SCOS II: ESA'S NEW GENERATION OF MISSION CONTROL SYSTEM

M Jones, N C Head
Flight Control Systems Department, European Space Operations Centre (ESOC)
Robert Bosch Str. 5
64293 Darmstadt, Germany

K Keyte, Vitrociset SpA, Rome, Italy
P Howard, Science Systems Ltd, Bristol UK
S Lynenskjold, Computer Resources International (CRI), Birkerød, Denmark

ABSTRACT

New mission-control infrastructure is currently being developed by ESOC, which will constitute the second generation of the Spacecraft Control Operations system (SCOS II). The financial, functional and strategic requirements lying behind the new development are explained. The SCOS II approach is described. The technological implications of these approaches is described: in particular it is explained how this leads to the use of object oriented techniques to provide the required "building block" approach. The paper summarises the way in which the financial, functional and strategic requirements have been met through this combination of solutions. Finally, the paper outlines the development process to date, noting how risk reduction was achieved in the approach to new technologies and summarises the current status future plans.

1. INTRODUCTION

This paper describes the new infrastructure for Mission Control Systems which is being produced at the Operations Centre of the European Space Agency in Darmstadt. This infrastructure is the second generation of the Spacecraft Control Operations System (SCOS) and will replace the current generation SCOS I (Mullet et al., 1990) for all new ESOC mission implementations. First candidate client missions are ARTEMIS (a data relay mission), ENVISAT (an earth observation mission) and HUYGENS (a Titan probe). The paper concentrates on the programmatic and the main functional aspects; technical details related to the implementation techniques and technologies can be found, for example, in Keyte (1994).

2. WHY SCOS II?

In order to provide the context for a discussion of SCOS II and its features it is important to have an understanding of the motivations behind the development of "yet another" set of mission control infrastructure and of the general operational environment in which SCOS II based systems will be used. The main factors which led to the SCOS II development are broadly as follows (Jones et al. 1993):

- financial:

  The development of Mission Control systems based on ESA's current generation of infrastructure software (SCOS I) is costly. This is due, at least in part, to the inflexibility of the SCOS I system structure and the resulting difficulty of customising SCOS I software to a mission and of adding mission specific software to the basic system.

- functional:

  The increasing complexity of missions requires a corresponding increase in the capabilities of the control systems. For the same reason the effort involved in preparing and monitoring mission operations is increasing.

- vendor independence:

  The cost and flexibility of computer hardware for previous systems have been item of concern. The centralised, host-based, architecture of these systems which, resulted in the use of large mainframe computers to support mission operations. This resulted in dependence on the operating system and basic software provided by vendors of the particular host computers chosen, thus effectively tying the Agency to these vendors.

The major drivers for SCOS II can thus be summarised as reduced cost per mission with increased flexibility and portability.

3. THE SCOS II PROJECT: OVERALL APPROACH AND PROGRAMMATICS

The SCOS II project began in 1990 with the general aims outlined in the previous section. A large investment of effort was made to define a comprehensive set of users requirements and associated operations concepts resulting in a very substantial User Requirements Document (URD). This work is outlined in a companion paper (Kaufeler et al. 1994). At an early stage a decision was made to use object oriented analysis, which with its focus on the Problem Domain, encouraged interaction between the User Requirements work and the software requirements analysis. This led to the need to cope with evolving user requirements and overlapping development phases. How this was resolved in terms of software development approach is discussed by Pujo et al. (1994) and Symonds et al. (1994). The implementation language is C++.

The implementation is proceeding in a series of releases, which will successively add functionality to cover the all areas of the URD. Release 1, due in early 1995 includes the new concept of "system elements" explained in the next section and will have equivalent functionality to the existing SCOS I infrastructure , including in addition telecommand functions (missing from SCOS I) and more modern user interfaces. A "Proof of Concept" prototype was developed and demonstrated in early 1993 to verify the feasibility of the distributed system technology. At the end of 1993 a "telemetry demonstrator" was available, which showed telemetry processing basic functions and associated man-machine interface.

Release 2 (1995-1996) and Release 3 (1997-1998) will add further advanced functionality including areas such as mission planning which have never been treated generically within ESOC before.

4. WHAT IT DOES: THE PROCEDURE-ORIENTED OPERATIONS APPROACH AND SYSTEM ELEMENTS

As in most Operations Centres, ESA mission operations are centred around "procedures" which are executed automatically subject to the occurrence of specified events (usually anomalies) in either the flight or ground segments. Non-procedural (i.e. manual) operations are

reserved for those inevitable cases where appropriate procedures have not been foreseen.

Mission operations engineers usually regard spacecraft as being composed of a number of systems, sub-systems or assemblies. The process of mission control consists of performing actions (either active, controlling ones or passive, monitoring ones) with one or more such systems or sub-systems. Each of these actions is driven by an appropriate procedure.

A particular procedure may "call-up" other procedures to perform some portions of its work. Similarly, a procedure may be called by other, higher level, procedures to perform some actions on their behalf. Loosely speaking, the set of procedure for a mission can be viewed as forming a tree-like hierarchy whose structure is very closely related to the hierarchy formed by the system, sub-system and assembly relationships of the spacecraft itself.

SCOS II infrastructure directly supports the modelling of systems, sub-systems and assemblies. These components are all represented as objects referred to as "System Elements". The relationships between these Elements are stored in the mission database in a tree-like structure (see fig. 1). System Elements are used in a number of ways in the SCOS II system.

4.1 Abstract Monitoring & Activities

The execution of a typical procedure consists of three major phases:

- setup: checks to ensure that preconditions for execution of the procedure are satisfied and that required tools are available
- execution: use of the tools to perform the activity (this may be a passive, monitoring only activity)
- assessment: check that the results of the activity are as expected and that all required post-conditions are satisfied.

SCOS II System Elements provide support for all three phases, hiding the use of subordinate System Elements from the user once this use has been defined in the database:

- a System Element provides an high level view of the current status and mode of the unit which it represents; initiation criteria for the procedure can be expressed in terms of these

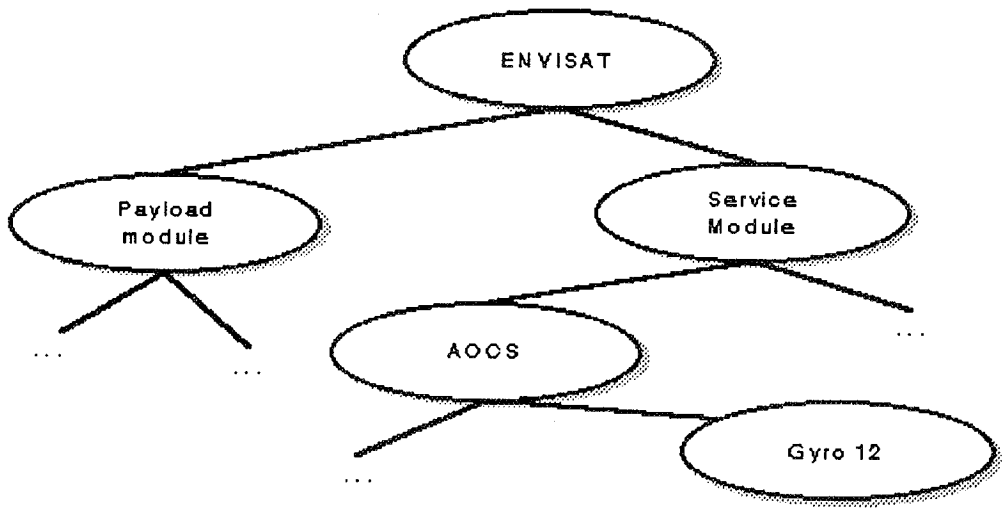

Figure 1 A simple hierarchy of System Elements

values (for example "is the AOCS in fine pointing mode?")

- a System Element provides a set of high level activities which can be initiated either directly or on behalf of other procedures (for example 'perform thruster cat bed preheat')

- a System Element, again based on high level status and mode assessments, allows simple assessment of the success of the activity (for example, 'is the AOCS now in sun-pointing mode with nutation < 0.1 degrees?')

4.2 Event-based procedure initiation

SCOS II provides capabilities for System Elements to signal the occurrence of "Events" (a mode change for example) and to associate "Actions" with these occurrences. One of the types of Action which will be supported is the initiation of a procedure (referred to as an Activity in SCOS II) which may be either diagnostic in nature or may, in the case of unexpected or critical events, take some form of safe mode initiation.

4.3 Building system elements - element templates

Often a spacecraft will have a number of similar devices (gyro's for example) which have an essentially identical set of operational procedures; differences are only to be found in the specific parameters and command encoding details. SCOS II supports the concept of System Element Templates which contain a master definition of the Element behaviour with empty slots for such specific data. Populating the mission database for each of the specific instances of the templated unit is then a matter of 'filling in the blanks' in the template. This should greatly ease the version control of the database as updates need only be applied once to the template rather than several times. testing of the database will be similarly reduced in cost.

4.4 Element Connections & Dependencies

Many operational constraints and checks in traditional systems are centred around a relatively small number of issues (power status, redundant unit status etc). The configuration of a traditional system to deal with these consumes a significant proportion of the overall configuration effort. SCOS II explicitly supports the concepts of relations between System Elements for (a) power supply and consumption, (b) redundant sets of devices and (c) data routing and forwarding.

These relations, once defined in the database, allow the system to automatically perform many of the processing and control functions which have previously required explicit implementation. Again, this will reduce the cost of configuring the system for a specific mission

4.5 Navigation at the user interface

The System Element hierarchy is also used to provide structure for the user interface navigation facilities; the MMI allows navigation through the database and through the online parts of the system by following the various links between the System Elements. This allows easy movement from say a gyro pack to its power source or to its redundant unit.

4.6 Procedure manipulation

The actual text of the procedures will be made available via the System Elements. For example, when viewing the contents of the AOCS System Element the user will be able to access all AOCS related procedures directly from the MMI rather than via some separate application and a numbering convention to locate AOCS items.

4.7 Integration of ground & flight segments

Perhaps most importantly the concept of System Elements has been extended to allow their use to represent also portions of the mission ground segment (for example ground station equipment, wide area networks, SCOS II workstations themselves). This allows integrated monitor and control of a complete mission system from a single position. A particular advantage of this is the possibility to merge actions for the flight segment with actions for the ground segment in a single SCOS II Activity in the same way as they are merged in the paper procedures of the current systems. An example of such a merged activity might be the AOS (Acquisition of Signal) for a low earth orbiting spacecraft. A simple summary of the steps involved might be:

1. Perform pre-pass dataflow tests (TC to station)
2. verify dataflow tests(in flight & station TM)
3. Transfer orbital elements to antenna controller (TC to station)
4. Select Program Track (TC to station)
5. Wait for notification of receiver lock (in Station TM)
6. Initiate uplink sweep TC to station)
7. Wait for onboard receiver lock (in flight TM)
8. Select Auto Track (TC to station)

Previous mission systems have implemented a variety of *ad hoc* approaches to such combined control and monitoring of flight and ground segments which however have confirmed the benefit of such integration.

5. CUSTOMISATION FOR MISSIONS

The greater capabilities of SCOS II are obtained at the cost of extra information required to set up the system during mission preparation.

To minimise this cost, the System Element concept described in sect. 3 offers an obvious vehicle for implementation of mission specific requirements. System Elements can be viewed as "building blocks" which can serve as a basis for the implementation of these requirements. They can be extended and configured in two different ways (a) by specialising building blocks and (b) use of Operations Language (Baldi et al., 1994):

5.1 Specialising Building Blocks

This is done by a mission specific software engineering team. SCOS II is implemented following an object oriented approach; in particular the System Element is the base of a class hierarchy which allows for progressive,incremental specialisation towards a final System Element representing, for example, an onboard computer for the 'XYZ' mission - see fig 2. This is in fact the genuine software reusability offered by object oriented techniques.

In the long term it is hoped to achieve further reuse of specialised building blocks by sharing them between missions which use the same units in the flight and ground segments. Standardisation of mission

hardware units could thus bring much larger cost savings than any of the measures taken to improve the efficiency of implementation of a single mission system.

5.2 Use of Operations Language

Operations engineers can also perform customisation to make limited changes to existing building blocks. SCOS II allows configuration of many aspects of a System Element through the use of the SCOS II Operations Language. This language is a synthesis of previous languages used in both operations and checkout and allows the production of not only of procedural or algorithmic parts of System Elements (for example command sequences, synthetic telemetry parameters, verification algorithms) but also rule-based parts which allow the identification of Events (described above) leading to the triggering of Activities. The Operations Language may be either compiled or interpreted; this choice will be made by the operations team, based on the conflicting needs of performance and ease of modification for each System Element.

6. HARDWARE CONFIGURATION

Initial installation of SCOS II at ESOC will be on a Local Area Network of SUN Sparc 10 workstations running the Solaris 2.3 operating system. However SCOS II is being implemented to be portable across almost any Unix (System V or POSIX compatible) workstation platform. Parts of the system developed to date have been successfully run on SUN IPC and IPX platforms; respectable performance has been achieved without any particular attention to optimisation. Small parts of the system have also been run on Intel 486 based machines

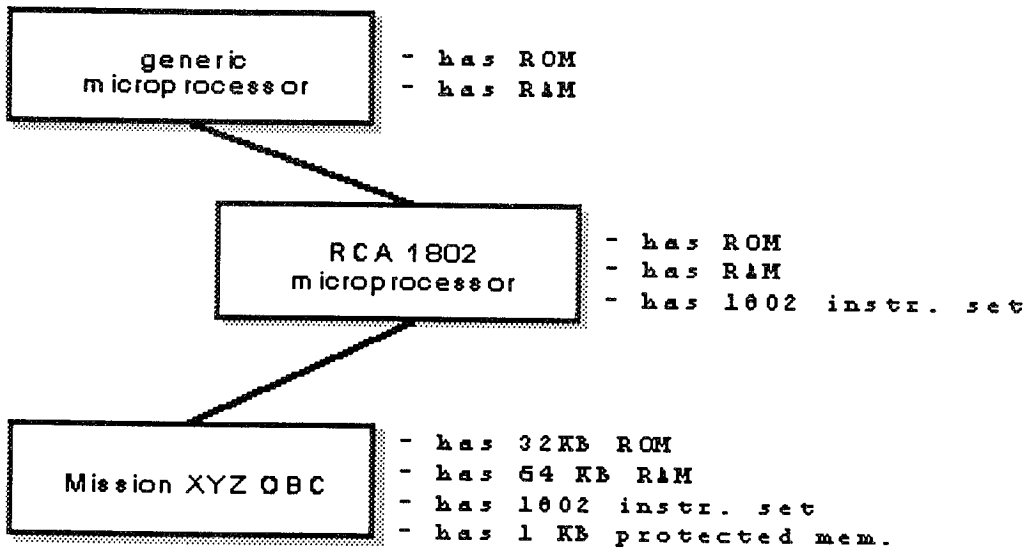

Figure 2 An example of progressive specialisation

(running a largely System V compatible Unix clone); initial indications are that performance is comparable to that of the smaller SUN machines and that this is also a viable platform for missions with low data rates (less than 10 kbits/s) and without exotic science data processing needs.

Although designed to be a distributed system running on large networks of processors SCOS II is also able to run on a single workstation (although obviously no redundancy is available in such a configuration and some performance limitations are to be expected) while still supporting all functions of the distributed system. No software or database modifications are needed to run in this manner. This configuration is known, informally, as "SCOS II-in-a-box".

7. CONCLUSIONS

In conclusion it can be said that the SCOS II project is the first attempt within ESA to provide a highly configurable and reusable software toolbox for building mission control systems. Its main aim has been to reduce costs, increase functionality and achieve vendor independence. To achieve *cost saving* mission specific costs, it uses object-oriented and other modern techniques to increase reusability and allow easy customisation. Greater *functionality* is provided; even in its Release 1 version there is more functionality than in previous ESA mission operations infrastructures and this will improve further with Release 2 and 3 work foreseen in 1995-1998. *Vendor independence* is provided through choice of UNIX and suitable implementation measures to achieve portability. This means that SCOS II could be used for a wide range of missions range from large ones requiring 30-40 workstations and high data rates down to small, low cost missions based on one or two low-cost platforms. Extension of its use to other areas of the ground segment or mission lifecycle (eg. spacecraft checkout, backup control centres at station, ground segment control) hold out the possibility of further rationalisation and cost saving.

REFERENCES

Baldi, A., Elgaard, D., Lynenskjold, S., Pecchioli, M., *SCOS II OL: A Dedicated Language for Mission Operations, (1994),* Proceedings of the Third International Symposium on Space Mission Operations and Ground Data Systems, November 1994, Greenbelt, Maryland

Jones, M., Head N.C., Keyte, K. & Symonds, M., (1993) *SCOS II - ESA's New Generation of Mission Control System*, ESA Bulletin 75

Kaufeler, P., Pecchioli, M., & Shurmer, I. (1994), *SCOS II - ESA's New Generation of Mission Control System- The User's Perspective*, Proceedings of the Third International Symposium on Space Mission Operations and Ground Data Systems, November 1994, Greenbelt, Maryland

Keyte, K.P., (1994) *SCOS II - A Distributed Architecture for Ground System Control*, Proceedings of the International Symposium on Spacecraft Ground Control and Flight Dynamics,J. Brazilian Soc. of Mech. Science, XVI(319).

Mullet, B., Kaufeler, J.F., Bowers, J. & Ahier, M.,(1990) *The SCOS: A Configurable Infrastructure for the Control of Spacecraft Using Packets* , ESA SP-308(29)

Pujo, O., Head N.C. & Jones, M., (1994) Object Oriented Design, *The Software Life Cycle and Software Engineering Standards*

Symonds, M., Lynenskjold, S., Müller, C., *SCOS II: An Object Oriented Software Development Approach, (1994),* Proceedings of the Third International Symposium on Space Mission Operations and Ground Data Systems, November 1994, Greenbelt, Maryland