the MARECS-B2 FOP have been implemented, including station keeping, eclipse operations, recovery from payload switch-off, and recovery from automatic reconfiguration. The syntax driven procedure editor and the knowledge base inspectors together with the methodology for procedure generation should permit this set to be extended.

A flexible multiprocess architecture for real-time expert systems makes possible the communication and integration with the ESOC MSSS. Evaluation of the EOA is carried out in close cooperation with potential users, namely the operators and spacecraft engineers working at ESOC. There are good hopes that the EOA will demonstrate the feasibility and utility of knowledge based operator assistance for spacecraft control.

REFERENCES

Georgeff, M. P. (1985). Reasoning about procedural knowledge. In *Proceedings of the AAAI 85 Conference* (pp. 41–49).

Georgeff, M. P. (1986). Procedural knowledge. *Proceedings of the IEEE 86 Conference, 74*(10).

Haziza, M. (1988). An expert system shell for satellite fault isolation based on structure and behaviour. In *Proceedings of the ESA Workshop Artificial Intelligence applications for space projects*. ESA ESTEC.

Wilensky, R. (1983). *Planning and understanding: A computational approach to human reasoning*. Reading, MA: Addison-Wesley Publishing Company.