Functional Verification manageR (FaVouR)
A new generation of editors for the EGS-CC

Wolfgang Heinen (RHEA Group), Steve Pearson (RHEA Group), Francesco Sgaramella (ESA)

The FAVOUR study, conducted by RHEA for ESTEC, was targeted at the preparation and execution of spacecraft checkout campaigns and mission operations for EGS-CC-based systems. It resulted in a Mission Model Editor (MME) for editing EGS-CC tailoring data, and a Procedure Management Environment (PME) with test session management for procedure development and execution.

The presentation provides an overview of these tools and their application to an EGS-CC-based mission. It also describes the experience and the lessons learned using them.

MME

The Mission Model Editor (MME) is an editor for EGS-CC Conceptual Data Model (CDM) mission tailoring data. It extends the OPEN framework developed by ESA.

MME Context

The EGS-CC CDM defines a hierarchy of Monitoring and Control (M&C) Elements which themselves contain Aspects such as Parameters, Activities and Events. This part drives the EGS-CC core components. The CDM also defines implementations for the EGS-CC adaptation components. These include packets, procedures and displays. Finally, it defines the mappings between the M&C Elements and Aspects, and their implementations.

The OPEN preparation environment provides a framework for the preparation of EGS-CC CDM-based tailoring data. It includes model-agnostic functionality such as version control, workflow management and data import and export. It has an automatically generated form-based data model object editor, an M&C Element tree navigator and a Configuration Item (CI) explorer.

OPEN exposes the CDM structure and its mechanisms directly so a complete knowledge of this software level model is needed to understand it. The main purpose of the MME is to hide this complexity from the AIT or Operations user.

MME Concepts

The MME adds a **presentation layer** on top of the CDM. This layer defines the items that one would expect to manipulate, such as packets, parameters and procedures, and locates them in a Mission Model tree. This helps the MME portray the CDM data in a simple, useful and intuitive way. The Mission Model presentation layer is initially created from the CDM-based data. It can then store additional information such as the location of packets and displays in the Mission Model tree, even though these are not located in the CDM M&C Element tree. This Mission Model extension is not exported to the EGS-CC run-time.

The MME is CDM-specific. It hides the management of the M&C Elements, their definitions and contents, the mappers, the mapped Implementations and the file-backed CIs that contain them. It imposes a policy of one M&C Element per CI in order to facilitate the modular management of sub-trees. If the MME is used exclusively then any M&C Element sub tree will consist of an independent (and thus replaceable) set of CIs.

To illustrate the complexity that the MME manages to hide, a simple drag & drop of a 3x3 matrix telemetry parameter into a telemetry packet results in the MME executing around 96 separate operations on the CDM-based data. In a generic editor such as OPEN, all these operations would have to be done manually - an unmanageable task. Similarly, for the copy & paste of one telecommand, the MME copies the Activity and its Arguments, creates and maps a corresponding Activity in the M&C Element's Definition, copies the TC packet and finally creates and links a new set of mappers.

The user can switch from the MME perspective to the OPEN "expert mode" perspective to see what the MME has done 'under the hood'. An Undo/Redo view lists the individual operations and, since each MME operation is transactional, it can be reversed in its entirety.

MME PUS Modelling

The MME has an extended Packet Utilization Standard (PUS) model to support the generation of CDM packets and data from PUS tailored definitions. PUS services and subtypes are first defined and parameterised for a given mission. These PUS reports and requests are then assigned to (typically on-board) Application Processes (APs). They can then be instantiated as TM and TC packets for EGS-CC processing. Packet containers and parameters are mapped back to their originating PUS services and consistency checked against future changes.

This facility provides the means to get from PUS tailoring, in terms of the PUS-A and PUS-C standards, to CDM-based data understood by the EGS-CC.

The MME also supports an import in PUSGEN/OPUS format, which defines a PUS service tailoring.

PME

The Procedure Management Environment (PME) is used to prepare both satellite test & operations procedures and to automate the satellite testing and satellite operations.

PME Context

The PME has been developed for an EGS-CC-based system. Nevertheless, the design is such that the solution is control system independent with adapters for the different control systems. The EGS-CC adapter has been fully implemented within the FAVOUR study.

The PME provides different views of procedure data including a graphical overview and a script representation. Verification and validation within a test management system is performed by associating test requirements with test case implementations and then executing them. Test cases are mapped to executable procedures and/or to step(s) within the procedure. For steps, the execution language and the associated execution engine must support this mapping, or at least it must be possible to infer it from the logged execution results (perhaps using a script-based convention).

PME Concepts

Minimal procedure model – The EGS-CC manages predefined Activities such as telecommands (on board) or automation procedures (on ground). Activities can be sequenced as parameterised Activity Lists (similar to SCOS Command Sequences). The automation procedure exchange format for EGS-CC is the Java-based EGS-CC Automation Procedure Language (EAPL), executed by the EGS-CC Core Automation component.

Many procedure formats exist and we expect this situation to continue into the future. The PME does not try to impose yet another execution language. Instead, it has a minimal model defining formal steps, branching and requirement mappings, sufficient for a flowchart representation and step level validation. This minimal model is then implemented in, or at least linked to, a textual representation.

Requirements Testing & Verification – Steps have a single entry and a single exit with optional expression-based pre-conditions and post-conditions for validation. Sub-steps are supported. Standard procedural branching (If, for, while, for-each) is performed within a step with each resulting branch pointing to a new step.

Test requirements are mapped to steps (in general) via test cases. Steps with mapped requirements must implement a post-condition, the result of which is recorded during a test session and correlated with the requirement to produce the Verification Control Document (VCD). System logs from the EGS-CC run-time are merged with these results to form a complete record of the tests in a test session.

PME Client-Server Architecture – The PME system follows a client server architecture encapsulating all EGS-CC and CDM complexity on the server side and just exposing simple services to clients for the data access and for the execution. This leaves the implementation of the client simple and independent of the EGS-CC. It also facilitates both a web-based and rich-client UI solution.

The PME procedure editor implements such a client. The procedure model, based on the minimal procedure model, is kept simple and is represented in both a readable text format and as a flowchart. The procedure grammar is based directly on this model and is implemented in a DSL (Domain Specific Language) using Xtext. It supports many standard features such as drag & drop of M&C elements obtained from the PME server (Telemetry, Telemetry & Telecommand Packets, etc…), code assist (verifying the grammar), content assist (such as value enumerations), syntax highlighting, validation and quick fixes (where possible).

PME procedures can be exported in other forms including the EGS-CC Java-based EAPL procedure format. EAPL procedures are embedded in the CDM data so every procedure update unfortunately involves a lengthy process of installing a new CDM-based System Operations Baseline (SOB) in the EGS-CC.

PME Automation – The procedure model is the basis for the PME DSL grammar. The automation procedures of the PME are consistency checked against the Mission Model but unlike EAPL are not included in the exported CDM-based SOB – avoiding the lengthy release process.

PME scripts are automatically translated to executable Groovy and execution is delegated to a Groovy engine with a trace back to the source PME DSL. Procedures can be stepped through in their native simple and readable script format. They can even be edited while being debugged; the edit just has to be saved before continuing with the execution. Stepwise execution & debugging, inspecting local variables, and setting breakpoints directly in the DSL are possible.

Procedures can also be autonomously executed on the PME server without the PME UI. A web interface (and a REST API) is available for controlling and monitoring this remote execution.

For the FAVOUR study, the PME Automation System has been validated end-to-end with the EGS-CC & Simulator.

The presentation provides an overview of the MME and the PME architecture, its concepts, capabilities and technology choices. And it describes how they are used with an EGS-CC-based system.