



# 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

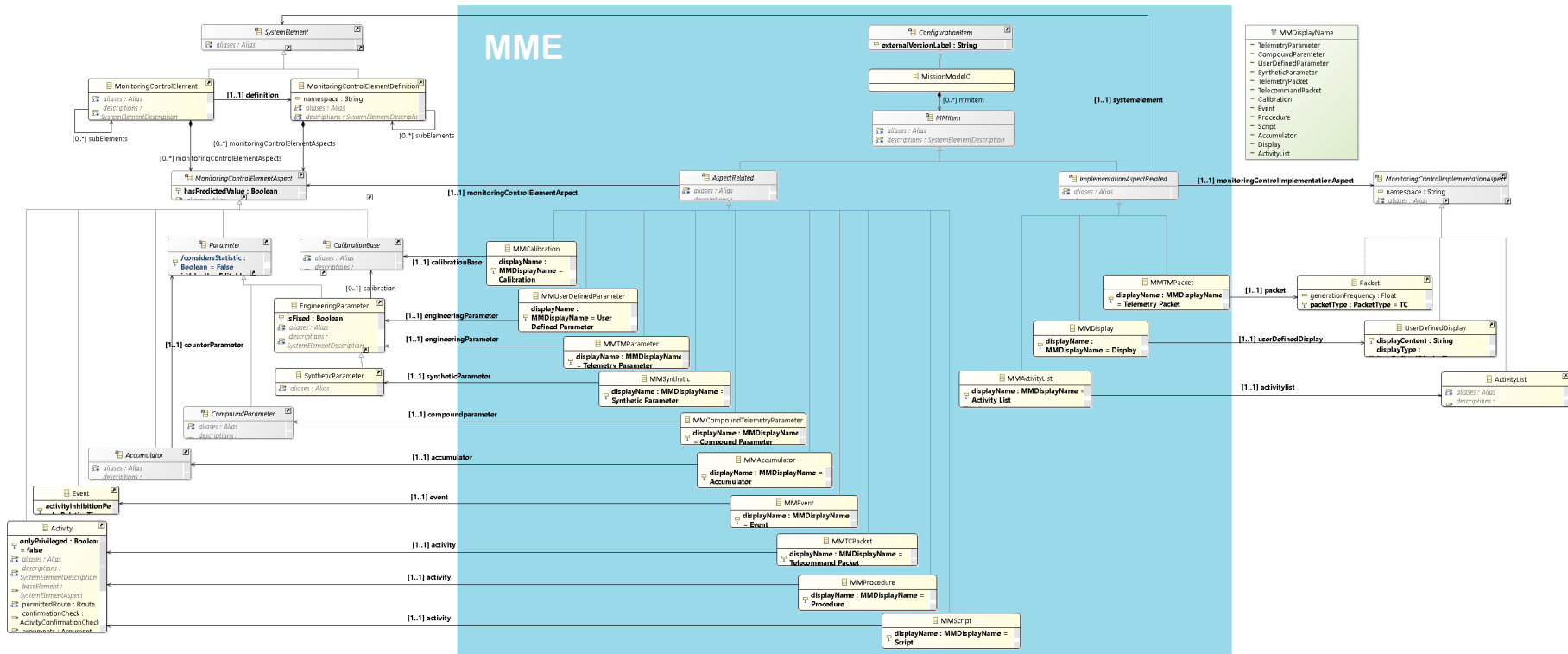
# MME Overview

## 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

TM1 (TM Parameter)	MMEv1 (MM Event)																									
Name*:	Default_event																									
Description*:	sdfgdsfgg																									
Has Predicted Value*:	<input type="checkbox"/>																									
Activity Inhibition Period*:	PT0S																									
Event Type*:	MyNewEventType -	 																								
Trigger Condition:		  																								
Activity Execution Requests:	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> <th></th> </tr> </thead> <tbody> <tr> <td>My ActivityExecutionRequest</td> <td>Activity Execution Request</td> <td>ttt</td> <td></td> </tr> <tr> <td>My ActivityExecutionRequest</td> <td>Activity Execution Request</td> <td></td> <td></td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>	Name	Type	Description		My ActivityExecutionRequest	Activity Execution Request	ttt		My ActivityExecutionRequest	Activity Execution Request															
Name	Type	Description																								
My ActivityExecutionRequest	Activity Execution Request	ttt																								
My ActivityExecutionRequest	Activity Execution Request																									
																										
																										
																										
Event Data:	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> <th></th> </tr> </thead> <tbody> <tr> <td>NewEngineeringParameterqq</td> <td>TM Parameter</td> <td>ewwert</td> <td></td> </tr> <tr> <td>NewEngineeringParameter</td> <td>TM Parameter</td> <td></td> <td></td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>	Name	Type	Description		NewEngineeringParameterqq	TM Parameter	ewwert		NewEngineeringParameter	TM Parameter															
Name	Type	Description																								
NewEngineeringParameterqq	TM Parameter	ewwert																								
NewEngineeringParameter	TM Parameter																									
																										
Trigger On Change:	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> <th></th> </tr> </thead> <tbody> <tr> <td>TM1</td> <td>TM Parameter</td> <td>CEB0005 VALIDITY</td> <td></td> </tr> <tr> <td>NewEngineeringParameter</td> <td>TM Parameter</td> <td></td> <td></td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>	Name	Type	Description		TM1	TM Parameter	CEB0005 VALIDITY		NewEngineeringParameter	TM Parameter															
Name	Type	Description																								
TM1	TM Parameter	CEB0005 VALIDITY																								
NewEngineeringParameter	TM Parameter																									
																										

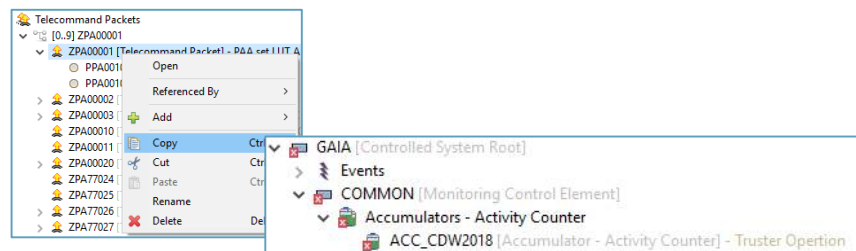
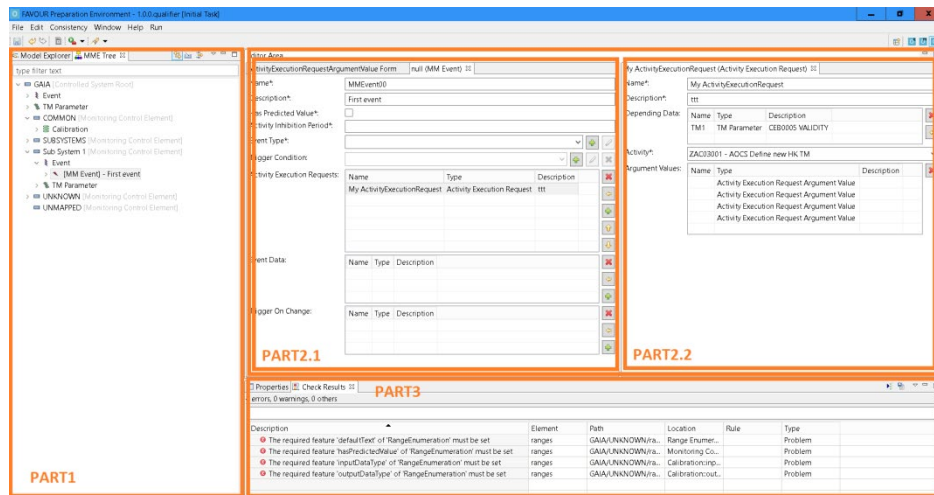


# MME Views

## 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

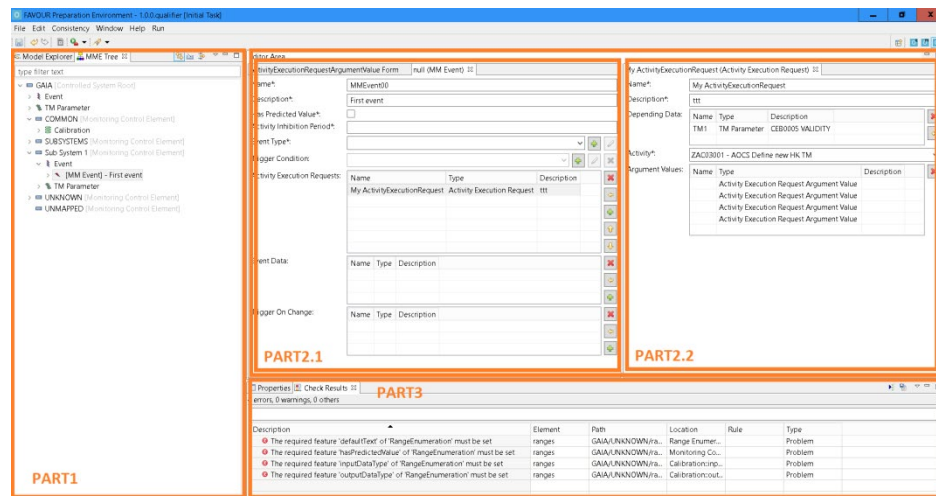
## MME Perspective – Parts 2 & 3

### 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

The screenshot shows the Mission Model Navigator on the left and the Editor Area on the right. The Navigator displays a tree structure with 'SBV\_calib\_param\_1' selected. A context menu is open over it, with 'Add' > 'Telemetry Parameter' highlighted. The Editor Area shows the configuration for 'SBV\_calib\_param\_1'.

**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

# MME Forms

## Creating a TM Parameter with Calibration

The screenshot displays the Mission Model Navigator (left) and the Editor Area (right) in a software application. The Mission Model Navigator shows a tree view of the model structure, with a context menu open over the 'SBV\_calib' folder. The Editor Area shows the configuration for two parameters: 'SBV\_calib\_param\_1' and 'POLY\_CAL\_6'.

**Mission Model Navigator (Left):**

- 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 [Monitoring Control Element]
      - Calibrations - En...
      - 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...

**Editor Area (Middle):**

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

**Editor Area (Right):**

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.

The screenshot shows the Mission Model Navigator interface. On the left, a tree view shows the hierarchy: mcmRoot [Monitoring Control Element] > Telemetry Packets > TMPkt1 [Telemetry Packet] - A TM Packet w... > ScenarioBasedValidation [Controlled System F...]. Under Telemetry Parameters, a parameter [0..9] SBV\_calib\_param\_1 is selected. A blue arrow points from this parameter to the main workspace.

The main workspace shows the details for TMPkt1. The Name is TMPkt1 and the Description is A TM Packet with a repeat group and a deduced parameter. Below this is a table of parameters:

Name	Description	Offset	Type	Data Type	Value (hex)	PID (hex)	Length	Position
> PUS_TM_Header		0					(136)	
• S_SBV_cond_parameter	Description of SBV_cond_parameter	0		Boolean		12 (C)	1	0
• S_repeatCount	Group repeater	0		Unsigned Integer			8	1
> S6,10_Repeated-Container		0					(80)	
↳ S_Deducing	Deducing parameter	0	Deducing	Enumeration Data Type			8	89
• S_SBV_padding0	Description of SBV_padding0	0		Boolean			1	97
•• S_SBV_calib_param_1	Description of SBV_calib_param_1	0	Deduced				0	98

Below the table, a detailed view of the selected parameter S\_SBV\_calib\_param\_1 is shown, including its Name, Description, Offset, Type, Data Type, Value (hex), PID (hex), Length, and Position.

# MME Features

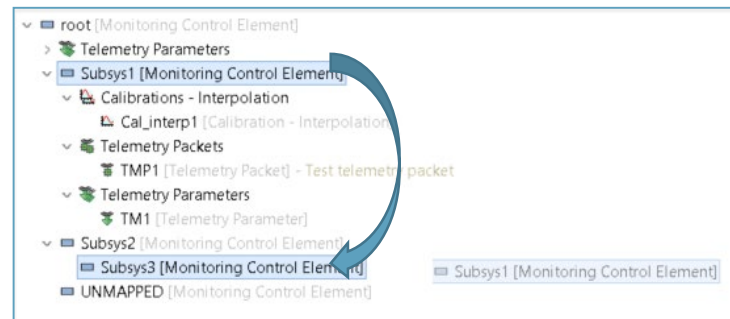
## Configuration Item (CI) Management

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

**Editor Area**

ACT\_P020

**Name:** ACT\_P020

**Short Description:**

**Long Description:**

**Aliases:**

Alias	Domain
P020	

**Namespace:** SV.SC16\_Automation

**Permitted Route:**

Name	Type	Description
DefaultRoute	Route	

**Default Route:** DefaultRoute

**Default Service Access Point:** 13 - Service Access Point used by automation

**Arguments:**

Name	Type	Description	Default Value
------	------	-------------	---------------

**P020.java**

```
package SV.SC16_Automation;

import esa.egscc.kernel.automation.ap.ConfirmationStatus;
import esa.egscc.kernel.automation.ap.exceptions.EGSCCEException;
import esa.egscc.kernel.automation.ap.support.library.utils.LogUtil;
import esa.egscc.kernel.automation.automationExecution.execution.procedure.Procedure;

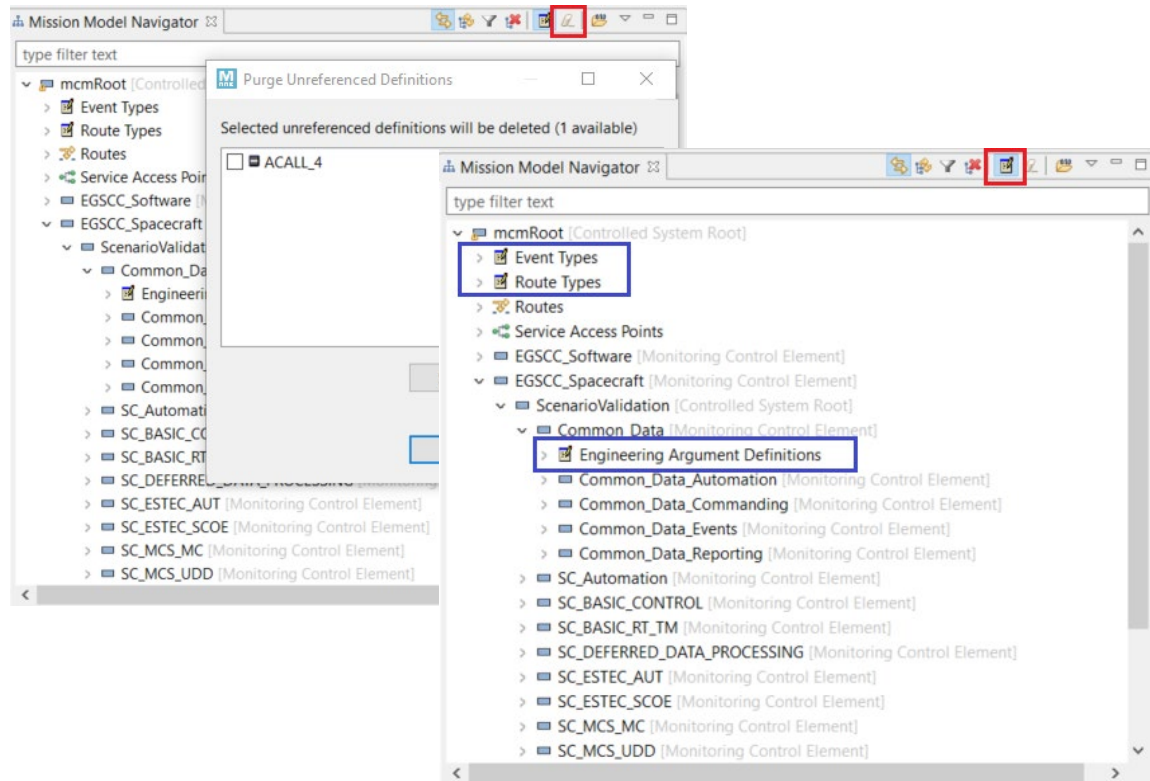
public class P020 extends Procedure
{
    @Override
    public ConfirmationStatus precondition() throws EGSCCEException
    {
        LogUtil.alane("P020 not implemented yet");
        return ConfirmationStatus.NOT_CONFIRMED;
    }

    @Override
    public void body() throws EGSCCEException
    {
        // TODO implement
    }
}
```

# MME Features

## Definitions Display & Management

- Purge
- Editing



# MME – PUS Addition

## Services

### Treewiew

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

### Editors

- PUS-A & PUS-C
- Synchronize edits

The screenshot displays two windows: 'Mission Model Navigator' and 'PUS Model Navigator'. The 'Mission Model Navigator' shows a tree view of services under 'Services Root [Pus Root]'. The 'PUS Model Navigator' shows a detailed view of a service, including fields for Name, Short Description, Long Description, Aliases, and Service Sub Type. A table at the bottom shows the PUS Service Structure with columns for Name, Description, Offset, Type, Data Type, Length, and Position.

Name	Description	Offset	Type	Data Type	Length	Position
n1		0		Unsigned Integer	8	0
instructionsToDistributeACpduCommand Container		0		(16)		
cpdul		0		Unsigned Integer	8	8
n2		0		Unsigned Integer	8	16
commandPulseInstructions Container		0		(15)		
outputLineId		0		Unsigned Integer	12	24
durationExponentialValue		0		Unsigned Integer	3	36



# MME – PUS Addition

## Application Processes

### Treeview

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

### Editor

- AP specific
- Synchronize edits

The screenshot displays the Mission Model Navigator and PUS Model Navigator software interface. The top window shows a treeview of application processes under "APIDs Root [Pus Root]". The tree includes "Application Processes", "Heater [Application Process 10]", and "APID\_161 [Application Process 161]". Under "APID\_161", there are "Service Reports" (Service Report 1, Service Report 1, Service Report 1, Service Report 2, Service Report 5, Service Report 17) and "Service Requests" (APID\_162 [Application], APID\_163 [Application], APID\_164 [Application], APID\_167 [Application], APID\_169 [Application], APID\_170 [Application]).

The bottom window shows the "Editor Area" for "APID\_10". The editor area contains the following fields and controls:

- Name: APID\_10
- Short Description:
- Long Description:
- Aliases: Alias Domain
- APID#: 10
- Checks TC Acceptance:
- Checks Start Of Execution:
- Checks Progress Of Execution:
- Checks Completion Of Execution:
- Global Waiting Margin: 60
- Default Verification Stages:
 

Name	Type	Description
ACCEPTANCE	PUS Verification Stage	
COMPLETION	PUS Verification Stage	

A "COMPLETION" dialog is also visible, showing:

- Name: COMPLETION
- Short Description:
- Stage Type: Complete TC Execution
- Timeout: PTIM
- Is Fatal:

A separate screenshot at the bottom shows a context menu for "Service Reports" with "Referenced By" selected, pointing to "APID\_161 [MM Pus Application Process]".

# MME – PUS Addition

## Packets

### Treeview

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

The screenshot displays the MME Preparation Environment interface. The PUS Model Navigator on the left shows a tree view of Application Processes, with 'Service Report 4.2 [Service Report (4.2)]' selected. The Editor Area in the center shows the configuration for 'APID\_10\_4\_2', including a table of headers and a data field table. The Service Report 4.2 configuration panel on the right shows the 'Service Sub Type' set to 2 and a table of parameters.

**Header Table:**

Name	Description	Offset	Type	Data Type	Value
CCSDS_TM_Primary_Header	CCSDS packet primary header for TM	0			
Packet version number	CCSDS space packet identifier (=0)	0		Bit Stream	
TM Packet type	TM = 0, TC = 1	0		Unsigned Integer	
Secondary header flag	Has secondary header	0	Discriminant	Boolean	
Application process ID	On-board application process identifier	0	Discriminant	Unsigned Integer	
Sequence flags	Stand-alone PUS packets (=3)	0		Bit Stream	
Packet sequence count	Wrapped counter incremented by TM source	0		Unsigned Integer	
Packet data length	Data Field length set by TM source	0		Unsigned Integer	
PUS_TM_Secondary_Header	PUS secondary header for TM	0			
PUS version number	Version of PUS standard	0		Unsigned Integer	
Spacecraft time reference status	See E-ST-70-41C	0		Bit Stream	
Service type ID	PUS service type	0	Discriminant	Unsigned Integer	
Message subtype ID	PUS service message subtype	0	Discriminant	Unsigned Integer	
Message type counter	Destination-specific message counter	0		Unsigned Integer	
Destination ID	Application process addressed by report	0		Unsigned Integer	
Time Reference	Packet time stamp	0		Absolute CUC Time	

**Data Field Table:**

Name	Description	Offset	Type	Data Type	Value (hex)	PID (hex)	Length	Position
startTime		0		Unsigned Integer			16	0
endTime		0		Unsigned Integer			16	16
n		0		Unsigned Integer	1 (1)		0	32
parameterStatisticsNotificationNotificati		0		Unsigned Integer			(64)	
parameterId		0		Unsigned Integer			8	32
maximumValue		0		Unsigned Integer			8	32
maximumTime		0		Unsigned Integer			16	40
minimumValue		0		Unsigned Integer			8	56

**Service Sub Type: 2**

**PUS Service Structure:**

Name	Description	Offset	Data Type	Length	Position
startTime		0	Unsigned Integer	16	0
endTime		0	Unsigned Integer	16	16
n		0	Unsigned Integer	0	32
parameterStatisticsNotificationNotificati		0	Unsigned Integer	(64)	
parameterId		0	Unsigned Integer	8	32
maximumValue		0	Unsigned Integer	8	32
maximumTime		0	Unsigned Integer	16	40
minimumValue		0	Unsigned Integer	8	56
standardDeviation		0	Unsigned Integer	8	88

The interface also shows a console at the bottom with 6 errors and 17 warnings, and a table with columns: Description, Element, Path, Location, Check, Groovy Rule Id, Type.

# 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

The screenshot displays the PUS Model Navigator and Editor Area. The Navigator shows a tree view of the GAIA [Controlled System Root] structure, including Events, CDU [Monitoring Control Element], Calibrations, and Telemetry Packets. The Editor Area shows the configuration for APID\_161\_1\_1, including Name, Short Description, Long Description, Header, and Data Field tables.

**Header Table:**

Name	Description	Offset	Type	Data Type	Value (hex)	PID (hex)
CCSDS_TM_Primary_Header	CCSDS packet primary header for TM	0				
PUS_TM_Secondary_Header	PUS secondary header for TM	0				

**Data Field Table:**

Name	Description	Offset	Type	Data Type	Value (hex)	PID (hex)	Length	Position
------	-------------	--------	------	-----------	-------------	-----------	--------	----------

**Error View:**

10 errors, 20 warnings, 0 others

Description	Element	Path
MME: The feature 'inneElements' of 'Container' with 0 values must have at least 1 values.	APID_161_1_1	GAIA/CDU

# 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 Blommesteijn"
5 testObjectives "Start from cold or warm an
6 testConfiguration "SUT serial number ABC12
7 {
8
9 // Procedure body starts here
10
11 // Requirements (from requirements databa
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 Ia Off
27 ExecuteActivity ZAC03005 arguments PDW90011="ENABLE" // AACS 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" // AACS Set TM HK interval
43 Synchronise
44 }
45

```

Test requirements to Step mapping

Boolean Groovy expression

TCs sent asynchronously

'interval' variable used as argument



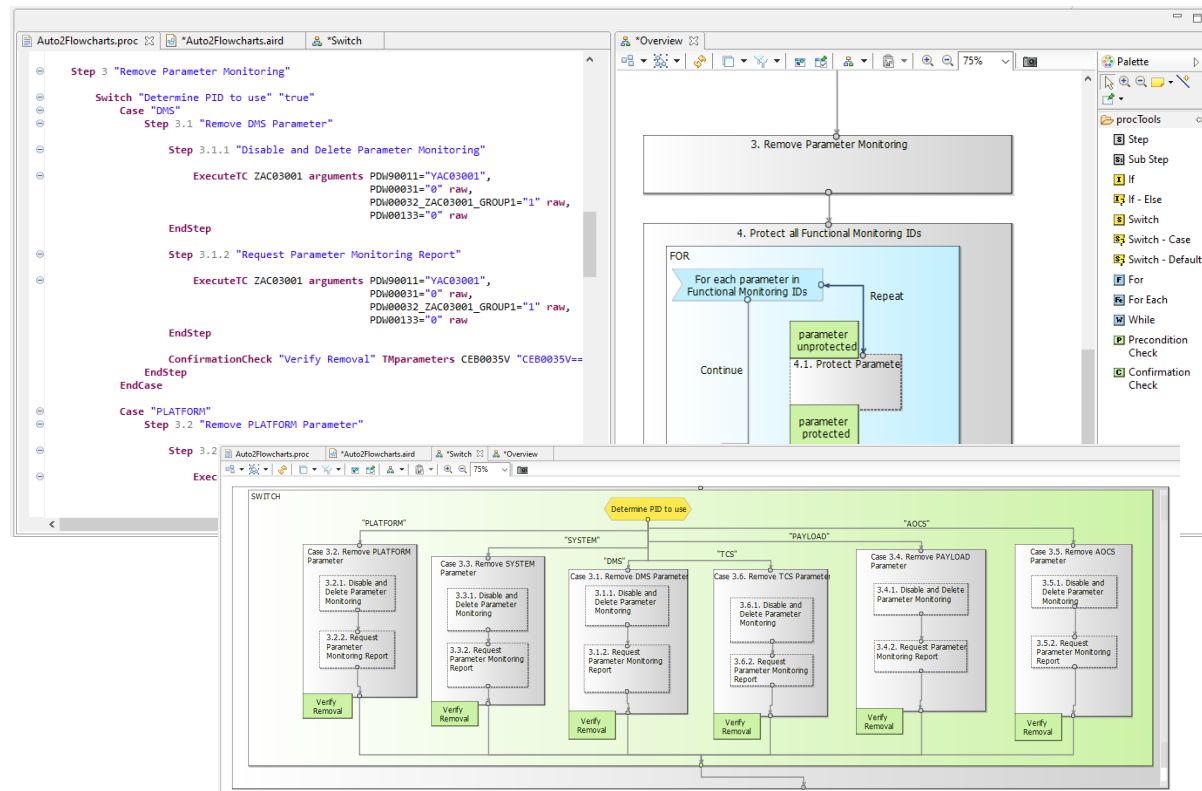
# PME Procedure Model Flowcharter

## 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

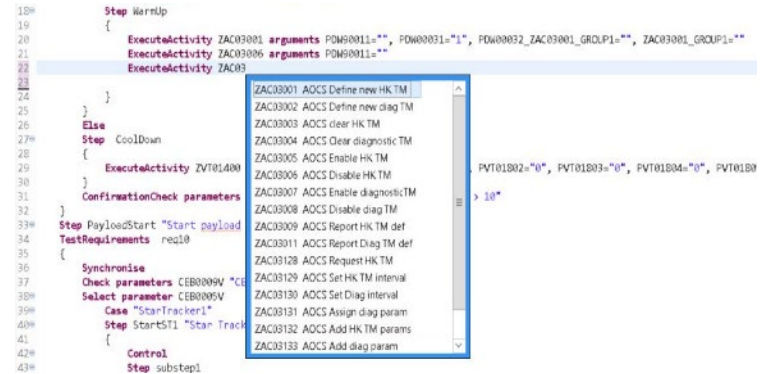
### Features

- 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.

```

18# Step WarmUp
19 {
20     ExecuteActivity ZAC03001 arguments POW0011="", POW00031="1", POW00032_ZAC03001_GROUP1="", ZAC03001_GROUP1=""
21     ExecuteActivity ZAC03005 arguments POW0011=""
22     ExecuteActivity ZAC03
23
24 }
25 }
26 Else
27 Step CoolDown
28 {
29     ExecuteActivity ZVT01400
30 }
31 ConfirmationCheck parameters
32 }
33# Step PayloadStart "Start payload
34 TestRequirements req10
35 {
36     Synchronise
37     Check parameters CEB0009V "CE
38     Select parameter CEB0005V
39     Case "StarTracker1"
40     Step StartS71 "Star Track
41     {
42     Control
43     Step substep1

```



The screenshot shows a code editor with a procedure model. A dropdown menu is open, listing activities such as ZAC03001, ZAC03002, ZAC03003, ZAC03004, ZAC03005, ZAC03006, ZAC03007, ZAC03008, ZAC03009, ZAC03010, ZAC03011, ZAC03012, ZAC03013, ZAC03129, ZAC03130, ZAC03131, ZAC03132, and ZAC03133. The dropdown is positioned over the code, and the selected activity is ZAC03001. The code is syntax-highlighted, and the editor shows line numbers on the left side.

# 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)

```

1 Procedure StartupTest
2 author 'Anon E Maus'
3 testObjectives 'Start from cold or warm and ensure all instruments are in their nominal states'
4 testConfiguration 'SUT serial number ABC1234'
5 {
6   Step Initialisation 'Initial boot sequence'
7   {
8     // Check if system is in expected state
9     If parameters NPF11119, NPA00017, CEB0033 'NPF11119 + NPA00017 > CEB0033 + 1'
10    Step StartupSequence 'Start platform tasks'
11    {
12      ExecuteActivity ZAC03002 arguments PDW90011='ENABLE', PDW00031='678'
13      ExecuteActivity ZAC0000 arguments TIMEOUT='210'
14      Synchronise
15
16      Log 'Execute other statements'
17
18      ExecuteActivity ZACZZZ
19      Synchronise
20    }
21  }
22 }
  
```



```

import java.util.concurrent.Future
import java.util.concurrent.Executors

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) {

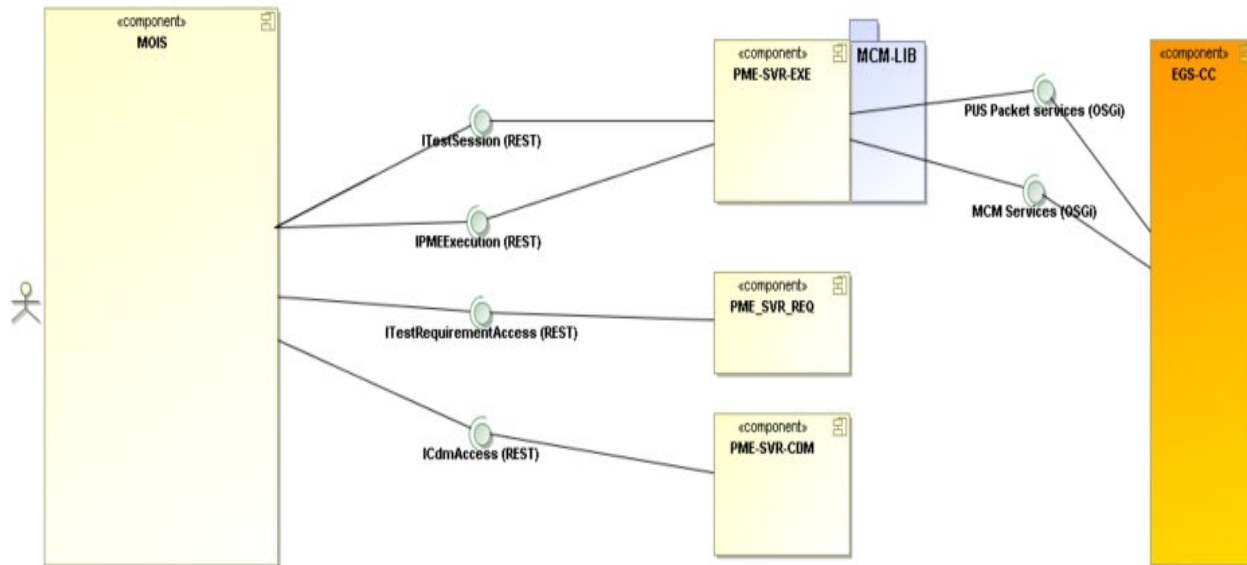
    println 'StartupSequence : Start platform tasks'

    // Initiate Activities asynchronously
    sendtcc('ZAC03002', "ENABLE", "678")
    sendtcc('ZAC0000', "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

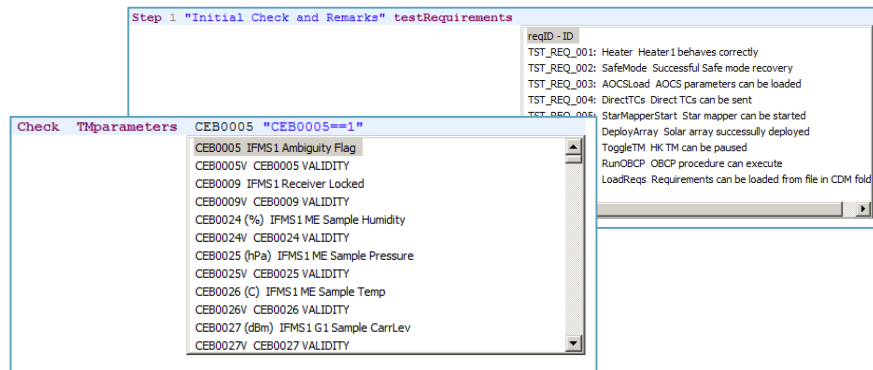
- Test Requirements
- Mission Model

## Procedure Editing Services

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

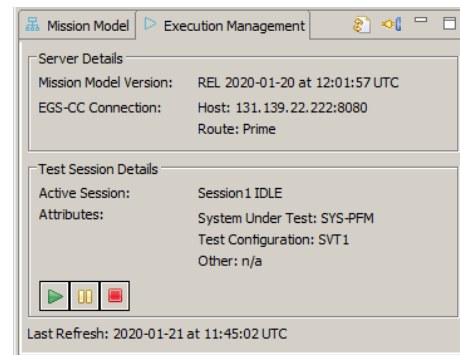
## Execution Services

- 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 Operations Information System** is the solution for missions preparation & manages the mission lifecycle

Used for over 100 spacecraft by

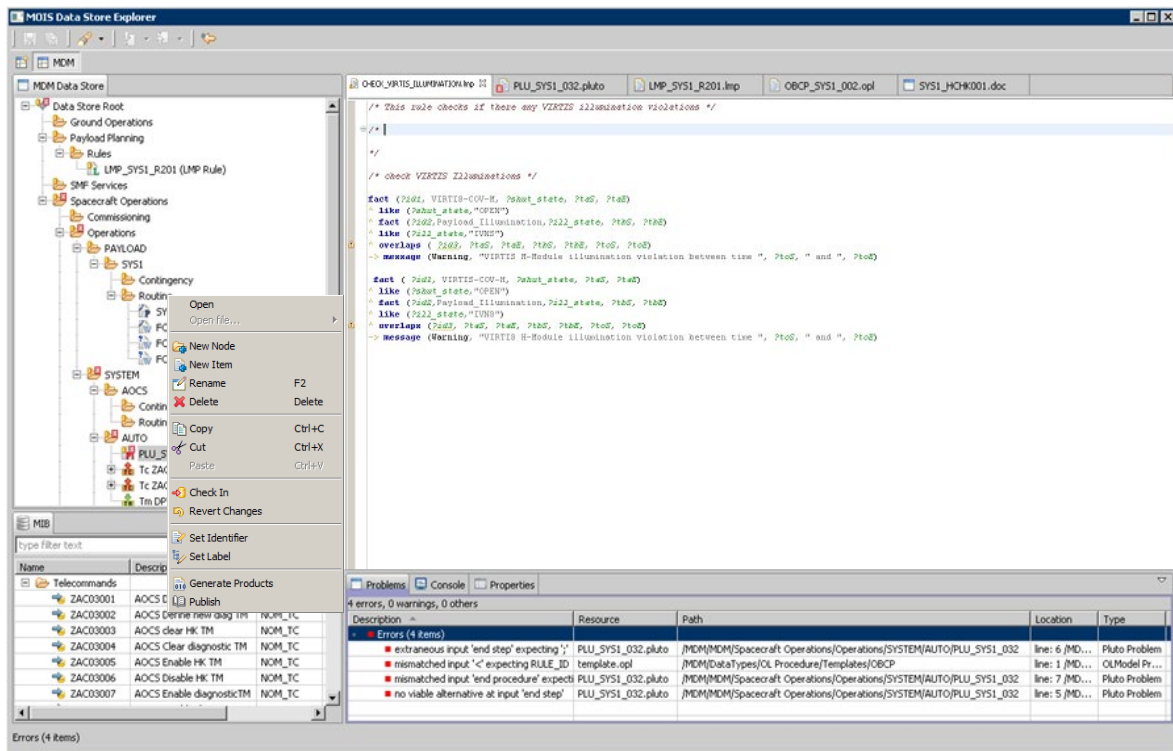
- ESA
- SpaceOpal (Galileo)
- Airbus-DS
- DLR
- ASI
- TAS-I
- TESAT
- Astroscale
- GMV
- ...and many more

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

The screenshot displays the Mission Model Execution interface. At the top, a 'Mission Model' window shows 'Server Details' (Mission Model Version: Issue 2.1, Requirements Version: DOORS1.1, EGS-CC Connection: Host: Local) and 'Test Session Details' (Active Session: Session1, Attributes: Session Type: LOCAL, System Under Test: AOCs, Test Configuration: SVT0, Other: n/a). Below this is a table with columns: Date, Session, Procedure, Step (confirmation), and Message type. The table contains several rows of test execution data, including a failure at 2020/02/25 15:06:43.793 and a finished procedure at 2020/02/25 15:08:33.132. To the right of the table, a log window shows messages such as 'Getting parameter CEB0024 = 9.1', 'Sending TC ZAC03004 with args: [PDW90011 = YAC0...', and 'Session Session1 ended'. At the bottom, two windows show 'Procedure Summary' and 'Start of Procedure' logs, detailing test steps and configuration checks.



# 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

The screenshot displays the MOIS Data Store Explorer interface. On the left, a treeview shows the project structure under 'Data Store Root', including folders like '\_CDM', '\_REQ', 'Main', and 'EGSCC'. A 'Property Search' panel is visible below the treeview. The main area shows the 'RoundTrip.proc' procedure code, with a callout 'Context Execute/Debug' pointing to the 'Execute' button in the treeview. The code includes comments and logic for sending TC, getting counters, and waiting for TM effects. A 'Logging' callout points to the 'Console' tab at the bottom, which displays a log table.

Date	Session	User	Procedure	Step	Type	Message
2020/04/30 09:01:03.987	EGSCC				PME	Getting parameter SBV_param_int_1 = 61.0
2020/04/30 09:01:06.095	EGSCC				PME	Getting parameter SBV_param_int_1 = 61.0
2020/04/30 09:01:08.224	EGSCC				PME	Getting parameter SBV_param_int_1 = 61.0
2020/04/30 09:01:10.363	EGSCC				PME	Getting parameter SBV_param_int_1 = 62.0
2020/04/30 09:01:10.504	EGSCC				PME	Getting parameter SBV_param_int_1 = 62.0
2020/04/30 09:01:10.530	EGSCC	spearson	RoundTrip-20	1	Step confirmation	(SBV_param_int_1 == count + 1) passed
2020/04/30 09:01:10.537	EGSCC	spearson	RoundTrip-20	1 (passed)	Finished step	
2020/04/30 09:01:10.544	EGSCC	spearson	RoundTrip-20		Finished procedure	

# 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

The screenshot displays the MOIS Data Store Explorer application. The interface is divided into several panes:

- Left Pane:** A tree view showing the project structure under 'MOIS Data Store'. A blue callout box labeled 'Toolbar' points to the toolbar area above the tree view.
- Code Editor:** Displays the source code for 'StartupSequence2.proc'. A blue callout box labeled 'Currently executing line' points to the line: `If Tparameters NPF11119,NPA00017,CEB0033 (dbm) "NPF11119 + NPA00017 < CEB0033 + 1"`. Another blue callout box labeled 'Breakpoint' points to a red dot on the left margin of the code editor.
- Bottom Pane:** A 'Debug Console' showing the execution status. It indicates that the thread is suspended at a breakpoint on line 16 of 'StartupSequence2.proc'. The console output lists various method calls and their return values.

# 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

The screenshot displays the PME Procedure Controller web client interface. At the top, there are navigation buttons: Launch, Monitor, Launch & Monitor, and Log. The interface is divided into two main sections: '19 Available Procedures' and '4 Monitored Procedures'.

**19 Available Procedures:**

Procedure	Action
CallProc	Execute
p2	Execute
ManyParams	Execute
StarMapper_T1	Execute
ForeverLoop	Execute
OnePacket	Execute
test1	Execute
RoundTrip	Execute

**4 Monitored Procedures:**

Procedure	User	Start Time	End time	Step	State	Clear All
ForeverLoop	tomcat	2021/10/20 14:21:38		1.1	RUNNING	Pause Stop Abort
OnePacket	tomcat	2021/10/20 14:19:36	2021/10/20 14:19:42	1	FAILED	Clear
StarMapper_T1	tomcat	2021/10/20 14:19:25	2021/10/20 14:19:30		COMPLETED	Clear
ForeverLoop	tomcat	2021/10/20 14:18:06	2021/10/20 14:19:09	1.1	STOPPED	Clear

**Log Execution:**

Timestamp	User	Test	Step	Step Name	Description
2021/10/22 08:57:12	spearson	Test6	2.1.1	FINISH_STEP	
2021/10/22 08:57:12	spearson	Test6	2.1.2	START_STEP	Description
2021/10/22 08:57:12	spearson	Test6	2.1.2	PME	Obtained parameter CEB0009 = 5.1, timestamp 08:57:12.476
2021/10/22 08:57:12	spearson	Test6	2.1.2	IF	(CEB0009 > 2) is true
2021/10/22 08:57:12	spearson	Test6	2.1.2.1	START_STEP	
2021/10/22 08:57:12	spearson	Test6	2.1.2.1	PME	Obtained parameter CEB0009 = 6.1, timestamp 08:57:12.496
2021/10/22 08:57:12	spearson	Test6	2.1.2.1	SWITCH	(CEB0009) is 6.1
2021/10/22 08:57:12	spearson	Test6	2.1.2.1	FINISH_STEP	
2021/10/22 08:57:12	spearson	Test6	2.1.2	FINISH_STEP	
2021/10/22 08:57:12	spearson	Test6	2.1	FINISH_STEP	
2021/10/22 08:57:12	spearson	Test6	2	FINISH_STEP	
2021/10/22 08:57:12	spearson	Test6		FINISH_PROC	

Set Active Session:  Autoscroll  Show Full Log

Updated: 2021/10/22 09:52:19



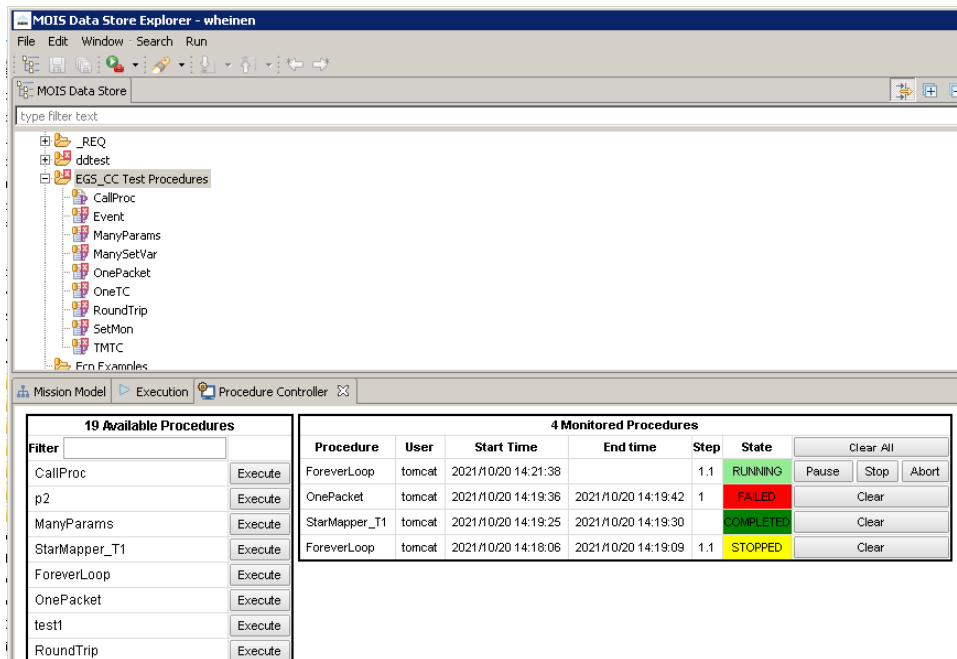
# PME-Controller Web Client

## Datstore 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

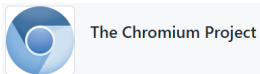
## Execution REST API

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



The screenshot shows the MOIS Data Store Explorer interface. The top part displays a tree view of the data store structure, including folders like \_REQ, ddttest, and EGS\_CC Test Procedures. The bottom part shows a table of monitored procedures with columns for Procedure, User, Start Time, End Time, Step, and State. The table also includes buttons for Clear All, Pause, Stop, Abort, and Clear.

19 Available Procedures		4 Monitored Procedures										
Filter		Procedure	User	Start Time	End Time	Step	State					
	CallProc	Execute	ForeverLoop	tomcat	2021/10/20 14:21:38		1.1	RUNNING	Clear All	Pause	Stop	Abort
	p2	Execute	OnePacket	tomcat	2021/10/20 14:19:36	2021/10/20 14:19:42	1	FAILED	Clear			
	ManyParams	Execute	StarMapper_T1	tomcat	2021/10/20 14:19:25	2021/10/20 14:19:30		COMPLETED	Clear			
	StarMapper_T1	Execute	ForeverLoop	tomcat	2021/10/20 14:18:06	2021/10/20 14:19:09	1.1	STOPPED	Clear			
	ForeverLoop	Execute										
	OnePacket	Execute										
	test1	Execute										
	RoundTrip	Execute										



# 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**

