Mission Operations Preparation Environment: A new approach for the future

W.Heinen[1] , S.Reid[2], S.Pearson[3]
Rhea System S.A. Avenue Pasteur 23, 1300 Wavre, Belgium

This paper will present concepts and preliminary results of a major study (MOIS&MORE) currently being conducted for the European Space Agency, to prepare a fully integrated environment for mission Operations Preparation. The Operations Preparation Phase is one of the most challenging in the mission lifecycle. The spacecraft itself is supported by complex systems for on-board control, mission planning, monitoring and control, flight dynamics and communication, all of which contribute to the continued ability to fulfil the mission. The overall picture is of a "system of systems", each very complex in its own right but only effective when working correctly in concert with the others.

The primary focus of the MOIS & MORE project is the safe and efficient management of this system of systems, through the coordinated management of all configuration data used for Satellite Operations. This will be achieved conceptually by the definition of a hierarchical offline data model designed to capture all classes of configuration data in a consistent way, providing the means to define relationships and dependencies and providing safe storage and release control. This data model is based on the principles of the Space System Model (SSM) defined in ECSS standard ECSS-E-ST-70-31C.

I. Introduction

THE MOIS operations preparation toolkit is well established throughout the European Space Industry and is the standard tool at ESA/ESOC for the preparation and maintenance of Flight Operations Plan(s), Automated Procedures, Command Sequences. It is used for the entirety of its ongoing missions.

The MOIS Toolset

RHEA System's Manufacturing and Operations Information System (MOIS) is an integrated set of software tools for spacecraft mission preparation and execution. MOIS is designed to provide a practical and flexible method for handling all aspects of a mission. A key feature of MOIS is its independence from individual mission control infrastructures. The resulting flexibility in adapting to any operating environment has led to MOIS becoming the pre-eminent procedure development, execution and automation platform for spacecraft missions in Europe – it is used as standard by the European Space Agency. The main tools that constitute MOIS are the following:

- ***Writer*** enables the creation and development of procedures or timelines that may be viewed and worked on both in linearised form, showing steps and associated statements, and as a flow chart graphical display of the step structure (via Flowcharter).
- ***Flowcharter*** enables the creation, editing and display of procedure structures through a graphical flow chart.
- ***DB Editor*** controls the creation, import or editing of the MOIS operational SCOS spacecraft database.
- ***Function Editor*** provides a user-friendly interface to create and maintain functions and directives for use in procedure writing.

[1] MOIS Product Manager, w.heinen@rheagroup.com
[2] Chief Technical Officer, s.reid@rheagroup.com
[3] System Engineer, s.pearson@rheagroup.com

- ***Validator*** controls validation testing of a procedure and stores the results. A Test Harness can optionally be used to emulate a control system/simulator when using with Validator to test a procedure.
- ***Scheduler*** enables mission planning for satellite operations and station scheduling.
- ***Publisher*** allows producing mission documents that do not fall into the procedure category. It also takes the hard copy production of procedures and timelines as painless as possible by automating the printing of up to thousands of documents under configuration control.
- ***Reporter*** logs the definition, analysis and solution of problems as they arise, and also performs consistency checking and analysis of sets of procedures.
- ***Library*** provides a fully-integrated configuration management for mission data, including procedures, schedules, operational databases and documents, across the full suite of MOIS tools.

The General Model for Mission Preparation Products

The user view of Mission Operations Systems is very often through the prism of available applications and known data structures which are usually managed independently. This paper will show how the lack of a single management system for all mission configuration data and the limitations imposed by the available views of these data can be addressed.

The principles which are presented here derive from the ECSS E31 Space System Model [ref[1]]. This SSM defines an extensible hierarchical view of the space system where elements such as telecommands and telemetry parameters are associated with nodes (System Elements or SEs) representing components of the system such as an on-board instrument. Its objectives are to describe the system more meaningfully, to provide a means for common data transfer between systems and to permit isolation, duplication and replacement of individual branches. This idea is particularly useful in AIT where the data model can be assembled in parallel with the spacecraft. However, it has several benefits for Operations as will be demonstrated hereafter.

- The E31 definition will be extended to cover all mission configuration data, not just the space system; and
- the spacecraft database will be subdivided into re-usable 'elements'. The model will keep track of these elements and their evolution.

The intention is ***not*** to re-format or re-package any data. The resulting model is a view of the system which when created and populated will provide a clearer view of the mission data and enable better management of it.

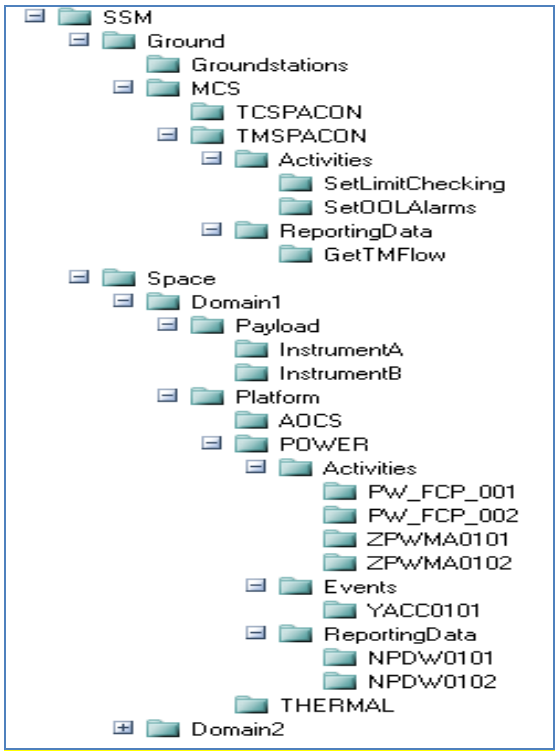

Figure 1. classification of elements within the SSM

II. Revisit the Data Model

A. The Spacecraft Database

The SCOS Spacecraft Database will be taken as a starting point for the description of the revisited data model. It is 'monolithic' in the sense that it configures SCOS for the entire space segment. Databases for other control systems in Europe, such as CGS and SIS for OpenCenter, are mostly similar in this respect. Sometimes the ground segment configuration is included or specified in a separate database.

A SCOS spacecraft database is derived from the MIB ICD which specifies a set of files which implicitly map to a set of relationally interrelated tables. These data serve 2 purposes: to configure SCOS and to describe the space segment in terms of telecommands, telemetry, their packet containers and associated displays. This relational scheme minimises duplication of data (shared digital calibration mappings for example) but by definition can have relational dependencies which make it impossible to guarantee a separation between physically separate parts of the system – different on-board instruments for example could share a calibration curve.

The ECSS-E_ST-70-31C Space System Model (SSM) takes a different approach. It envisages a hierarchical breakdown of the space system with telecommands and procedures (both seen as Activities), telemetry (now Reporting Data) and Events such as packet arrival and anomaly signalling, all located in the System Element (SE) nodes of an SSM hierarchy reflecting the physical breakdown of the space system (see Figure 1).

This is not just a way of managing complexity; it also facilitates a breakdown of the system that can lead to compartmentalisation and therefore less validation and testing.

To realise the potential of an SSM a mapping of SCOS elements (Telecommands, Telemetry and their associated calibrations, packets and monitoring data) must be made to the SE nodes where they belong in a carefully designed mission SSM hierarchy. This hierarchy is then navigated whenever the data are accessed - from the spacecraft database editor to the procedure editors to the execution environment.

Both procedure and spacecraft database development will therefore be based on an SSM governed by a hierarchical configuration-controlled structure. Procedures will be located in SE nodes of the tree and will be checked out from these locations in the usual way. When performing spacecraft database updates it will be possible (implicitly at least) to check out MIB elements in any branch of the tree, from individual TC or TMs, to whole subsystems to the entire SSM.

This view facilitates user access control at branch level, not just for the procedures but also for the elements within the spacecraft database. If it were designed so that edits to one branch could be guaranteed not to affect the rest of the model, management of the database could be distributed and there would be no need for all database edits to go through one central authority.

B. Integration of the MIB Editor

SCOS MIB data will not be converted to the detailed E31 form since such a two-way conversion between MIB and E31 formats may not be possible (at least this has not been demonstrated) and it would serve no real purpose. The main SCOS elements must, however, be mapped to locations in the SSM to achieve this system view.

The SCOS spacecraft database editor will need to be aware of this mapping. Elements to be edited should first be identified by their SE location which is then reserved for editing. Since elements such as TCs and TMs have a single namespace in the MIB (they are table Primary Keys) they could also be searched for by name in the usual way to find their allocated SE.

The database editor would reserve (check out) from the SSM all element references which are related to the current edit in the spacecraft database and are therefore affected by it. This information is obviously available via SQL query, but the relationships could be complicated. It will depend on how tightly coupled the MIB data are; for example how many elements share the same calibration data or telecommand parameter definitions.

The MIB defines 3 broad table categories: Monitoring, Displays and Commanding. There are many table dependencies such as Packets on both Monitoring and Commanding (separately), Monitoring and Commanding on calibration (separately) and Displays on Monitoring.

The Issue of Data Duplication

The MIB relational scheme allows calibration data to be shared between two TCs or two TMs (but not between a TC and a TM). TC parameter data can be shared as can limits and range data but in practice this is not that common. Analogue calibration data are rarely shared and often only simple digital data are shared such as e.g. 1=ON, 0=Off. It may be possible to achieve complete decoupling of these element types without much data duplication. It can be noted that this is the E31 XML schema view (hierarchical and not relational).

It should be possible to place Monitoring (TM) and Commanding (TC) elements at the SE leaf nodes, and Packets and Displays at different levels in the SSM depending on their dependencies (in general they should be placed at the lowest level branch that contains all their dependencies).

If, say, a calibration curve or an OOL check needs an update, it would be necessary first to locate the TM or TC that uses it. The editor would then have to check out all the TM elements in the SSM nodes that use the same calibration curve. Thus the editor, while presenting a hierarchical SSM framework would need to be aware of all the relational interdependencies of the MIB database.

Once the SSM hierarchal structure has been designed for the mission a drag and drop editor will facilitate the location of the MIB elements (TCs, TMs, Packets and Displays) within it. These references will then maintain their own history like any other configuration-controlled item.

The database editing experience may not be significantly different to before. The same MIB fields will be available for editing but on top of this the mapping of TM, TC Packets and Displays to the SSM will be visible and navigable. Importantly, no elements will be editable unless they have been mapped to the SSM. This is a fundamental requirement of the model. All the procedure editors will reference elements via their SSM references.

At the end of the editing session the spacecraft database will be checked in. At the same time references within the SE nodes to all updated elements, together with all other elements affected relationally by these edits, will also be checked into the SSM configuration-controlled structure. This will provide a clear record of which branches and SE nodes of the SSM have been affected by each database update and hence what may eventually need to be re-tested.

Finer Grained Spacecraft Database Elements

Telecommands, Telemetry and Event Packets can be mapped to E32 Activities, Reporting Data and Events located at the SE nodes. However, it may well prove that the spacecraft database is too relationally coupled to be split up into these separate elements without an unacceptable amount of duplication. It is not just calibration data that can be shared between elements. For example: in the MIB it is possible to share parameter definitions between TCs. An example is the Group Repeater parameter which is defined once and shared between all TCs with a group repeater – which of course is a perfectly valid thing to do in a relational scheme. It is also possible to share parameter limits and ranges between parameters.

A finer grained division of the spacecraft database data is shown in Figure 2. In addition to the main elements we have the *Parameter* element (TC or TM) and the *Calibration* and *Limits/Ranges* elements (which are both attributes of Parameter).

Dividing up the spacecraft data like this should avoid most if not all data duplication. It would then be possible to change a TC or TM definition without affecting all its dependent and related data and to update a calibration curve without affecting all the TCs that use it.

The *Parameter*, *Calibration* and *Limits* elements in the SSM hierarchy are not the same as the TC and TM elements: they never need to be referenced directly and do not map to physical parts of the system. It may therefore be possible to locate them automatically and promote them up the tree when they need to be shared more widely.

Such a categorisation would greatly help the management of constellation databases.

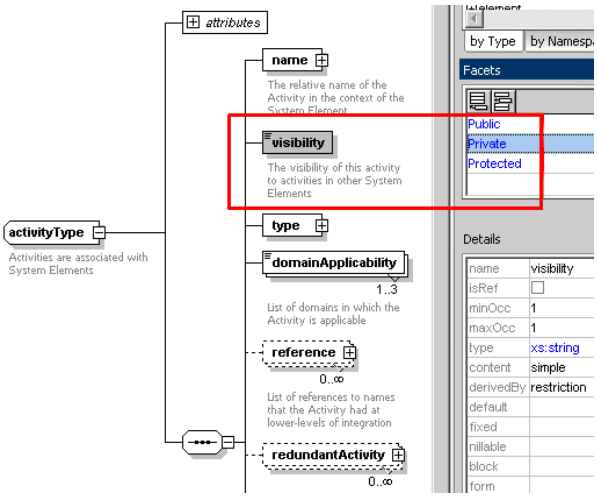

Figure 3. E31 Schema Branch with Activity Visibility Attribute.

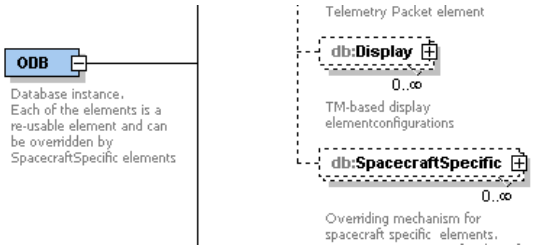

Figure 2. Finer Grained Spacecraft Database Elements.

C. Integration of the Procedure Editor

Procedure editing will follow a similar but less complicated path. Procedures must be located in the SE nodes of the SSM hierarchy, and Activities (TCs or procedures), Reporting data (TM) and Events (e.g. synchronising on packet arrival or signalled anomalies) will be referenced within procedures via this structure.

It will then be possible to generate reports detailing which parts of the SSM have changed since a specified baseline was established.

This is a fundamentally different view since the spacecraft database and procedures are now a combined data set. The spacecraft database does not have to be viewed as a huge interrelated single item anymore and the system data it contains can be modelled and displayed more clearly and logically. The hierarchical view is kept separate from the spacecraft data via an element mapping. This mapping could equally apply to other database formats: an OpenCenter SIS database and a SCOS database used for the same mission could use the same SSM structure. The BepiColombo mission, for example, will use both SIS and SCOS databases. Although the data are equivalent (a SIS to SCOS converter is available) the elements (TCs for example) will be named differently due to SCOS naming limitations and ESOC and Astrium conventions). It may then be useful to have a common system model as a reference.

D. Adding Scope to the SSM

The SSM as defined in E31 has no scope limitations. Despite its hierarchical structure all its data are accessible everywhere. In particular, all TCs and TM parameters are available to all procedures and a procedure can be called from any other. This lack of information hiding in different contexts can lead to a lack of clarity and to greater testing needs.

There is a direct parallel in the software domain. Encapsulation and information hiding are basic tenets of object oriented programming. The principle is that objects do not need to expose all their inner workings so only a subset of their methods are made public. Having a public interface also means that inner mechanisms can be altered without affecting the rest of the system.

The CPE study [ref[2]] suggested a simple update to the E31 SSM to achieve this: the addition of a scope (or visibility) attribute which can take the following values (following Java and other languages):

- Public – can be accessed any System Element (default);
- Private – can only be accessed from the same System Element; or
- Protected – can only be accessed from activities in the same System Element or its children (recursively).

Such a mechanism could provide information hiding to reduce complexity and simplify integration testing, in line with standard object design principles. Figure 3 is an amended branch of the E31 schema with this attribute added to an Activity (which can be a TC or a procedure)

All references in the SSM could be public initially (as at present with the flat-structured MIB). Once it becomes natural to access all database elements and procedures in the context of an SE in the SSM hierarchy, it may be considered operationally useful to reduce the scope and hence visibility of certain elements to procedures written for other SEs, both to reduce unnecessary complexity and to ensure that correct processes are always followed.

For example, if an instrument must be switched on in a certain way with specific checks performed during the process, it may be comforting to know that other procedures are not able to attempt to perform this procedure in another way. TCs that perform this state transition (and therefore located in the local SE) could then be hidden from procedures written for another SE. Only the procedure (Activity) that performs the switch-on correctly would be exposed publically.

Another example: perhaps only certain state transitions for a particular instrument are legal. Its SE could be designed to expose only the procedures that perform valid mode changes between valid states – preventing say a switch from *dump* to *inactive* without going via the *active* state.

III. Expanding the SSM Definition

ECSS-E_ST-70-31C [RD5] is a delivery format for exchanging a data model of the space system including TCs, TM parameters and procedures (but not display data). The hierarchical model should facilitate the generation of an E31 electronic database (so far by including SSM-referenced data from the MIB and from procedures).

We are now proposing to extend this model to a repository structure for all types of mission data. It should be able to hold any mission data that could benefit from being located in an extensible hierarchical structure constructed specifically for the mission and subject to configuration control from high-level branches down to the level of individual files, procedures and MIB element references.

As a structure it should ***not*** be limited to the space segment. The following would constitute the data that could be located in corresponding SEs of a ground segment model within it.

1) The **Spacecraft Database** elements
2) The **Ground configuration data**
3) The **Pluto Procedures**
4) The **Flight Control Procedures** produced by MOIS Writer. Note that Writer also produces a simplified Pluto output.
5) The **System Documents** that belong to the Flight Operations Plan that are written with Publisher.
6) The **Mission Planning Rules**
7) **On-board Control Procedures** produced by Writer.

These data items are expressed as SSM Activities, they would naturally be stored there providing the possibility to write ground station procedures using the same MOIS editors as for space segment procedures. This is an important benefit of the E31 SSM's *Activity* abstraction. Once everything is an *Activity* it can be called from a procedure (which itself is an *Activity*) in exactly the same way. Commanding the ground station can be done via procedures in the same way as commanding the spacecraft.

IV. Implementation Details

The objective is to redesign MOIS to create a flexible, open environment in which all types of configuration data can be managed. It will continue to provide the necessary tools for preparation for all types of procedure, but additional editors can be integrated and added. New editors for TM/TC database and Mission Planning rules will be developed within this framework and at the same the existing toolset remains an integral part of the next MOIS product issue.

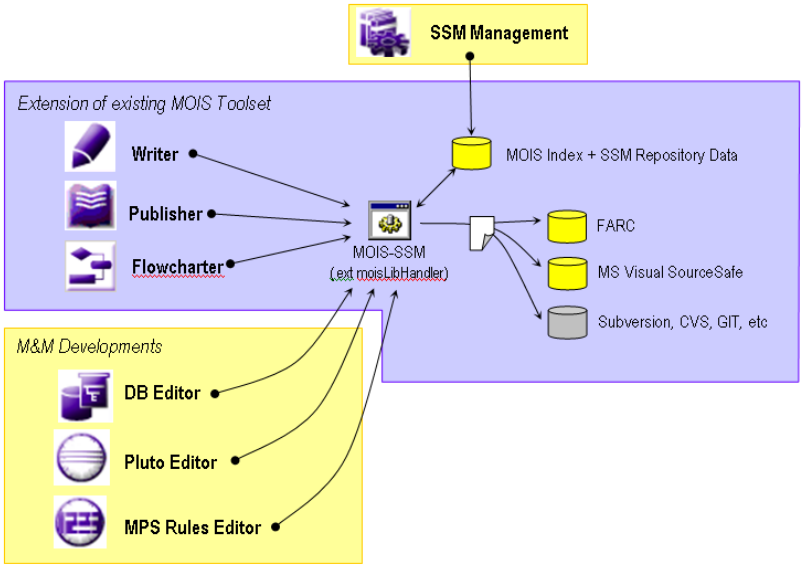

Figure 4. The diagram shows the breakdown of existing and new components and how their interaction can be envisaged.

A. The Spacecraft Database Editor (DB Editor)

The model can be introduced without necessarily having to rewrite the database editor. A data export has to be done from the database tables into an element specific dataset. For SCOS, one can adapt the "dat/text file export & import" with the difference that not entire tables are handled but just element specific data. For example a calibration curve with 10 points will form a dataset of one CAF.dat file with one entry and a CAP.dat file with 10 entries. It is then a matter of importing a set of dat files that are classified somewhere in the branch of the hierarchy to form the database for an editing session of that branch.

The configuration control is ensured by using that hierarchy as the repository structure and merges & compares can be done very easily, outside of the specific DB Editor. The end or checkin of an editing session will regenerate the dat files and deltas will be checked in. Whether file reservation is necessary or not is questionable and there are diverging opinions on it.

Reusing a database editor as-is represents a minimal solution. The study however intends to provide an enhanced DB Editor which takes full advantage of the model – for the system element selection, filtering, checkin etc… ESOC have recently developed DABYS as a framework for the management of databases. This provides an environment loosely equivalent to the MS-Access tool which allows development of specific database applications. It stores data in a MySQL RDBMS and provides rudimentary configuration control. The GAIA mission implementation is the candidate for this study.

B. Ground configuration data

Ground Configuration data can be directives for the MCS (NIS) and other configuration parameter settings (for example MISC variables). The management of this configuration data is subject of the IDEA study – see [ref[6]].

C. Pluto Procedure Editor (Pluto Editor)

An Editor that is compliant with the E32 procedure model and based on the established model-view design. Users can create procedures in MOIS Writer as normal but automated procedures may also be viewed in the PLUTO text editor [ref[2]].

D. Flight Control Procedure Editors (Writer, Flowcharter)

The existing MOIS Writer & Flowcharter editors can remain, but the new model offers a new way for accessing and filtering data for the design of the procedure. Similar to the Pluto Editor, the registration into the SSM determines the filter for the available procedure constructs.

E. System Document Editor (Publisher)

Lightweight MSWord procedures & documents can be included in the FOP in the same way that Writer procedures are.

F. Mission Planning Rules (Rules Editor)

Within the MOIS & MORE study a language specific editor will be developed, initially based on the LMP (Language for Mission Planning) rule language. In a similar way to to the Pluto Procedure editor, a language sensitive editor will be developed within this study, using the LMP language specification and accessing the SSM for element insertion and rule checking.

G. On-Board Control Procedures (Writer, Flowcharter)

The different nature of OBCPs compared to ground FCPs make it necessary to extend the MOIS procedure preparation tools (Writer & Flowcharter) in a few areas. These are mainly additional statements (Set Telemetry, Send Event, Wait Event), the support for OBCP parameters and the support for subprograms (sub-procedures and functions).

Apart from these adaptations, the procedure preparation process is the same for FCPs and OBCPs. In particular, the connection to the spacecraft database ensures overall consistency and hence the editor can profit from the same additions as for the Flight Control Procedure Editors. The MOIS & MORE study provides a complete OBCP solution (chosen target missions are GAIA and Bepi-Colombo) – see [ref[5]]). These types of procedure remain equivalent and are managed the same way within the SSM.

H. Managing the SSM (SSM Editor)

Finally, the MOIS & MORE outputs will provide the tools needed to build and maintain the SSM.

References

[1]Space engineering, Ground systems & operations – Monitoring and control data definition E31: ECSS-E-ST-70-31C, 31 July 2008

[2]Space engineering Test and Operations Procedure Language E32: ECSS-E-ST-70-32C, 31 July 2008

[3]CPE Study TN3. ECSS extensions and language conversion RHEA.CS1097.DOC.03, 2010-08-04

[4]W. Heinen, S. Reid , S. Varadarajulu , "Automation through On-Board Control Procedures: Operational Concepts and Tools", Spaceops 2010

[5]A.Schwab, W.zur Borg, "OBCPs - an integrated part of BepiColombo Autonomy and Operations Flexibility", Spaceops 2012

[6] S.Pearson , F Trifin , S.Reid , W.Heinen, "An Integrated Development and Validation Environment for Operations Automation", Spaceops 2012