SCOSII: ESA'S NEW GENERATION OF MISSION CONTROL SYSTEMS

J. F. Kaufeler and N. C. Head

Flight Control Systems Department
European Space Operations Centre
Darmstadt, Germany

ABSTRACT

The paper describes the next generation Spacecraft Control System infrastructure (SCOSII) which is being developed at the Operations Centre (ESOC) of the European Space Agency (ESA). The objectives of the new system and selected areas of the proposed hardware and software approach are described.

Key Words: software, mission control, object oriented, infrastructure

1. INTRODUCTION

The Operations Centre of the European Space Agency is currently developing the next generation of its Spacecraft Control and Operation System (SCOS).

Over the years ESOC has made great efforts to develop and use mission independent and configurable control system software. This software is also sometimes referred to as multi-mission software, in that it can be configured to support different missions. The first system of this nature was the Multi Satellite Support System (MSSS) which entered service in 1977 with the launch of GEOS-1; this was followed by the first SCOS systems which are used to support the Hipparcos, ERS-1 and Eureca missions. It will continue to be used for all launches until approximately 1995. Ref. 1 provides an overview of the history of ESOC spacecraft control infrastructure.

This paper describes some of the principle characteristics of the next generation system (SCOSII) and discusses the improvements over the earlier generation systems.

Two viewpoints are taken for these discussions, corresponding to the two major user communities; the mission software engineers who must configure and extend the infrastructure in order that it be usable for a specific mission and the end users, that is the operations engineers who must control the spacecraft with the help of configured software.

2. OBJECTIVES

2.1. Functional improvements

The end users require a number of improvements and extensions to the functions provided by the system in order to mitigate the increasing workload resulting from the complexity of modern spacecraft. Use of onboard microprocessors to provide some level of autonomous functions on the spacecraft is an example of such complexity.

The essence of any control system is the description of the device (in this case a spacecraft and its components) to be controlled and its expected and desired behaviours under various conditions. With growing complexity of spacecraft the expressive power of the database oriented description techniques used in previous generation systems is no longer adequate.

It is also clear that there is a need for improved performance and flexibility of the Man Machine Interface elements of the system; in particular inadequacies in the MMI have sometimes slowed down the response of operations engineers to

the contingency situation.

These lacks are seen to be of two types:

> insufficient ability to obtain synthetic overviews of high level status of the spacecraft and reduce the information overload on the operations engineers
>
> slow response when reviewing historical data in order to evaluate spacecraft performance or to locate the cause of anomalous events.

In view of these problems two of the major goals of the new infrastructure are to (1) improve both the functional scope and flexibility of the system and (2) provide greatly increased performance as seen at the man machine interface.

2.2. Maintainability

The software engineering community have voiced concerns about the maintainability and configurability of the existing systems; the effort involved in adapting these systems to the needs of a specific mission is significant, it has to be repeated (at least in part) for each update of the infrastructure which is both tedious and error prone.

Typical activities involved in configuring the infrastructure for use with a specific mission fall into two broad categories; specification and development of new control and analysis facilities in support of special characteristics of the target spacecraft which cannot be handled by the generic software (for example the control and monitoring of an Onboard Computer) and modifications to existing portions of the system to account for small differences in behaviour (for example different command encoding schemes or unusual telemetry layout policies).

The new infrastructure will apply some of the more recently developed software implementation approaches and technologies in an effort to reduce the effort involved in both development and maintenance of such specific mission configurations.

3. TECHNOLOGIES

There have been some recent advances in the areas of both spacecraft and software engineering which contribute to the concepts of the SCOSII system; with hindsight it is clear that previous versions of the infrastructure have been handicapped by the lack of these advances.

3.1. Standardisation

The application of standards in spacecraft design and operations, either explicit or *de facto*, will obviously greatly ease the provision of infrastructure software systems by helping to reduce differences between missions. The SCOSII development takes advantage of a number of such standards.

3.1.1 Explicit standards

The Consultative Committee for Space Data Standards (CCSDS) has produced definitions for the space/ground link protocols (Ref. 2) and transport data formats; although concerned mainly with the issues of data transport they will considerably reduce the proliferation of transport protocols and simplify the job of interfacing to the ground station equipment which (in ESA systems) provide the transport services.

ESA's Committee for Operations and EGSE Standardisation (COES) is working on a standard (Ref. 3) which will cover many presentation layer issues; this will prescribe the telemetry and telecommand data encoding schemes for the data contained within the CCSDS packets. It also defines the protocols to be applied for control of certain onboard applications (Master Timeline management or Onboard monitoring for example). These matters are of direct relevance to control systems which have the job of presenting the telemetry contents to the engineers and controlling the onboard applications. Despite its preliminary nature this standard is being adopted as one of the foundation stones of the SCOSII system.

3.1.2 de facto Standards

The European space industry is developing, in cooperation with ESA, a set of effectively standard components for the construction of space-

craft (typical "almost" standards are to be found in the onboard data handling systems as well as in low level components such as command decoders). As these devices are employed on more and more spacecraft it is planned that the relevant portions of the ground control systems will also be re-used.

ESOC has also published a set of guidelines (Ref. 4) for spacecraft manufacturers describing the operational constraints which should be respected by ESA spacecraft if they are to be operated by ESOC. This document is in the process of being transformed into a COES standard.

3.2. Object Orientation

The Object Oriented approach to analysis, design and implementation of software systems has now matured to the extent that it has been adopted for the implementation of SCOSII.

The intention is to make use of the inheritance and polymorphism properties of the technique to promote the use of specification (and implementation) by "difference". It is expected that such an approach will have two benefits; firstly, the differences between a proposed operational facility and an existing one will be explicitly stated and reviewed and will be subjected to more intense scrutiny than if they were simply part of a overall description of the facility and secondly, the implementation and maintenance of the modified facility will be limited to the code which provides the differences in functionality.

These benefits will however only be realised if a thorough analysis of the problem domain is applied to new mission systems and if the various users of the infrastructure are prepared to resist the temptation to "re-invent the wheel" for each mission.

4. ARCHITECTURE

4.1. Hardware environment

The SCOSII system will be implemented on a network of Unix workstations. This is a major departure from previous infrastructure systems which have been centred around a host machine with, in some cases, intelligent terminals for user interfaces.

This change is motivated by two factors; the availability of large amounts of extremely cheap computing power compared to the centralised approach and a desire to achieve a system which is more tolerant of failure of either processor hardware or software.

The initial platform selection for SCOSII consists of SUN Sparcstations connected by an Ethernet network. Each workstation will have local storage (at least 1Gbyte per station), 2 or 3 colour screens and pointing device (mouse or track ball). The processors will run the SUN SOLARIS 2.x system which is System VR4 compatible and provides a number of services making it more appropriate for real-time or time-critical usage that the previously available SUNOS versions which were derived from BSD Unix.

A typical mission configuration will have one workstation for each operations engineer position plus a further 2-3 server nodes providing common functions. It should be noted that there will not be a Network File System (NFS) server; this has been identified as a potential single point failure. Each workstation will use some portion of its local disk space for storage of executable images and static configuration data and will obtain dynamic data from one of the functional servers located on the network as needed. It is hoped that a minimum service can be identified which can run without any support from these servers and allowing each individual workstation to function independently to a limited extent.

An important goal of the architecture is to ensure that the physical location of system functions (in particular the servers) is transparent to other applications. This will allow a small system for a simple, low data rate spacecraft to be configured on a single workstation.

4.2. Basic Software

The underlying approach to the structuring of the SCOSII software that of the client/server pair. Much of the functionality of the system will be provided by servers of various kinds (a telemetry server or an uplink server are possible examples). These will communicate with their clients over the network. In order to help keep the resulting network load within manageable limits the concept of a "broadcast" server has been introduced.

This concept makes use of the observed fact that, in a multi-user control system, most of the users perform roughly similar activities at roughly the same time and so will have very similar needs for server data. A broadcast server will therefore distribute its replies to all user nodes on the network with the assumption that it is, in most cases, directly relevant data and will avoid a subsequent repeat request for the data from each user workstation in turn.

As the broadcast data may not be immediately applicable when it arrives at a node (due to small differences of processing sequence for example) it will be stored in a cache which is maintained by each node in its virtual memory using the local disk storage as backing store. This cache is then checked before issuing possibly redundant requests to the server for the data.

The cache management policy applied at a particular node may be subject to tuning depending on the nature of the applications running on the node; an engineer performing evaluation of a manoeuvre which took place several days or months ago is unlikely to be interested in real-time telemetry broadcasts, similarly the active spacecraft controller workstation will probably be configured not to cache data older than a few days. These tuning activities are not visible to the broadcast server; these behave in the same manner at all times.

4.3. Basic system model

As noted earlier it is important for a control system to be able to model the behaviour the device being controlled to a sufficient level of detail. Previous systems have attempted to provide this flexibility by the use of a single "engine" driven by a set of data tables of predefined layout. The approach cannot, in practice, describe all kinds of spacecraft behaviour and has resulted in a large amount of mission specific software being produced to handle devices which are not describable in this way.

The SCOSII infrastructure is adopts a different approach, suggested to the designers by the techniques applied in Object Oriented analysis and supported by the technology available in the chosen object-oriented programming language (C++).

A spacecraft is generally viewed by the engineers as an assembly of sub-systems (and sub-sub-systems etc). The behaviour of each component subsystem is generally fairly well defined as are its relationships to other sub-systems on the spacecraft. Thanks to the capabilities of Object Oriented software and databases this view of the target has been adopted as the foundation of the SCOSII "database"; each sub-system is represented as an object in the database. These objects maintain connections of various kinds (contains, is_part_of, refers_to etc) to the objects representing the other portions of the spacecraft with which the sub-system interacts. The benefit

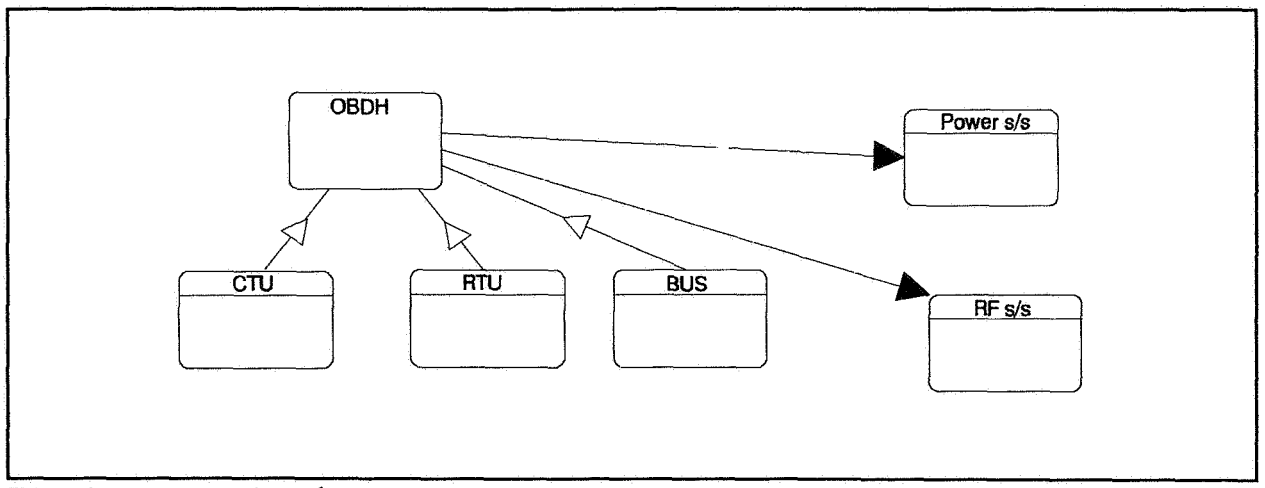

Figure 1 - Representation of an OBDH

of this becomes clearer when it is realised that each object may have a specific "engine" to define its behaviour and may use data structures best suited to the task of describing the object in question. A complete discussion of this topic is outside the scope of this paper but a brief example should help to explain the overall approach.

Figure 1 shows how a simple Onboard Data Handling (OBDH) sub-system might be represented in this manner. It contains several items (a Central Terminal unit, a number of Remote Terminal units, a data bus) and makes use of information from several other sub-systems (power, RF etc). This does not claim to be complete or fully correct. However it serves to demonstrate the principles involved.

The effort involved in setting up such models of a spacecraft will be large, even if sophisticated editors are provided, and it is necessary to take additional steps to help. As noted earlier, there is an advantage to be gained in specifying, and implementing, systems "by difference" from existing systems. This principle can also be applied to the modelling of the spacecraft. A basic set of foundation objects can be provided which can then be specialised to match a specific need for a specific spacecraft. This technique is referred to as "inheritance" in the Object Oriented view. Again a full explanation of this is outside the scope of this paper but an example should serve to clarify some of the issues.

Figure 2 shows a hierarchy of objects, using a notation loosely based on that of Coad/Yourdan (Ref. 5) which represent an onboard processor of an imaginary mission - DemoSat. At the highest level is an object representing a generic onboard processor which has a number (at least 1) of memory banks and a watchdog timer. At the second level is a multitasking processor which `is_a` generic processor and therefore has all the characteristics of a generic processor but, in addition, has descriptions of a number of Tasks which are able to run on the processor. Finally comes the DemoSat processor which `is_a` multitasking processor. The control logic for the DemoSat processor is similarly derived; as it `is_a` generic processor, for example, it is able to perform memory loads to its memory bank(s) (typically open the memory bank, send a number of load packets, close the memory bank again) and similarly understand the manipulation of the watchdog timer. The multitasking processor component pro-

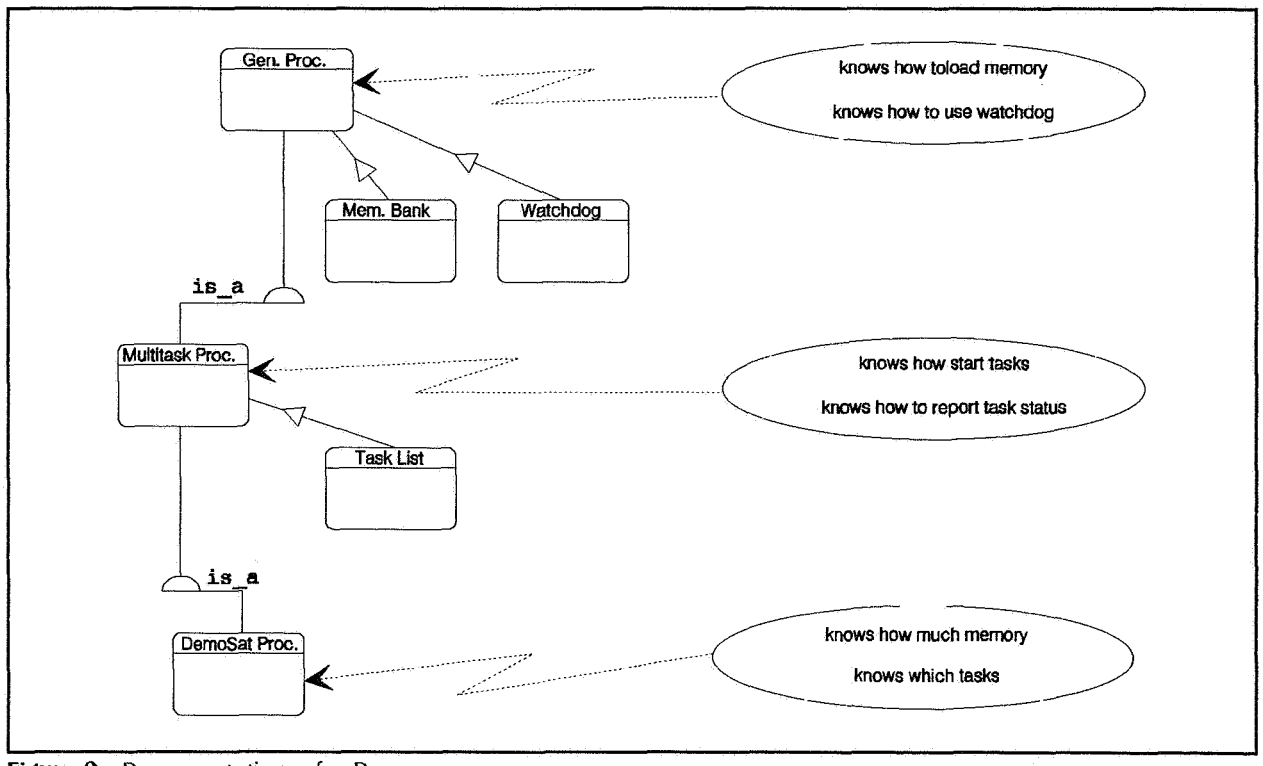

Figure 2 - Representation of a Processor

vides additional behaviours (for example start and stop of tasks, report on task status) and finally the DemoSat processor specific component provides information about the exact number of memory banks, the exact list of tasks which can be run, their constraints and so forth.

The major point of interest (and labour saving) is that the objects representing the generic processor and the multitasking processor have no mission specific elements and could be implemented once only; the implementation of the DemoSat processor object then requires only provision of those behaviours which are not covered by components or which are not standard (implementation by difference) and specification of particular characteristics (number of memory banks, valid task list etc) which are referred to but not defined by the higher level components.

It is clear that it will be necessary to perform a careful analysis of typical spacecraft in order to build a usable library of such generic objects which can be reused at some later date; the first clients of SCOSII may not reap all of these benefits and may actually be involved in slightly more work to participate in the analysis and creation of such standard building blocks.

4.3.1 Applications software

The actual applications tasks of SCOSII can be viewed mainly as gateways to the functions provided by the objects in the spacecraft model. To continue the example used above the Onboard Software Maintenance application will be mainly a shell providing a user interface to generate messages to the DemoSat object and providing an area of screen where the DemoSat processor will generate its visible representation.

Those parts of the representation which are arise from the generic processor (i.e. watchdog status) or from the multitasking processor (i.e. task status list) will not need to be reimplemented. This helps reduce effort and, perhaps more importantly, ensures a consistent user interface from one mission to the next wherever there are similar functions to be performed.

User interfaces in the SCOSII system will be provided via the X.11 & OpenLook standards with specific objects (a CTU object for example) providing the logic for processing input events and using a X-window provided to place their visible image. The image produced may vary depending on the context of the application; in some cases the current value will be needed, in other cases a "database" edit dialogue will be requested. The applications will be responsible for providing this context information.

5. SUMMARY

This paper has provided a brief glimpse of the plans for the next generation of the ESOC spacecraft control infrastructure - SCOSII.

It has discussed some of the motivations behind the decision to implement a new version of the infrastructure and has presented some of the areas where the project is planning to deviate from established practice in an attempt to improve the service offered both to the end users (the spacecraft engineers) and to the, oft forgotten, immediate users of infrastructure software who are responsible for the customisation and maintenance of the software for specific missions.

It remains to be seen whether all of the benefits hoped for by the SCOSII team will actually be realised in practice but initial positive responses have been received from all sides. We are taking a cautious approach however; the very first version of SCOSII will contain only functions known to the users from previous generations of infrastructure systems with introduction of the more advanced aspects being left until the safety net of backwards compatibility is complete.

6. REFERENCES

1 - Debatin,
SPACEOPS 1992

2 - CCSDS,
Packet TM & TC standards

3 - COES,
Packet Utilisation Standard

4 - SOIRD,
Spacecraft Operations Interface Requirements Document

5 - Coad/Yourdan,
Object Oriented Analysis