FAVOUR

A new generation of editors for the EGS-CC

TEC-ED & TEC-SW Final Presentation Days 2021, W. Heinen, S. Pearson

ESTEC TO: F. Sgaramella (TEC-SWT)

Introduction

EGS-CC is the Core for New Ground Systems

- Collaboration of ESA and ESA Member States
- European Open Source Licence (Available to ESA member states)
- Provides the core elements of a test and/or mission control system
- Tailoring Data defined in a Conceptual Data Model (CDM)
  - UML-defined data class model
  - Inheritable classes with data fields, linked by containments or references
  - Has a hierarchical tree – the Monitoring and Control Model (MCM) tree
  - Custom model extensions possible

Introduction

The Complexity of the CDM

Not easily understood or manipulated in its raw form

- Logical items such as TM, packets and calibrations are spread across classes
- Mappings between the MCM tree and implementations such as packets have to be created and maintained
- Duplicate data fields must be kept synchronised
- Monitor & Control Element (MCE) Definitions have to be created and kept synchronised with MCEs
- Procedures, scripts and expressions are just strings and need editors
- Automation procedures are written in Java Code (EAPL)
- Configuration Items (files persisting model data) must be strictly managed for the MCM tree to be modular

A tool is needed to manage this Mission Model in a logical, meaningful and consistent way

Introduction

FAVOUR Study Goals

1. Mission Model Editor (MME)

Allows users who are ***not EGS-CC experts*** to create Mission Model to tailor the EGS-CC CDM for their applications

2. Procedure Management Environment & Test Management System (PME)

Allow users to prepare satellite tests and operations easily in an intuitive graphical way and at a ***higher level than programming*** (…)

The Test Management System(…) is a system to manage ***Functional Verification*** (FV) and spacecraft testing with an EGS-CC based test facility

ESTEC, Technical Officer: F. Sgaramella

1. MME
Mission
Model
Editor

- Overview
- Architecture
- Views
- Forms
- Features

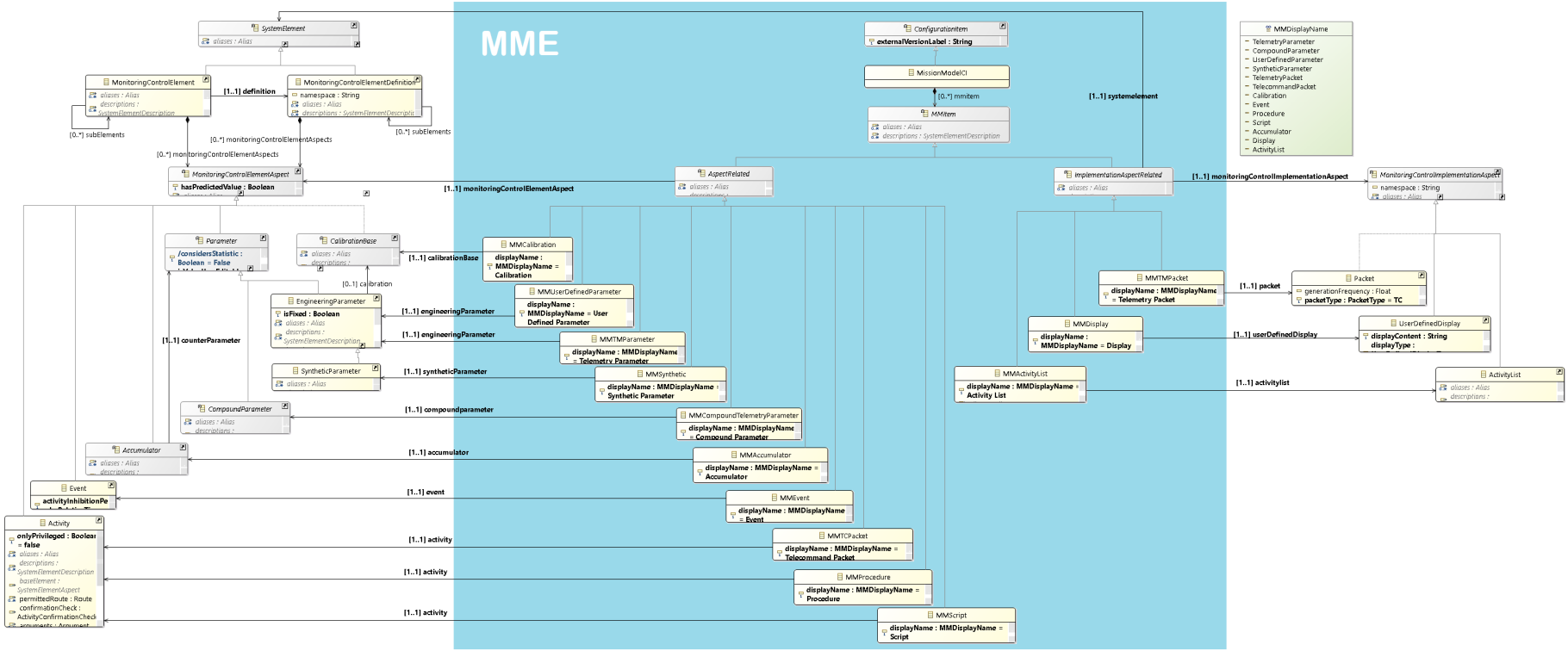
MME Overview

Features

- Provide an editing view of EGS-CC CDM data in which all internal ‘software’ mechanisms and structures are hidden
  Presentation Layer -> using well known data constructs (e.g. TM Parameters, Calibrations, Packets ...)
- Configuration Item split to facilitate modularisation of the main MCM tree
  Implicit CI management
- Support import and export of sub-trees
- Prevent creating inconsistent data
- Provide clear error reporting & navigation
- Track all updates clearly and unambiguously

MME Architecture

Mission Model Management Layer



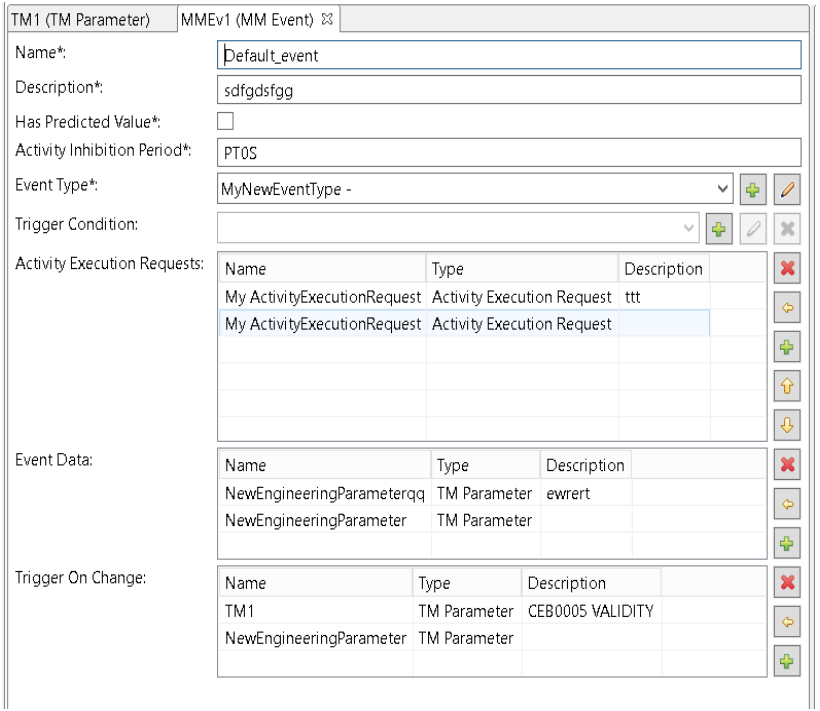
MME Architecture

Common Widget Library

- Set of widgets to standardise UI behaviour
  - Text
  - Combo
  - Combo Enumeration
  - Checkbox
  - Table
  - Values Table
  - Tree Table

- MM editors use these widgets
  Defining the CDM data update behaviour for their update, delete and create user actions

- Transactional Updates
  Undo / redo



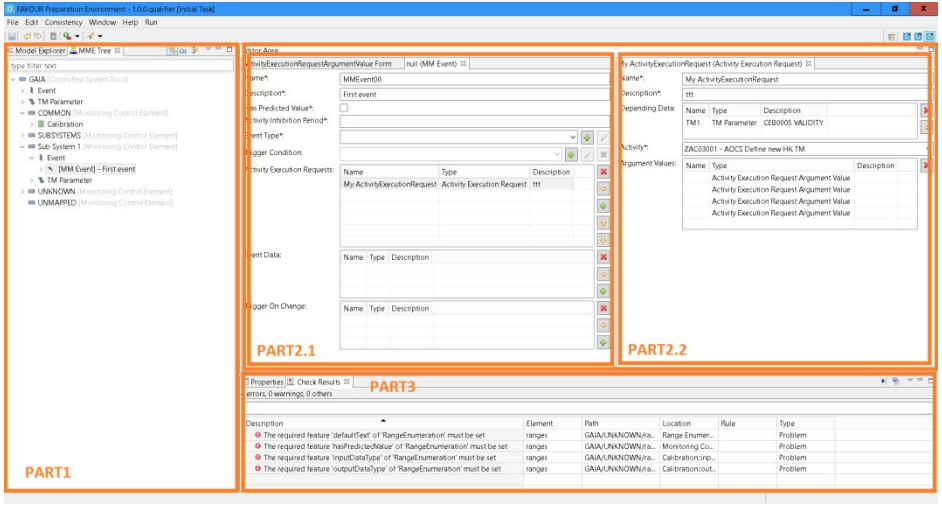
MME Views

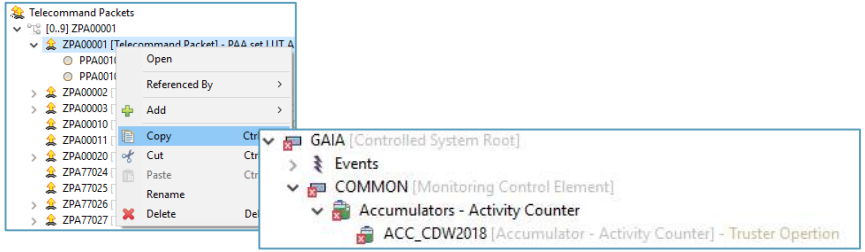
MME Perspective

PART1 - Mission Model Navigator

- Treeview organisation of MM Items
  - Types by icon
  - Name & Description
  - Group Types
  - Errors via decorators

- All operations via Context Menus
  - Single Item creation
  - Single/Multiple Deletion
  - Open Editor
  - Copy/cut and paste of single or multiple items
  - drag and drop of items between elements
  - drag and drop of items into editors
  - Filter by type, matching text





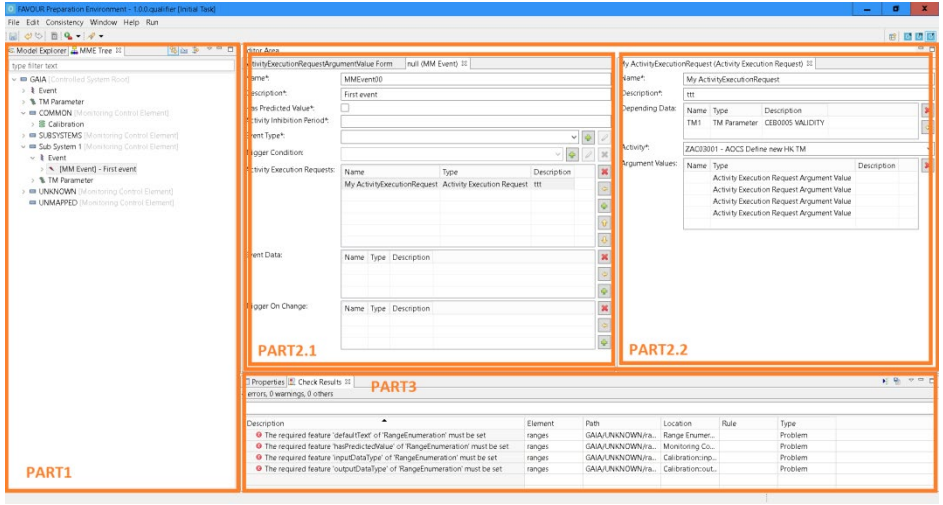
MME Views

MME Perspective – Parts 2 & 3

PART2 - Mission Model Editors

- PART2.1 MM Item Editors
  - Dedicated Editors
  - Items stacked

- PART2.2 Sub-Editor (if applicable)
  - Editor for the selected item in PART2.1
  - NOT stacked
  - Closes when distinct item is selected in PART2.1



PART3 - Other Views

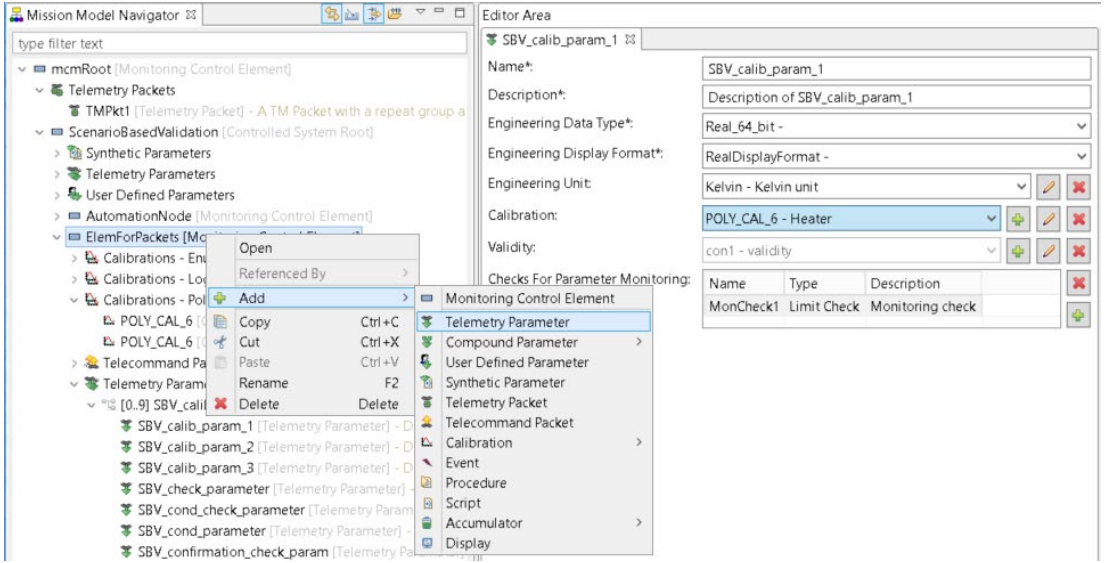
- Problems (Check Results)
- Console
- Properties
- Progress
- Configuration Control (other perspective)

MME Forms

Creating a TM Parameter

Telemetry parameter created from MCM tree

- Mandatory: data type and display format
- Optional: unit, calibration, validity and monitoring checks



MME Forms

Creating a TM Parameter with Calibration

Mission Model Navigator

type filter text

- mcmRoot [Monitoring Control Element]
  - Telemetry Packets
    - TMPkt1 [Telemetry Packet] - A TM Packet with a repeat group a
  - ScenarioBasedValidation [Controlled System Root]
    - Synthetic Parameters
    - Telemetry Parameters
    - User Defined Parameters
    - AutomationNode [Monitoring Control Element]
    - ElemForPackets [Mo
      - Calibrations - Eru
      - Calibrations - Lo
      - Calibrations - Pol
        - POLY_CAL_6 [(
        - POLY_CAL_6 [(
      - Telecommand Pa
      - Telemetry Param
        - [0..9] SBV_calib
          - SBV_calib_param_1 [Telemetry Parameter] - D
          - SBV_calib_param_2 [Telemetry Parameter] - D
          - SBV_calib_param_3 [Telemetry Parameter] - D
          - SBV_check_parameter [Telemetry Parameter]
          - SBV_cond_check_parameter [Telemetry Param
          - SBV_cond_parameter [Telemetry Parameter]
          - SBV_confirmation_check_param [Telemetry Pa

Open		
Referenced By		
Add		
Copy	Ctrl+C	
Cut	Ctrl+X	
Paste	Ctrl+V	
Rename	F2	
Delete	Delete	

- Monitoring Control Element
- Telemetry Parameter
- Compound Parameter
- User Defined Parameter
- Synthetic Parameter
- Telemetry Packet
- Telecommand Packet
- Calibration
- Event
- Procedure
- Script
- Accumulator
- Display

Editor Area

SBV_calib_param_1

Name*:	SBV_calib_param_1
Description*:	Description of SBV_calib_param_1
Engineering Data Type*:	Real_64_bit -
Engineering Display Format*:	RealDisplayFormat -
Engineering Unit:	Kelvin - Kelvin unit
Calibration:	POLY_CAL_6 - Heater
Validity:	con1 - validity

Checks For Parameter Monitoring:

Name	Type	Description
MonCheck1	Limit Check	Monitoring check

POLY_CAL_6

Name*:	POLY_CAL_6
Description*:	Heater
Input Data Type*:	Unsigned_Integer_64_bit -
Input Display Format:	ByteStream display format -
Input Engineering Unit:	
Output Data Type*:	Real_64_bit -
Output Display Format:	RealDisplayFormat -
Output Engineering Unit:	Kelvin - Kelvin unit
Cal Direction*:	SRC2CAL_Only
Coefficients*:	[1.0, 2.0, 3.0, 4.0, 5.0]

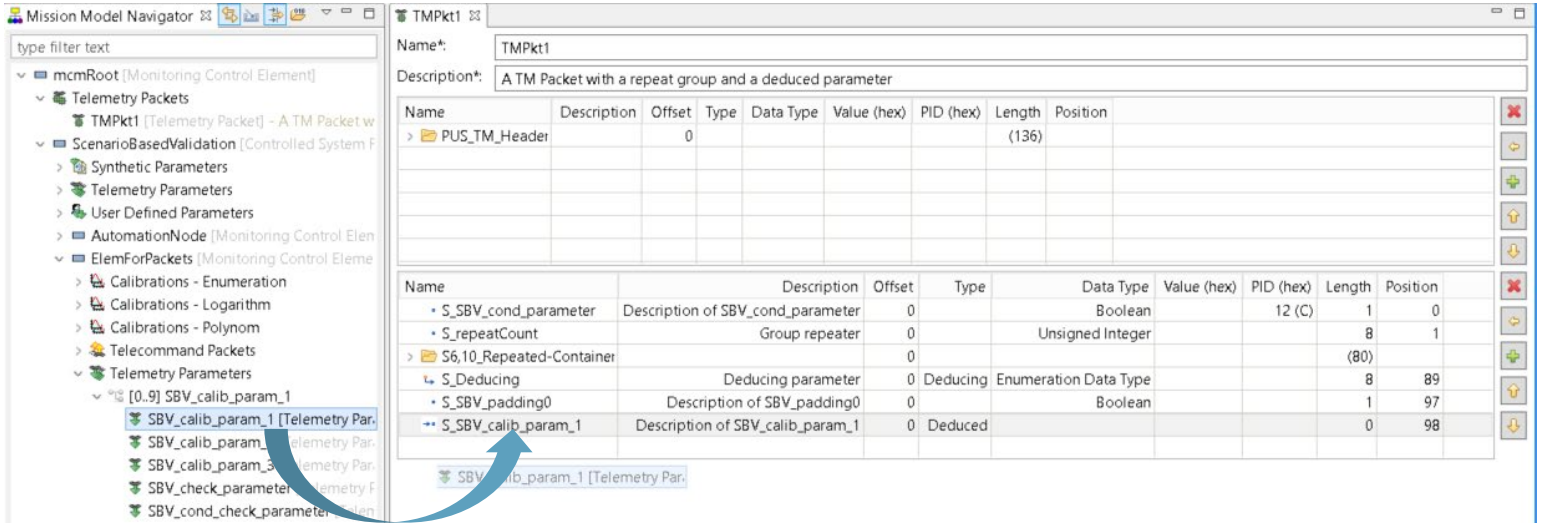
MME Forms

Adding a TM Parameter to a TM Packet

Telemetry parameter (MCM) dropped into packet (implementation)

Under the hood:

- A corresponding packet parameter is created and its data fields synchronised
- A mapping between the MCM parameter and the packet parameter is created and maintained.



MME Features

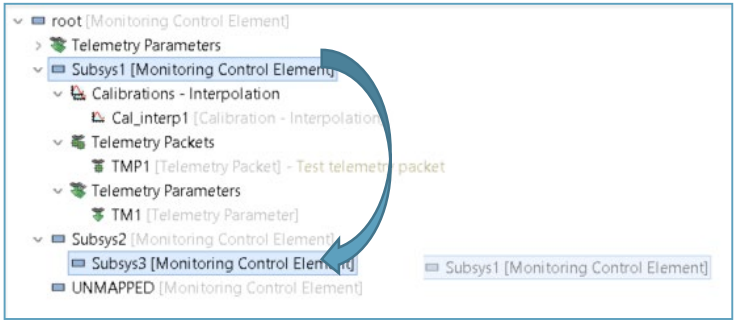
Configuration Item (CI) Management

The MME enforces the 1 CI per MCE(D) and per CI type

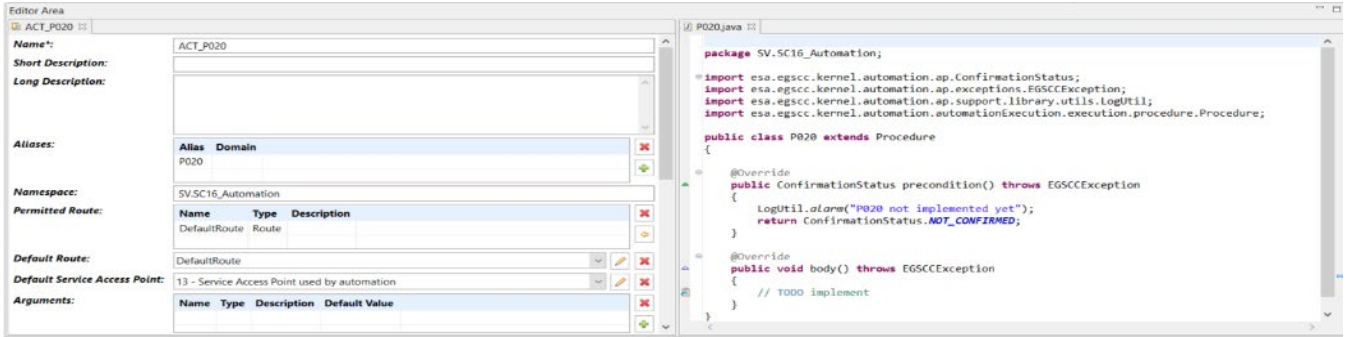
- MCEs and MCE Definitions (MCEDs) are placed in their own MCM CIs together with
- MCE-specific mapper CIs that reference the MCE's contents &
- MCE-specific Implementation CIs referenced by these mappers

MM Navigator Tree Editing

- MCED aspects & MCE-based CIs are managed invisibly



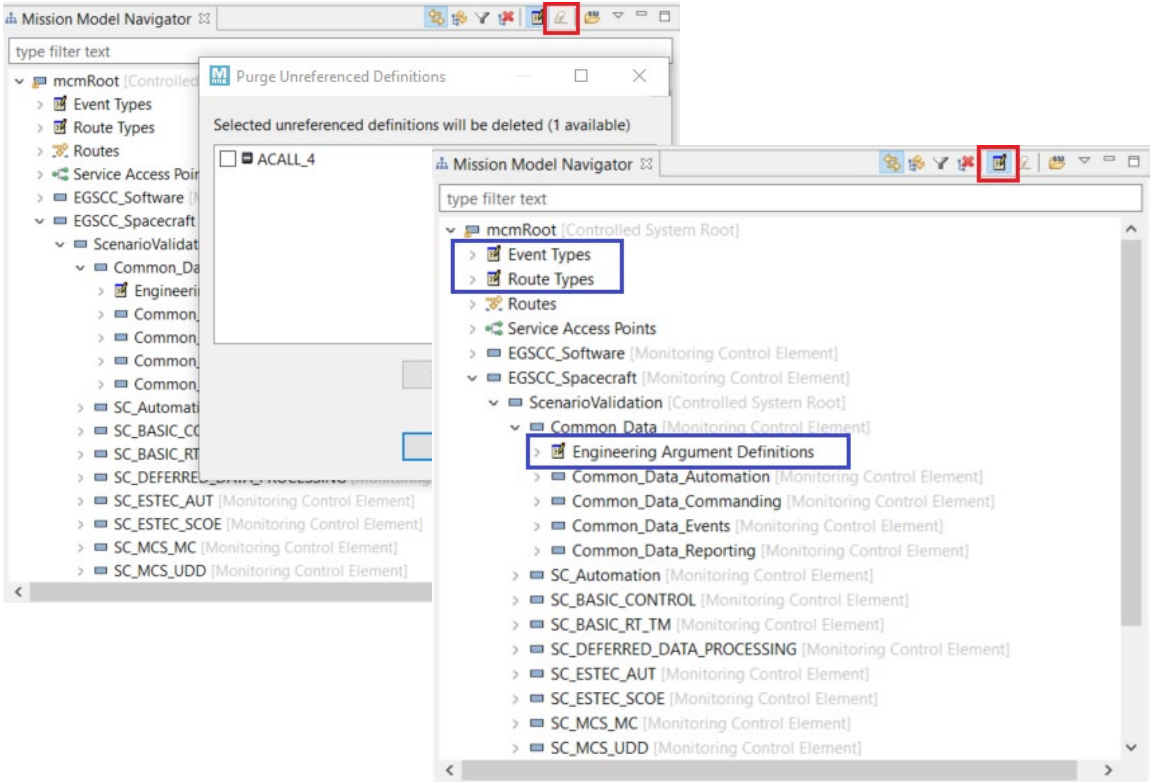
EAPL & Expression Editing



MME Features

Definitions Display & Managment

- Purge
- Editing



MME – PUS Addition

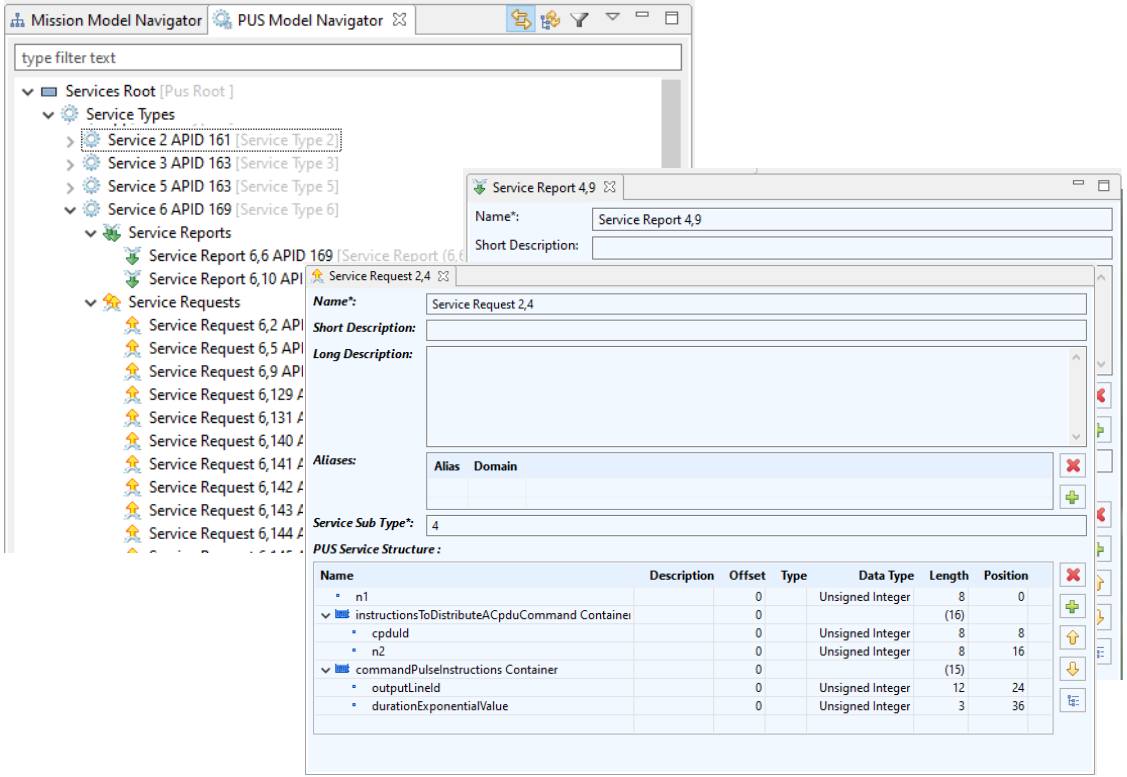
Services

Treeview

- New View of MME
- Classification in Treeview
- Request/Report, Type/SubType
- Add/Copy/Paste
- Drag&Drop into ApplicationProcesses

Editors

- PUS-A & PUS-C
- Synchronize edits



MME – PUS Addition

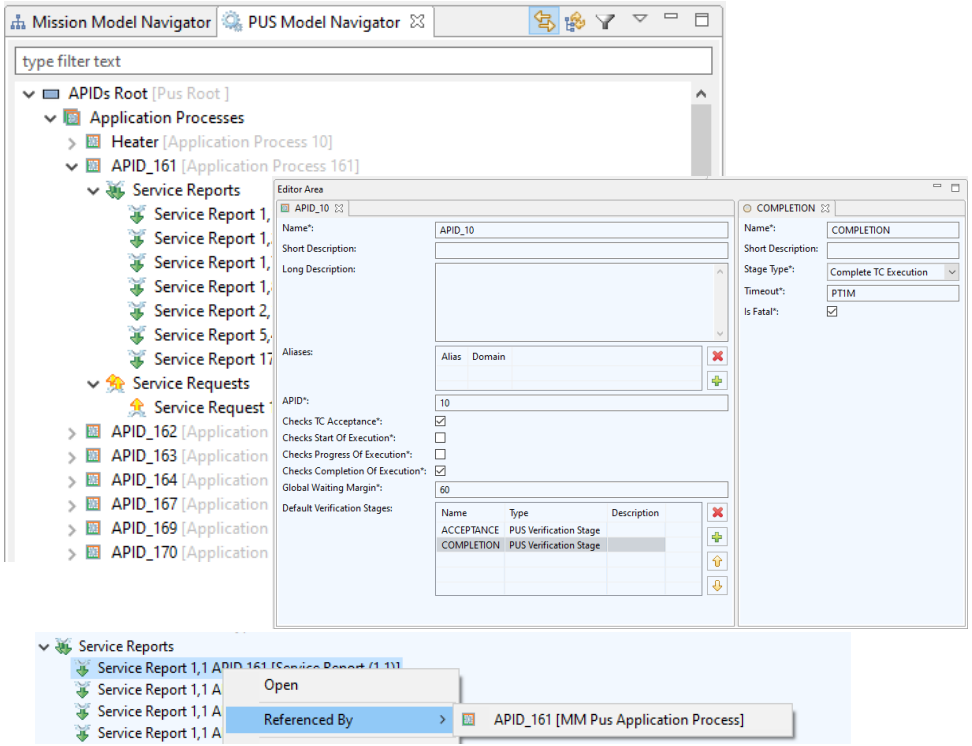
Application Processes

Treeview

- Classification in Treeview
- Request/Report, Type/SubType
- Drag&Drop into MM Navigator
- Referenced By

Editor

- AP specifics
- Synchronize edits

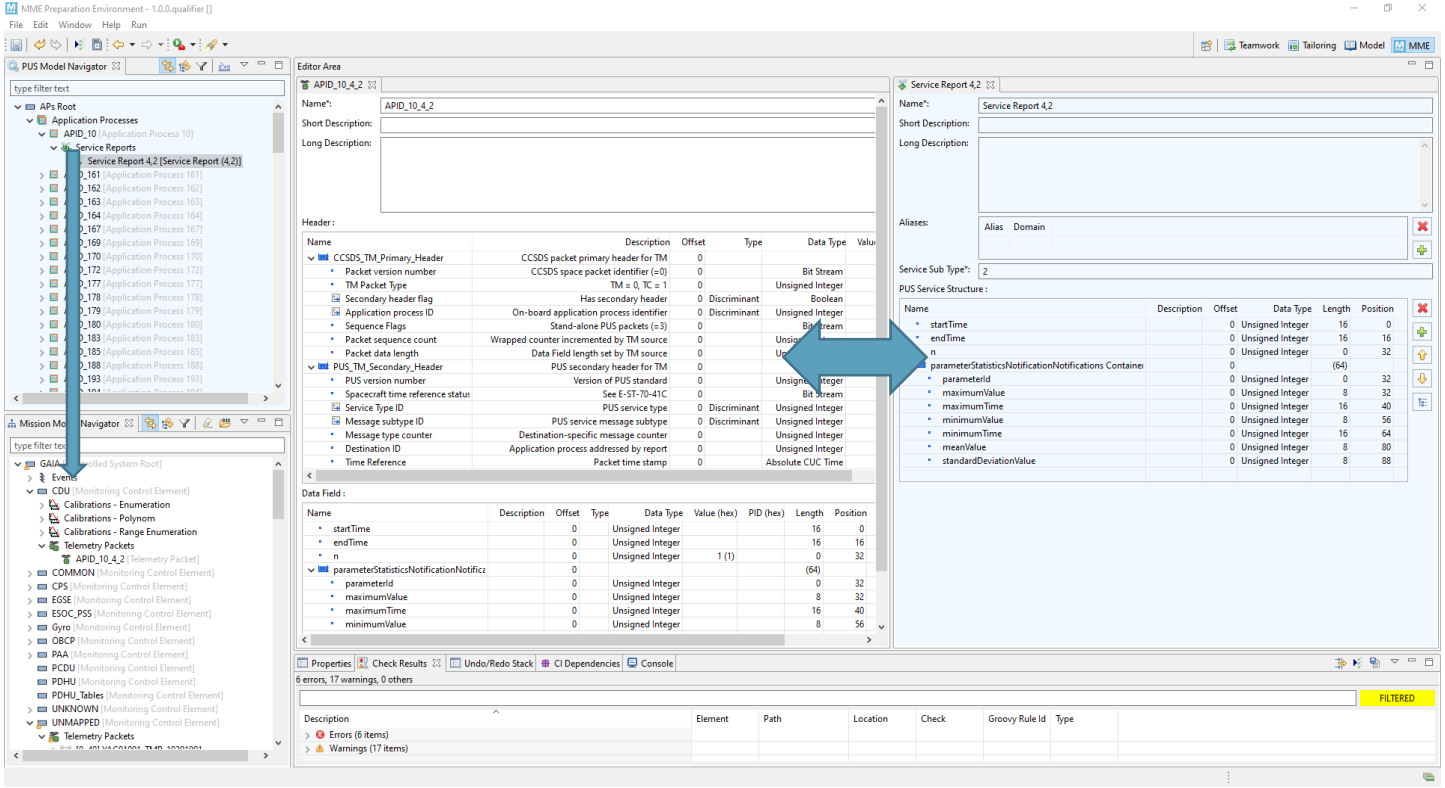


MME – PUS Addition

Packets

Treeview

- Drag&Drop creates packets
- Parameters can be exchanged
- Referenced By

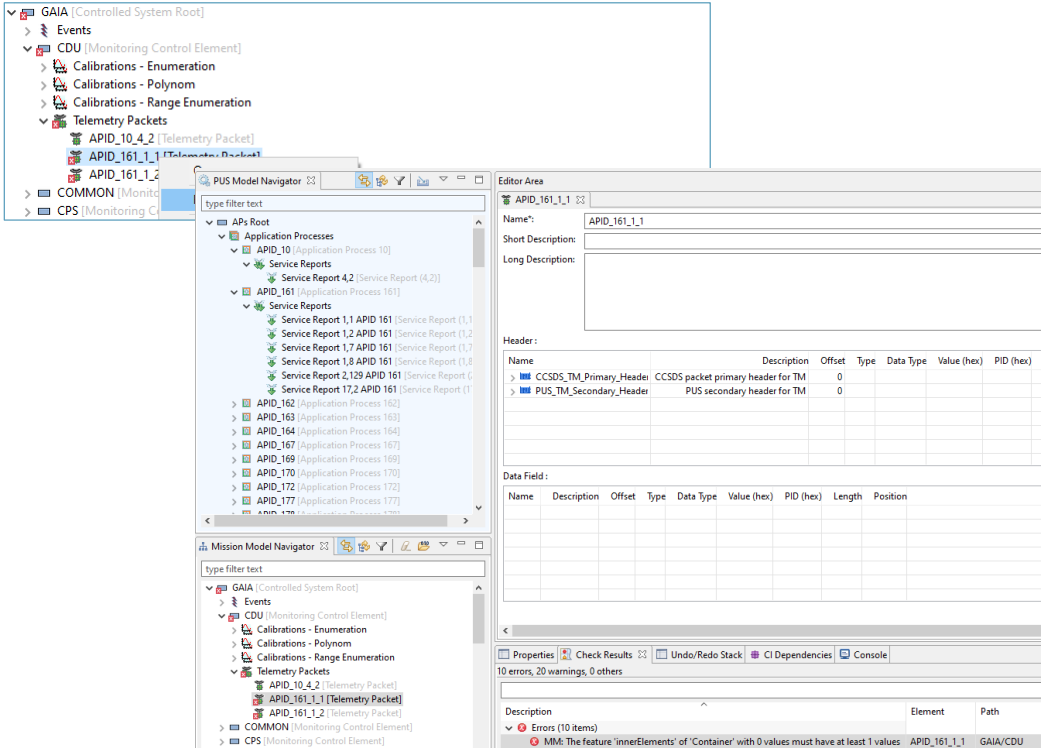


MME – PUS Addition

Packets

Checks

- Groovy, editable by user
- TC & TM Headers
- Data field checks (structure, order, data type, value checks,…)
- Markers in treeview
- Editor opened from Error View
- Checks performed on every packet update



MME SUMMARY

The Mission Model Editor

- **Hides and manages the complexity**
  Of the EGS-CC CDM via a simplified presentation layer and widget library
- **Supports drag and drop**
  For model refactoring & editors
- **Supports PUS Modelling**
  As a model extension
- **Enforces modularity**
  So that subtrees can be exported and imported as independent data models

It is targeted at AIT and OPS users

It makes CDM-based ‘tailoring’ data – the Mission Model – comprehensible to everyone

2. PME

Procedure Management Environment

- Overview
- Procedure Model
- DSL & Flowchart
- Client-Server Architecture
- MOIS Framework
- Screenshots

PME Overview

Features

Procedure Editor

- Allows users to view and edit satellite test and operations procedures
- Compatible with EGS-CC
- Visualizes procedures textually and graphically in a simple form
- Supports mapping of requirements to procedures or procedure steps
- Checks consistency of a procedure’s syntax and semantics
- Supports Product Export & Automation capabilities with EGS-CC
- Supports Publishing capabilities

Test Environment

- Test session management
- Logging of procedure execution in test sessions
- Requirements coverage in procedure test logs

PME Procedure Model

Principles and key points of the solution

The Procedure model consists of

- A minimal model with Steps, test requirements and branching (for a graphical representation)
- An executable extension to this minimal model (Groovy)

The Procedure model has

- Verifiable steps
- Executable statements
- Basic ‘procedural’ branching

PME Procedure Model

Model based DSL

PME DSL derived from the model

- Simple Step and Statement structure
- Basic branching and loops (If, While, For ...)
- All keywords are prompted for
- Mission Model look ups and checks
- Variable management delegated to Groovy

Other syntaxes are possible with the same procedure model

```
1  // PME procedure executable example.
2
3  Procedure BasicStructure
4  author "Robert Blommestijn"
5  testObjectives "Start from cold or warm an                                  nominal states"
6  testConfiguration "SUT serial number ABC12
7  {
8      // Procedure body starts here
9      Step initialisation "Initial boot"
10
11     // Requirements (from requirements datab  ) tested by this step
12     testRequirements reqXY1, reqAB2, reqCD3
13     {
14         // Step pre-condition. Step will not execute if this fails
15         PreConditionCheck parameters CEB0009V "CEB0009V = true"
16
17         // 'If' statement based on a Boolean test
18         If parameters NPF11119,NPA00017,CEB0033  "NPF11119 + NPA00017 < CEB0033 + 1"
19         Step StartupSequence "Starts basic platform"
20         {
21             // 'While' statement. Step is repeated until condition
22             While  parameters CEB0026 "CEB0026 < 373.0"
23             Step WarmUp "Perform warm up activities"
24             {
25                 // Send 2 TCs
26                 ExecuteActivity ZCPDU021                                    // Gyro 1a Off
27                 ExecuteActivity ZAC03005 arguments PDW90011="ENABLE"        // AOCS enable HK TM
28
29                 // Wait for the preceding TCs to complete their execution
30                 Synchronise
31             }
32
33             // Test a parameter until it is true
34             Wait pollingInterval 500 timeout 10000
35         }
36         Else
37
38         // Manual override
39         Step ManualInput "Perform a manual setting"
40         {
41             Prompt "Turn the dial 3 clicks to the right and enter the reading:" interval = "11.3"
42             ExecuteActivity ZAC03129 arguments PDW90011=interval, PDW00031="1"   // AOCS Set TM HK interval
43             Synchronise
44         }
```

Test requirements to Step mapping

Boolean Groovy expression

TCs sent asynchronously

‘interval’ variable used as argument

PME Procedure Model

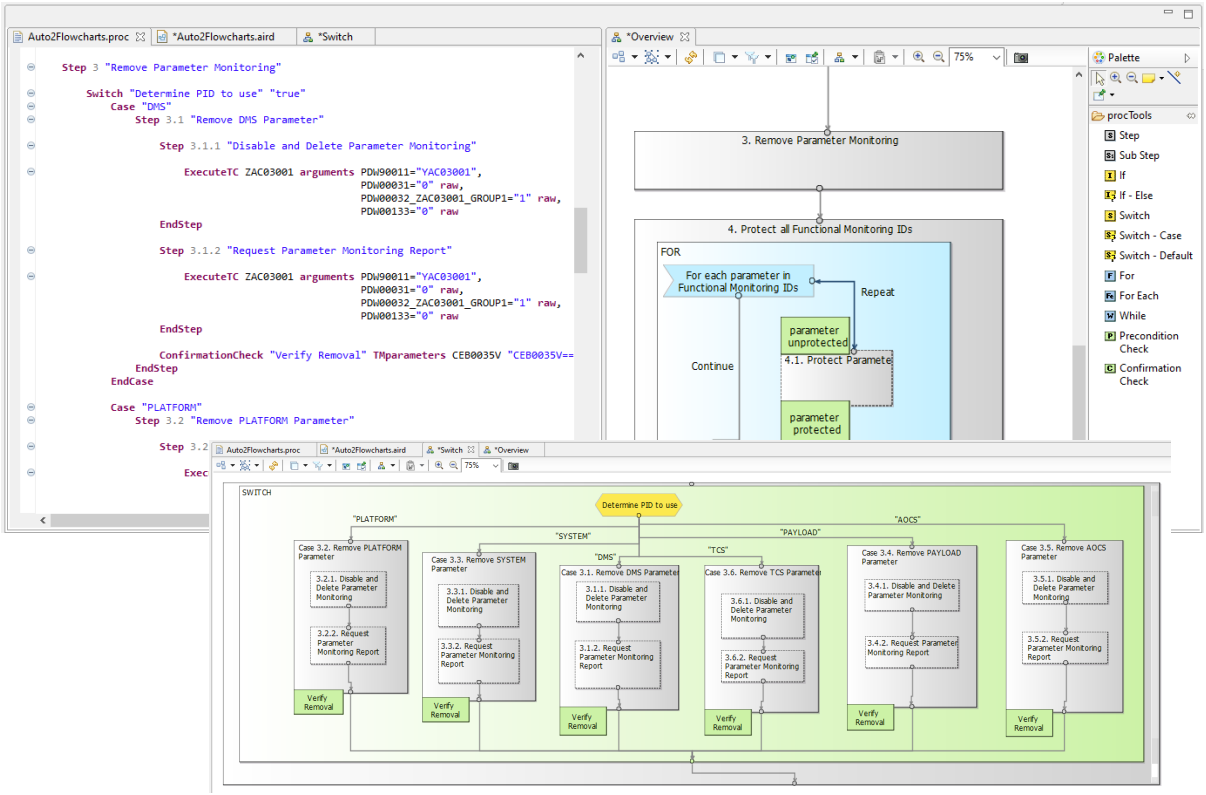
Flowcharter

Model Representation

- Steps (precondition & confirmation)
- Branching and loops

Features

- Several flowcharts
- Different level of granularity
- Mutual model updates
- Palette of shapes

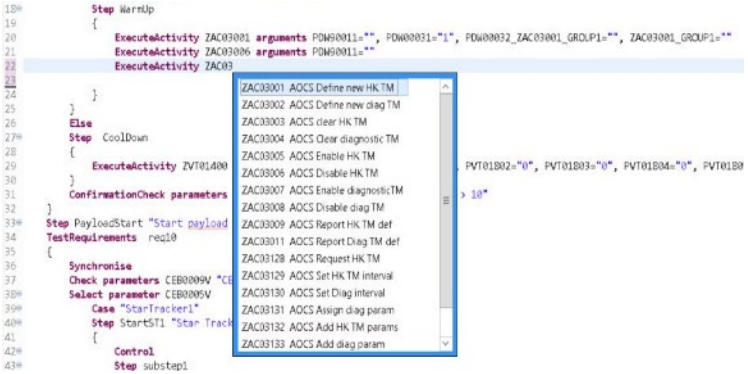


PME Procedure Model

Text Editor

Features

- Content assist for TCs, TC enumeration arguments, parameters, packets, events & requirements
- TC argument completion
- Raw and calibrated values for TC arguments and parameters
- Units insertion for TC arguments and parameters
- Expressions – delegated to Groovy
- Variable management
- Procedure calls with arguments
- Hover text for Mission Model and requirement descriptions
- Dynamic validation of syntax and content, including step numbering and expressions.



PME Procedure Model

DSL Execution and tracing

An executable Groovy script is generated on procedure save

- Groovy calls the PME Execution API
  (for retrieving parameters, sending TCs, waiting for Packets etc.)
- Runs anywhere
  (editor or on the server)
- Debugging / Stepwise execution
  Xtext supports tracing (executing Groovy script is hidden)

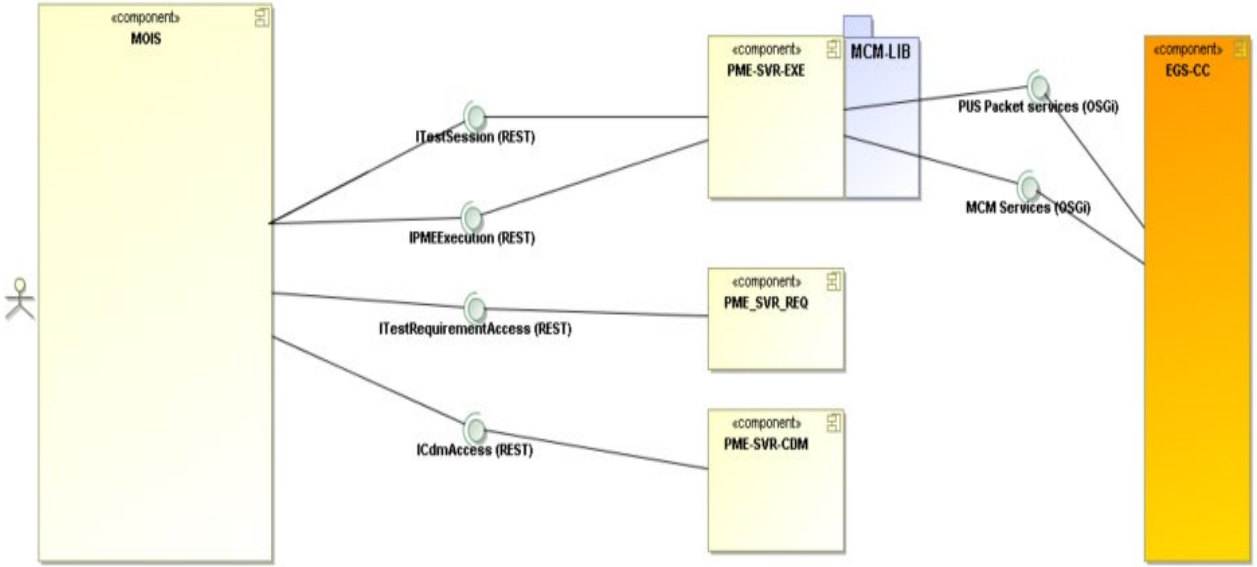
```
Procedure StartupTest
author 'Anon E Maus'
testObjectives 'Start from cold or warm and ensure all instruments are in their nominal states'
testConfiguration 'SUT serial number ABC1234'
{
    Step Initialisation 'Initial boot sequence'
    {
        // Check if system is in expected state
        If parameters NPF11119, NPA00017, CEB0033 'NPF11119 + NPA00017 > CEB0033 + 1'
        Step StartupSequence 'Start platform tasks'
        {
            ExecuteActivity ZAC03002 arguments PDW90011='ENABLE', PDW00031='678'
            ExecuteActivity ZACXXXX arguments TIMEOUT='210'
            Synchronise

            Log 'Execute other statements'

            ExecuteActivity ZACZZZ
            Synchronise
        }
    }
}
```

```
import java.util.concurrent.Future
import java.util.concurrent.Execut

def executor = Executors.newWorkStealingPool()
def executingTcs = []

// Start of execution
println 'Initialisation :  Initial boot sequence'
println 'Statement type:  If'
def NPF11119 = getParamValue("NPF11119")
def NPA00017 = getParamValue("NPA00017")
def CEB0033 = getParamValue("CEB0033")
if (NPF11119 + NPA00017 > CEB0033 + 1) {

    println 'StartupSequence :  Start platform tasks'

    // Initiate Activities asynchronously
    sendtc('ZAC03002', "ENABLE", "678")
    sendtc('ZACXXXX', "210")
```

PME Design - Client-Server Architecture

PME server provides REST API for

- Access to the Mission Model (CDM) and requirements
- Test session management
- Procedure Execution (PME server accesses EGS-CC services via MCM-LIB / JEEL)



PME-SVR-EXE: Procedure Execution & Test Session Service Interfaces connecting to MCM-LIB

PME-SVR-REQ: Requirements Access

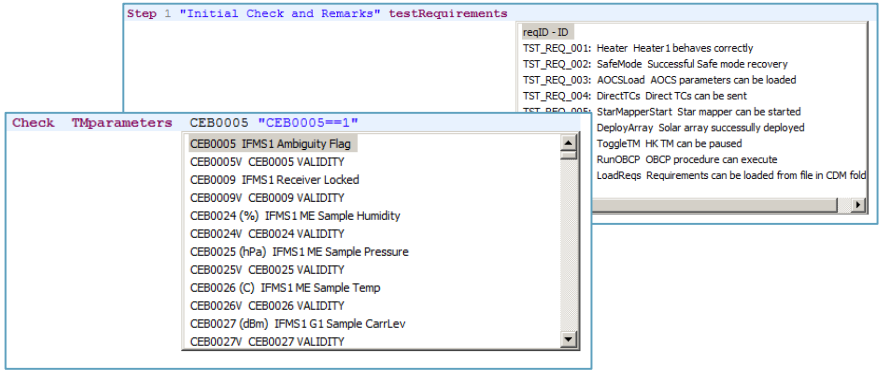
PME-SVR-CDM: CDM Access

MCM-LIB: Java Library connecting to EGS-CC MCM

PME Server Services

Upload

- Test Requirements
- Mission Model



Procedure Editing Services

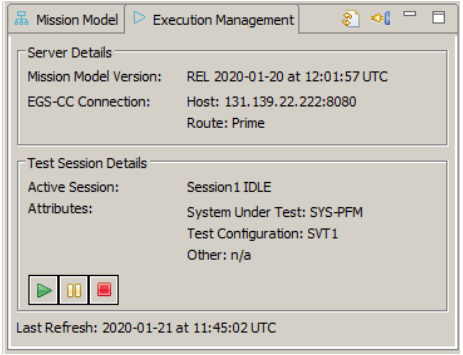
- Access to Requirements
- Access to Mission Model
- Consistency Checking

Execution Services

- Send Activity
- Sample Parameter
- Wait for Event
- Wait for Packet
- Enable/disable Parameter monitoring
- Log Message

Test Management Services

- Configure EGS-CC Connection
- Define Test Sessions
- Start/Stop/Pause Session
- Test Session Log Retrieval



MOIS Framework

Manufacturing and **O**perations **I**nformation **S**ystem is the solution for missions preparation & manages the mission lifecycle

Used for over 100 spacecraft by

- ESA
- SpaceOpal (Galileo)
- Airbus-DS
- DLR
- ASI
- TAS-I
- TESAT
- Astroscale
- GMV
- …and many more

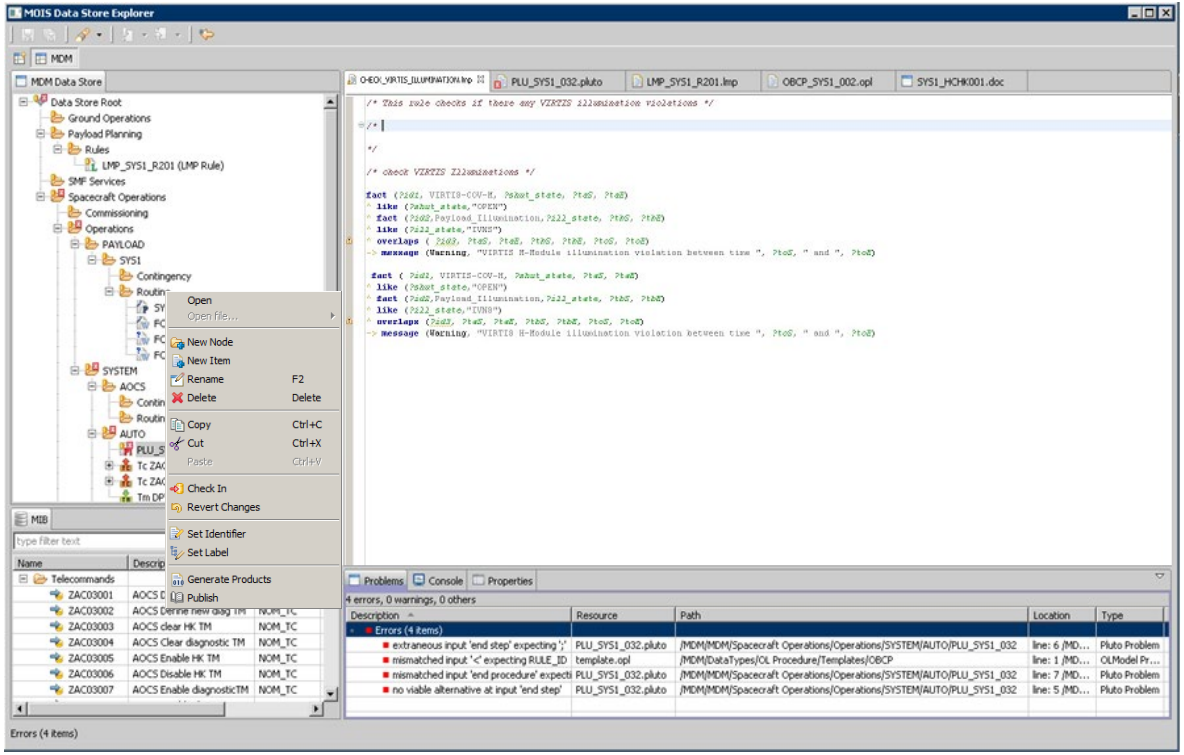
De facto standard for procedure exchange in ESA missions

FAVOUR & MOIS

MOIS Framework

- RCP / EMF
- Host any Editor as plugin

- Editors
  - DSL Procedure Script
  - Flowcharts
  - Custom editors
- Framework Services
  - Local Workspace
  - Configuration Control
  - Publishing
  - Reporting
  - Product Export



FAVOUR & MOIS

New Editors

- MME & PME

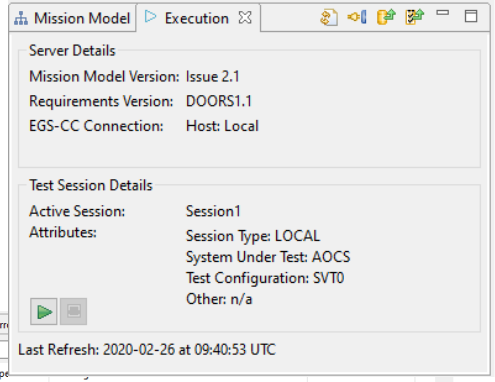
Test Session Management

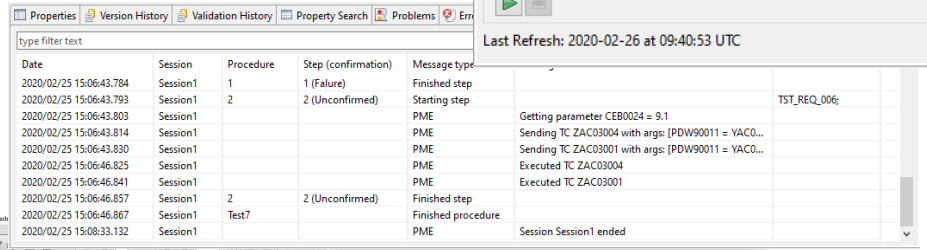
- New view
- Upload Mission Model
- Upload Requirements
- Manage (Configure, Attach, Start/Stop…)

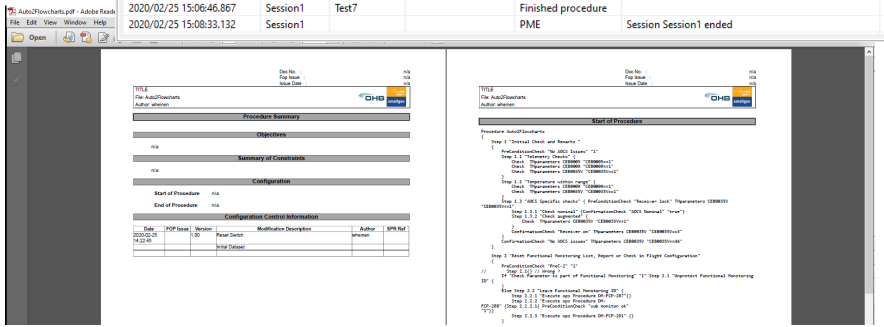
Test Logging

- New view
- Reporting

Publishing

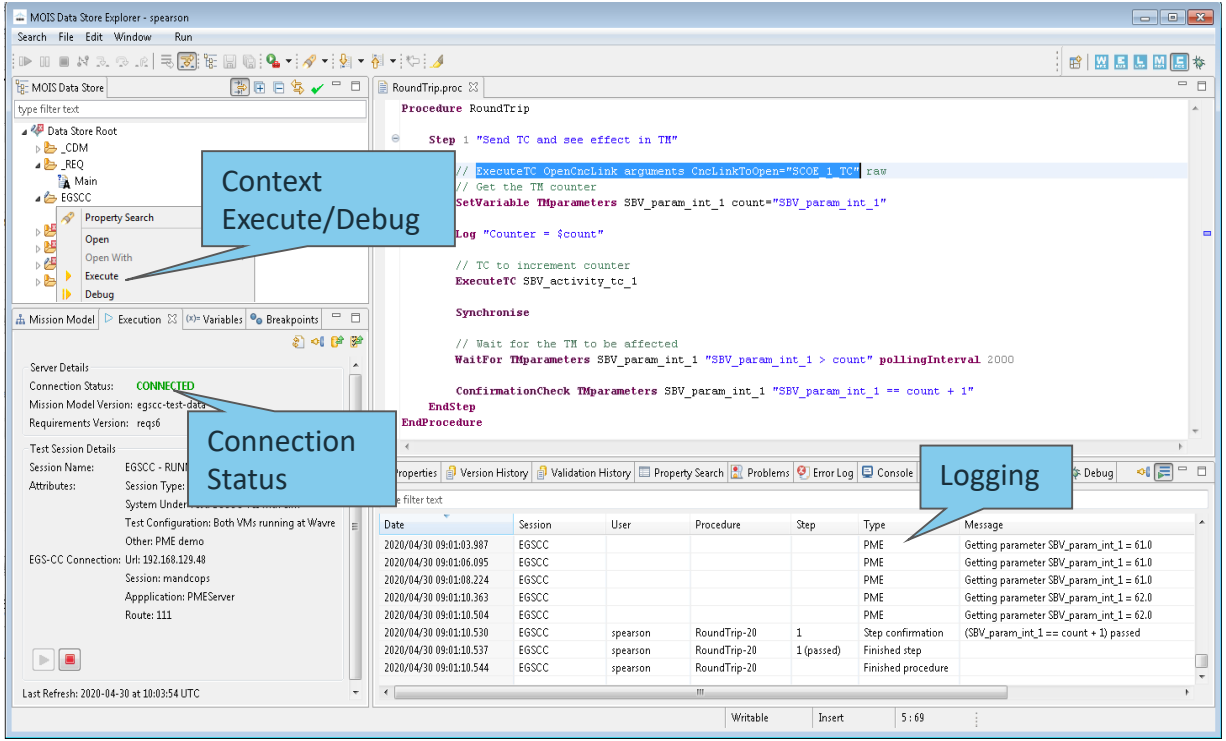






FAVOUR & MOIS Screenshots

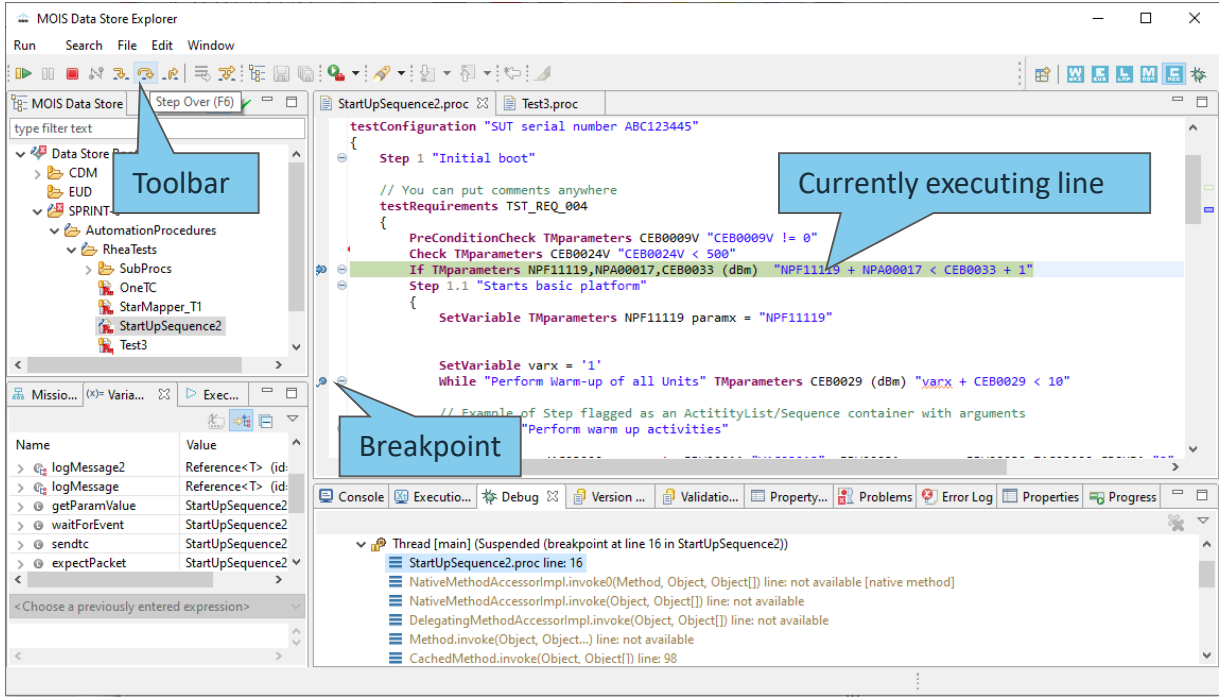
- MME is an external Editor
  Started from MOIS
  Data is configuration controlled
- PME procedures
  Create, open & execute from treeview
- Monitor connection, test session & logs



FAVOUR & MOIS Screenshots

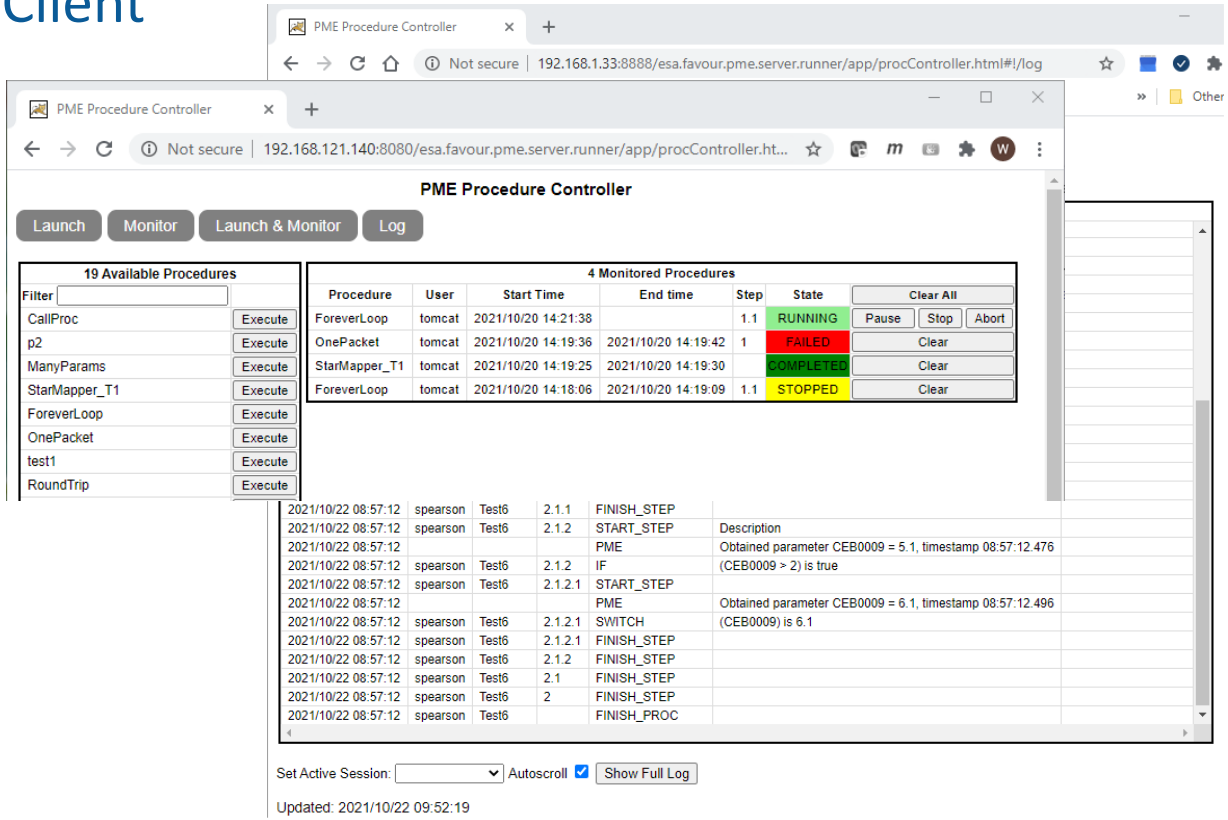
PME – Stepwise Execution / Debugging

- **Debug toolbar features**
  Step, go-all, pause, stop…
- **DSL features**
  Set breakpoints, save & run
- **Variables**
  Examine, set
- **Console & Debug View**
  Messages, execution status



PME-Controller Web Client

- **Launch procedures**
  - Uplinked procedures (as executable Groovy) listed
- **Monitor procedures**
  - Running and completed procedures with times, current step and state
  - Running procedures can be paused/resumed or aborted
  - Finished procedures can be cleared from the list
- **Log Execution**
  - Real time logging and retrieval



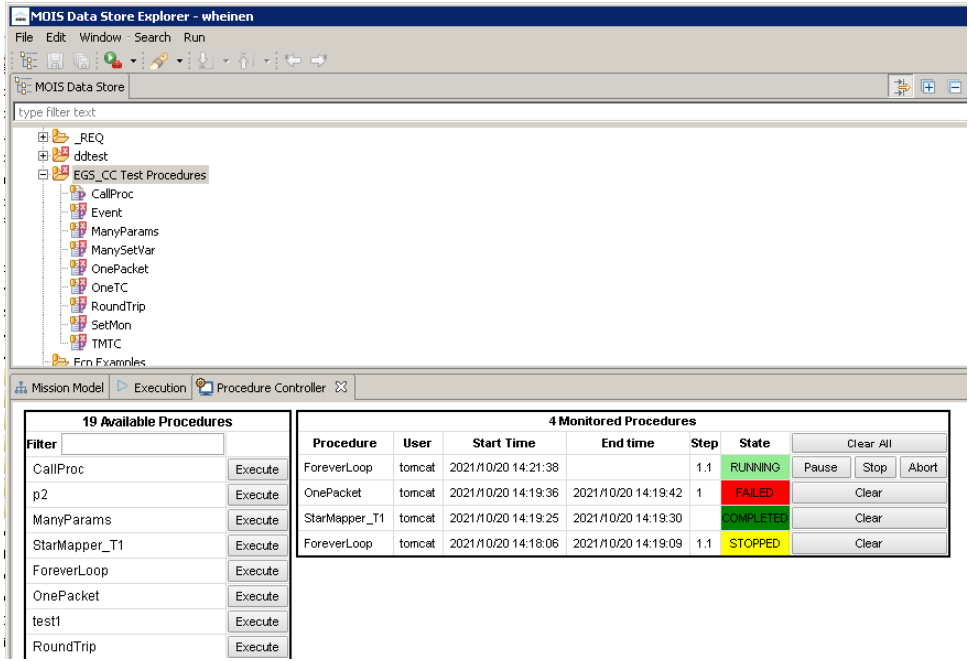
PME-Controller Web Client

Datastore integration with browser

- Same web view shown in MOIS Data Store Explorer
- Views served by REST APIs – can be transitioned to web views within MOIS

Execution REST API

- Procedure execution on server
- Remote (and local) debugging possible
- Ready for Scheduler integration



The Chromium Project

PME SUMMARY

PME Procedures

- Model based solution gives **flexibility** for many representations: DSLs and flowcharts
- Client-Server solution make the editors light-weight
- DSL **hides** unnecessary language specific mechanics (reduces testing effort & can be read by non-experts for review)

Test Management System

- Mapping of requirements to procedure content gives the test **coverage**
- Execution logging gives the successful requirements **verification** coverage (VCDs)

CONCLUSION

FAVOUR & MOIS

- PME & MME integrated into MOIS is the complete solution for EGS-CC tailoring **data preparation**
- PME gives an **automation solution** for EGS-CC
- The MOIS framework brings the **teaming, test management & publishing aspects** that make it a solution for missions at AIT or operations stage

THANK YOU