Title – Autonomous Operations Through Procedure Automation

Category – Autonomous Operations - II

Authors: Terma:: R. Patrick[i], Vitrociset SpA (Italy): F. Croce[ii]

ABSTRACT

The SCOS-2000 system, as developed by ESOC, has dramatically reduced the cost of Mission Control Systems for ESOC over the past 10 years – sometimes by a factor of as much as 10. This has been achieved by the increased standardization (CCSDS et al) and the use of lower cost technologies.

This is a considerable contribution to reducing the costs of the mission operations, but are there ways to find further savings beyond those already achieved? In this paper we cover two projects currently underway, which based on this SCOS-2000 infrastructure will help to further reduce operations and life cycle costs.

This paper describes an Automation tool – the ASE tool from Vitrociset – and how this has been integrated with the standard SCOS-2000 system. This will allow highly automated operations of both the spacecraft and the ground segment, allowing the mission operations team to only work during normal office hours. The cost saving is through the lower number of personnel required for the operations activities.

1 Introduction

This paper first gives a statement of the problem – how the operational concepts within ESOC and the resulting architecture of SCOS-2000 have been defined. We then describe the ASE tool that brings the automation to the system, and finally the two missions that have currently baselined the tool for use in their mission operations.

2 SCOS-2000

2.1 Introduction

SCOS-2000 is the latest spacecraft control software to be used by ESOC for it's mission operations.

The capabilities in SCOS-2000 are derived from the ESOC experiences in spacecraft operations since the 1970's. Prior to the introduction of SCOS-2000 as the de-facto baseline choice for all new missions, ESOC had developed two previous generations of systems.

The first was the MSSS system, developed on SEL-Gould machines and used for their first missions. This was followed by the SCOS-I system – based on a VAX/VMS host using SUN Workstations for the MMI displays.

The experience gained from these two systems are reflected in the functions offered by the newer SCOS-2000 Systems.

2.2 SCOS-2000 Technologies

SCOS-2000 was originally conceived as a Solaris system using a modest number of COTS Software products in addition to specially developed software. Terma and Vitrociset were responsible for taking this system to a common source code that runs on both Linux and Solaris. In the process some of the proprietary COTS products were replaced with opensource equivalents.

The result of this exercise has been to drive down the cost of a basic SCOS-2000 installation to a few thousand dollars, without any degradation in quality. The recurring costs are limited to the PC Hardware and one or two remaining proprietary COTS products.

The other cost benefit is that many of the development tools (C++ compilers etc) are also not only robust products due to their wide usage but available at little or no cost. Without the need for any great investment in hardware or software, potential users can have a system for very low initial costs.

3 Operational Concepts

3.1 Introduction

Until recently, all the users of systems at ESOC have not seen the need for procedure automation. The operations are conducted by means of written flight operations procedures, with the operations personnel interacting with the system for monitoring, sending of commands etc.

For some of the more repetitive operations, a few special functions have been introduced which make the life of the operator easier.

For example, when a lot of telecommands have to be uplinked to the spacecraft (as an output of the mission planning requirements or flight dynamics), the concept of autostacks has been introduced. The MPS (or FD or any other system) will output a list of commands which can be save in a defined format, known as a saved stack file. This file can then be loaded into the SCOS system and sent as an automated stack (as opposed to the manual stack for sending individual commands). This greatly reduces the operator workload, together with the reduction in the potential number of operator errors.

However, functions such as these can only be seen as easing operator workload and reducing the chance of introducing human error into the operational activities.

3.2 Automating Spacecraft Operations

The first point to make is that automating spacecraft operations can to a large extent be achieved by the actual spacecraft design. Spacecraft are tending to get more complex and a major part of this is due to the introduction of more autonomous systems on board which do not require intervention from the ground. Secondly, for many years, spacecraft have had on board queues for telecommands, so that the commands can be uplinked in batches (for example during a pass, or an operator shift), leaving the spacecraft to run itself for a period. Thirdly, some of the newer ESA missions have an on board control language where so-called On Board Control Procedures can be uplinked to the spacecraft and run autonomously from the ground.

3.3 Automating the mission

The next step in further reductions of operator workload can have three goals:

- Reducing the size of 24/7 operations teams
- Removing the need for 24/7 operations (e.g. reduce to 2 shifts, or normal office hours)
- Reduce spacecraft operations to on an as required basis

To do this requires more than just automating the spacecraft operations but also the ground segment operations. This includes control of the ground stations as well as control of the control centre itself. For Some particular types of mission this can also extend to distribution of the data generated by the mission (e.g. meteorological, Earth Observation and Science missions).

4 The Automation System

4.1 Main Requirements

Therefore we have the main high level requirements for an automation system:

- It must have a language and language environment that allows people to define and run schedules and procedures to automate the spacecraft operations (ability to send telecommands, access telemetry etc)
- It must have a high level of integration between the automation system, the spacecraft control system and ground segment to allow proper control of the SCS/GS – this covers both statements to allow the control of the SCS as well as access to data within the SCS/GS to allow monitoring of the different activities and status of the system. This latter point also relies on the other elements of the ground segment (ground stations, data centres etc) to provide whatever data is necessary to both provide such data and have an interface to allow their control.

4.2 The ASE tool

The Automated Scheduler Executor is a tool developed for automating such operations. The main characteristics of the tool are as follows:

- It uses a procedure automation language - PLUTO (Procedure Language for Use in Test and Operations) – which is an emerging standard being defined by ESA in this area.
- It allows abstraction of the space and ground system – the so called Space System Model – which, rather than giving the tool access to the full telemetry/telecommand data base (which is far too detailed), it allows access to those entities that are necessary for the automation of the system. It also allows definition of a tree structure for the space/ground system, which means that at a lower level node, all the telemetry, telecommands and automated procedures that are relevant to that node are organised together
- It is written in Java – making t platform independent

5 The ASE tool

5.1 Background

The "**Automatic Scheduling Execution**" (ASE) is a system developed by Vitrociset in answer to an ESA study for Mission Control System (MCS) procedure automation. The study addressed the need to have a degree of

automation in addition to the standard telemetry and telecommands monitoring and control functionality.

ASE is compliant to the proposed ESA standard ECSS (European Cooperation for Space Standardization) ECSS-70/32 "Procedure Language for Use in Test and Operations" (PLUTO). At a high level, PLUTO defines common specifications for procedures Mission Operations and the pre-lauch AIT/AIV operations.

PLUTO defines three key elements:

- a procedure reference language,
- the specifications of the procedure execution environment
- a Space System Model (SSM).

The SSM allow to represent mission functional and physical units into an hierarchical structure of software system elements with each element embedding the knowledge of the actual mission unit or equipment.

ASE is fully compliant to PLUTO extending what the standard defines with procedure execution scheduling capabilities. The system high-level functions include: PLUTO procedure editor, procedure execution engine, procedure execution tracing and debugging, execution scheduler, PLUTO Space System Model.

ASE currently includes interfaces to the ESA SCOS-2000 generic TM/TC kernel for telemetry, telecommand and events references.

ASE is entirely written in JAVA and includes.

5.1.1 ASE Functions and System Overview

ASE provides three major component functional areas:

- Procedure preparation environment
- Procedure scheduling and execution environment
- Schedule file reception and validation

The three functional areas are properly separated within the ASE environment - meaning that the user can prepare procedures independently from actual operations. The user has also the possibility to test procedures within the procedure preparation environment without interfering with the execution of operational procedures/schedules or with the schedule reception and validation process.

The following picture provides a high level view of the ASE packages with the dependencies among the packages.

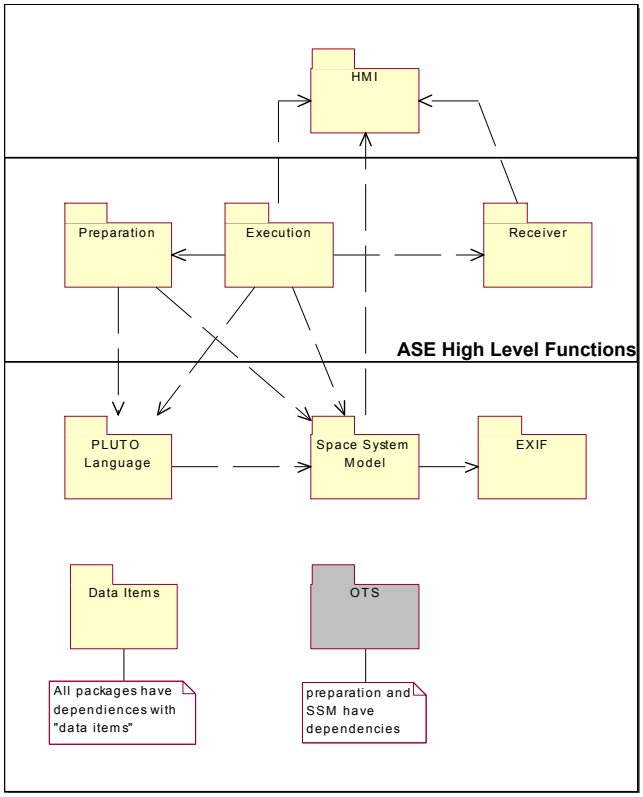

Figure 5-1 – ASE High Level Functional Packages

At very high level ASE includes:

- **Preparation**: procedure and Space System Model preparation and editing environment
- **Execution**: execution and scheduling environment
- **Receiver**: schedule file reception and scheduling environment

The three main packages make use of other packages providing lower-level services to them. The service packages are:

- **PLUTO Language**: providing the PLUTO compiler and the PLUTO execution engine
- **Human Machine Interface (HMI)**: displays, views and in general GUI components implementing the user interfaces to the different ASE functionality
- **Space System Model**: implementing the PLUTO Space System Model.
- **External Interface (EXIF)** used by Space System Model to interact with the external world
- **Data Item**: are data item models and related methods to handle them

- **OTS**: includes functionality identified to be implemented by external software packages.

The ASE execution environment includes:

- an **Agenda,** which is the higher level scheduling control and editing element where schedules (see next bullet) execute. The agenda is capable to handle and schedule an unlimited number of schedules (the actual maximum number is limited by the system resources).
- **Schedule**: software entity that can be scheduled for execution within the agenda. This the entity where a *schedule* is executed. Multiple schedules can be present in the agenda, with the possibility to handle execution of overlapping schedules (concurrent activities).
- **Task Request**: entity that can be scheduled for execution within the schedule and in charge to provide the runtime environment, where a procedure is executed. A procedure runs always within a task request.

The whole ASE scheduling environment, as defined by the user in terms of task requests (hence procedures) and interlocks among procedures (specifying when a procedure can be executed and if it has a dependencies with the execution status of other procedures), can be saved within files and then re-loaded for execution.

Time windows associated to task requests (as execution qualifiers) are handled as delta time according to a schedule **reference time**. Every time a schedule is re-loaded the user can enter such reference and, at schedule loading, task requests time windows are re-computed as absolute times according to the schedule reference time and the delta-time associated to each task request.

The PLUTO Space System Model (SSM) provides the definition of the mission and interface with the mission control system. The ASE SSM includes:

- Hierarchical decomposition of the mission into system elements
- Objects associated to each systems, that is, parameters (telemetry), activities (telecommands, procedures, system activities), events
- Interface with the mission control system

It must be noted that ASE has generalised the concept of the PLUTO SSM by including besides all mission elements (tied to the standard TM/TC data defintion) with system and application elements. As an example the ASE agenda is present in the ASE SSM as a system element. The Agenda system element can include schedules as dynamic (created as required) system elements and each schedule system element include task request as appropriate. All these elements have associated parameters.

Another basic functionality implemented by the ASE SSM is the interface to the mission control system. ASE is highly integrated with SCOS-2000 not only for telemetry and telecommand references but also for the following services:

- Event injection
- MISC Dynamic handling
- Roles and Privileges

The following figure summarises the system functional modules and their internal and external interfaces.

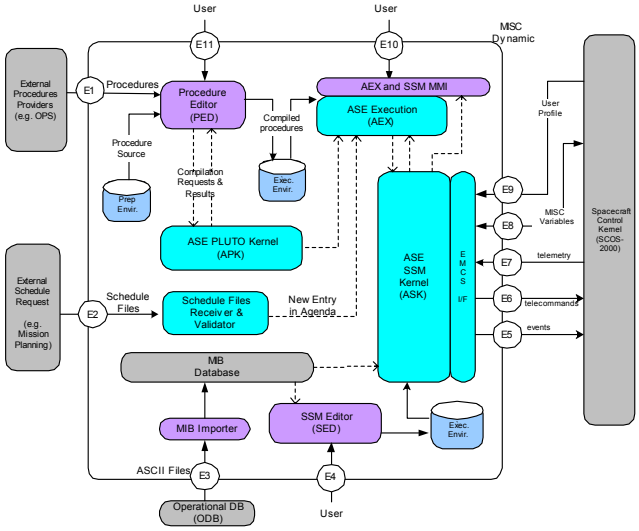

Figure 5-2 – ASE High Level Functional Modules

The data transferred across the external interfaces is summarised in the table below.

ID	Data	Description	Source	Destination
E1	Procedure	PLUTO Procedures imported into ASE	OPS	cfm
E2	Schedule File	XML File	Activities	receiver
E3	Operational Database	ASCII Files generated from the MS Access operational database	DBS	db
E4	N/A	User activities for editing and	User	SED
E5	Events	Events Injection	SSM	SCK
E6	Telecommand	Telecommand Injection	SSM	SCK
E7	Telemetry	Telemetry parameters values	SCK	ssm

ID	Data	Description	Source	Destination
E8	MISC Dynamic	MISC variable value and change value notification	SCK	ssm
E8	MISC Dynamic	MISC variable value setting	SSM	SCK
E9	User Roles and Privileges	Roles and privileges definition and authorization for ASE Login validation	SCK	Ssm
E10	N/A	User interaction with the ASE MMI	User	N/A
E11	N/A	User interaction for procedure creation and editing	User	PED

5.1.2 ASE Schedule Files

An ASE functionality relevant to the ATIS study is the format and reception of schedule files.

Schedule files are files defining the ASE scheduling environment in terms of :

- Schedule execution attributes (start time, finish time,…)
- Task requests definition (procedure name, procedure input parameters, start time, finish time,…)
- Execution constraints among tasks (follower, interlock type, interlock subtype)

Schedules are created within the ASE procedure scheduling environment, by editing task requests and associating execution parameters to each task request. They can be saved via the '***Save Schedule***' option available in the ASE schedule file menu and they can be loaded via the '***Load Schedule***' option available in the same File menu.

A schedule can also be made available to ASE by injecting the related file into a directory polled by the ASE receiver. The ASE user is notified of a new schedule detected by the ASE receiver and has the possibility (after validation) to schedule it for execution into the agenda.

Schedule files are formatted using the E**X**tensible **M**arkup **L**anguage (XML) data representation specifications.

5.1.3 ASE / SCOS-2000 Interfaces and Operational Scenario

SCOS-2000 includes a set of **external services** (**EXIF**) for data provision and injection. These services have been implemented as CORBA servers which can be referenced by client applications such as ASE, for data processing and system configuration at client side. Services are:

- **Data provision**: Telemetry, telecommand (telecommand history), events
- **Data injection**: Telemetry, telecommand, events

Mission data provision/injection and system services are SCOS-2000 external services used by ASE for interacting with SCOS-2000.

Currently ASE uses a subset of such services, that is:

- Data provision: Telemetry
- Data injection: telecommand and events

In addition to these services, ASE interacts with SCOS-2000 for other services access available as standard SCOS-2000 kernel interfaces. These are

- **Configuration**: access to SCOS-2000 MISC dynamic variable (to be noted that ASE reads also MISC static variable). ASE receives and sends changes to SCOS-2000 dynamics system configuration variables. This interface is supported on the Spacecraft Control Kernel by the existing MISCdynamic server. ASE is adapted to restrict the configuration variables that may be changed by ASE to a subset, to reduce the risk of system variables being inadvertently modified by a faulty procedure.
- **Roles and Privileges:** for user login and privileges control. ASE obtains user log-in information, and associated roles and privileges, in order to check the privileges for restricted activities such as sending TC directives. (ASE can be operated without commanding privileges for the purposes of procedure testing, in which case TC directives are not sent to the Spacecraft Control Kernel for uplink). In case SCOS-2000 server is not available ASE performs user roles and privileges checking through a local configuration.

The following picture report the overall set of interfaces available (EXIF and kernel) with the indication of those used by the current version of ASE.

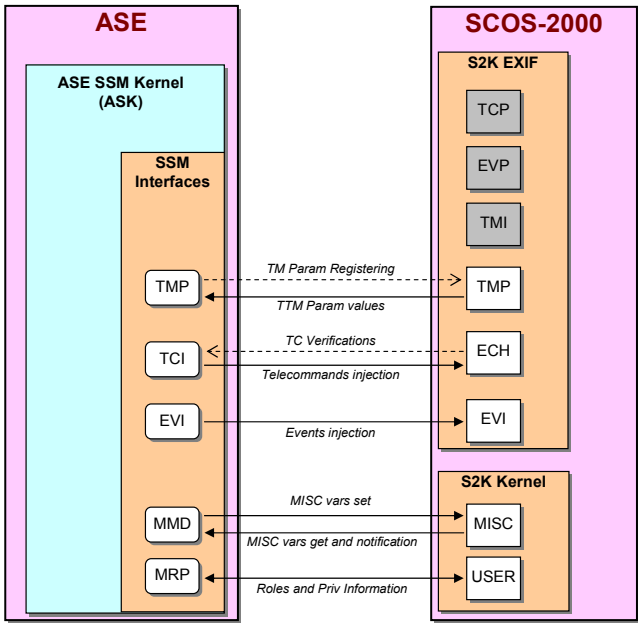

Figure 5-3 – SSM External Interfaces

6 Missions

It is currently baselined for 2 missions:

- The ESA Rosetta mission is a deep space mission to rendezvous and drop a lander on a comet. Launch is set for early 2004, with comet rendezvous in 2001. Rosetta already has a lot of on board autonomy – for example the use of on board control procedures. It has periods when it will be out of ground contact for many months (it passes behind the sun). When visible, ground contact will be limited to a few hours per day, and the majority of operations during these periods will be automated.
- The second mission is Radarsat-2. This is a commercial Earth Observation Mission. It is a LEO satellite which has passes of a few minutes 14 times a day. During the passes it is necessary to uplink many telecommands for the spacecraft operations (up to 48 hours ahead). The mission will provide SAR Images to customers. The customer requirements are put into a mission planning system, which generates an appropriate list of telecommands, which is then passed to ASE/SCOS-2000 for automatic uplinking during a pass.

7 Conclusions

The combination of systems and technologies demonstrates the following:

- SCOS-2000 is an open spacecraft control infrastructure that will allow efficient interfacing to other operational products
- The technology – Linux, Gnu compilers, Java etc provide a very low cost and robust baseline for the system, allowing the exploitation of cheap PC Hardware
- The ASE tool will allow much more cost effective operations through the high degree of automation it can bring to the system

---

End Notes:

[i] Roger Patrick – rmp@terma.com

[ii] Francesco Croce - Francesco.croce@vitrociset.it