18th International Conference on Space Operations, Montreal, Canada, 26 - 30 May 2025.

SpaceOps-2025, ID # 358

Automated Monitoring and Control Test and Operations Procedure Management and Execution - Concepts and Implementation

Sven Steininger[a], Dirk Peters[a], Pierre Denis[b], Philipp Hamacher[c], Tiago Simoes[d1], François Trifin[d2]

[a] *SpaceCube GmbH, Berliner Allee 47, 64295 Darmstadt, Germany*

[b] *Spacebel S.A., Rue des Chasseurs Ardennais 6, Liege science park, 4031 Angleur, Belgium*

[c] *Deutsches Zentrum für Luft- und Raumfahrt e.V. /GSOC, Münchener Straße 20, 82234 Weßling, Germany*

[d] *European Space Agency / [1] ESOC, Robert-Bosch-Str. 5, 64293 Darmstadt, Germany / [2] ESTEC, Keplerlaan 1, 2201 AZ Noordwijk, Netherlands*

Abstract

Managing monitoring and control procedures used for assembly, integration and testing and Ground and Flight control, including the development, verification, validation, and maintenance of those procedures, is a labor-intensive set of tasks requiring highly skilled personnel to handle mission-critical elements such as the pre-launch verification and validation of the Space segment components, launch and early orbit phase, spacecraft failures, degradation of spacecraft functions, hardware damages, recovery actions, and the avoidance of mission loss. It is therefore key that these tasks are supported by adequate authoring and executions concepts and tools.

Due to the evolution of their mission control systems infrastructure, the European Space Operations Centre (ESA/ESOC) and the German Space Operations Center (DLR/GSOC) require new Operations procedure management data systems adapted to the latest technological and user requirements. Acknowledging the common operational working practices and processes and the extent of the effort required, both control centres combined their effort to develop a joined Procedure Management Environment and took the opportunity to re-think how higher efficiency can be achieved and to consolidate procedure management concepts based on the previous systems and the lessons learnt.

Building upon the presentation of the Operations procedures management operations & system concepts of the joined ESA-DLR Procedure Management Environment at SpaceOps 2023, this paper introduces and describes the underlying Automated Test and Operations Procedure language (ATOP) core principles and the various end-to-end workflows for procedure development, verification, validation, execution and monitoring. The paper further clarifies the current and upcoming technical capabilities.

The procedure management data systems and concepts presented are planned to be used by all future Space missions operated at ESOC and GSOC as foreseen by the respective control centres strategic migration to their EGS-CC based infrastructure data systems EGOS-CC and GX-CC.

Keywords: Operations Mission Procedure Management Environment

1 Introduction

In today's fast evolving world of Space, Ground operations must scale quickly and the automation of operational procedures has become a key contributor to this. As missions grow in complexity and ambition, the need for efficient, reliable, and scalable solutions to manage these operations is paramount. Automation offers a transformative approach to addressing these key challenges, enhancing the precision and consistency of mission tasks, in particular routine ones, while reducing the risk of human error.

The importance of automation in Space operations is underscored by several key factors. Firstly, it enables the handling of large volumes of data and the execution of repetitive tasks with high accuracy, which is essential for the

success of operations, in particular for large scale missions. Secondly, automation facilitates real-time decision-making and rapid response to unforeseen events, thereby improving the overall resilience and adaptability of Space missions. Lastly, by automating routine procedures, human operators can focus on more strategic and innovative aspects of mission planning and execution, ultimately driving advancements in Space exploration and technology.

However, automation faces some difficult barriers for adoption, from human mistrust to incompatibility with outdated Ground systems. Another of these barriers is the heritage of how operational procedures are written. For decades, operational procedures have been written by humans for humans, meaning that procedures widely vary in their format/form and often do not contain the full information required for its execution, instead relying on the assumed knowledge of the reader which will execute the procedure. This puts the role of a human at the centre of any activity related to procedures, regardless of it being a testing campaign to verify spacecraft system integration, to the controlling of a spacecraft in flight, even if the procedure being executed is essentially the same. To remove these barriers, focus is being put in automation languages, frameworks and tools that allow the creation of operational procedures in an explicit, systematic and comprehensive way, eliminating the need of human interpretation for successful execution. Based on these explicit operational procedures, Ground operation systems can be created/adapted to allow their direct execution without reliance on translation layers assuming how a human would understand the procedure contents.

This paper covers the efforts of ESA, DLR and European industry with respect to automation for Ground operation systems, from the Automated Test and Operations Procedure (ATOP) language that is being specified to define operational procedures, over the Operations Preparation Environment for Missions (OPEN-M) as the tool that users can use to create and manage their ATOP procedures, to the ATOP execution engine in the European Ground System - Common Core (EGS-CC) Automation Component which allows the execution of the ATOP procedures in Ground control systems based on EGS-CC. A focus on a vertical approach is intended, covering most mission phases, from the initial spacecraft integration and testing to the final operation of a flying spacecraft. Readers will be introduced to these concepts, tools and frameworks as well as the associated activities used to automate Ground operations, from definition to execution.

2 Related Work and Background

EGS-CC is the underlying system of the next generation Central Check Systems (CCS) and Mission Control Systems (MCS) supported by European Large System Integrators and Agencies [7]. This system Reference Implementation allows the use of two Domain Specific Languages (DSL) for the definition of end user automated procedures:

- EGS-CC Automation Procedure Language (EAPL) and
- Automated Test and Operation Procedure (ATOP) language.

The automation component, provided by the Reference Implementation layer of EGS-CC, includes the capability to natively execute both EAPL and ATOP. EAPL, introduced at the start of the EGS-CC project, is an internal DSL built on top of the Java general purpose language. It leverages conventions and specific Java APIs to allow a user-defined Java class to be executed as a procedure within the automation component. ATOP, which has been added to the EGS-CC Automation component in 2024, is entirely specialised to the purpose of defining Space dedicated monitoring and control procedures for test and operations.

A procedure management environment integrated to the Operations Preparation Environment for Missions (OPEN-M) software system [8][10][11][12] has been developed by ESA and DLR and is introduced by the paper “Joined ESA-DLR Procedure Management Environment” [1]. This procedure environment has been developed by building upon the well-established systems (MATIS [4], ProToS [9], MOIS, ASE [5][6]) that have been in use for years at ESA, DLR and throughout the European Space industry. OPEN-M is planned to replace and enhance those systems, offering improved functionality and efficiency.

Furthermore, multiple research and development activities at ESA have also contributed to the development of this environment, particularly influencing the design of the ATOP language, which draws heavily from the specification of the Test and Operations Procedure language provided by the annexes of the ECSS standard ECSS-E-ST-70-32C [14].

3 Introduction to the Automated Test and Operations Procedure Language (ATOP)

The “Automated Test and Operations Procedure” language (ATOP) is a high-level procedural language used for the monitoring and control of a Space system. A Space system covers both Ground and Space segments dedicated to one of multiple spacecraft. The language allows the definition of procedures executed on the Ground, either manually or automatically. As the name suggests, it can be used for both verification, typically when the hardware is tested on Ground, and Operations, typically after a launch phase. Therefore, typical users of the language are control centre operators and assembly, integration and verification/Test (AIV/AIT) engineers.

ATOP is a DSL, meaning that the language is designed for and dedicated to be used in a specific domain, which is in this case the Monitoring and Control (M&C) domain. It is also an external DSL, meaning it does not reuse a general-purpose programming language as a host. The language therefore exclusively uses concepts and syntax relevant for its M&C purpose. The language has been designed for defining M&C procedures written by aerospace engineers, most of whom are not programmers and need unambiguous procedures fully consistent with the Space system.

ATOP differentiates itself from other adopted test and operation languages by

- its readability, due an explicit and natural language syntax, as procedures must be unambiguously understood with minimal background knowledge,
- its dedicated constructs and execution behaviours specifically designed for testing and operations,
- its capability to interact with a model of the Space system through well-defined and generic services,
- enabling the development of tools checking the full consistency between the procedure content and the Space System Model (SSM) during the preparation phase of the procedure,
- enabling the development of flowchart and tabular editors without the textual representation,
- allowing a transition path from manually interpreted procedure to fully automated procedures, as the language allow the modelling of both use cases.

ATOP is compliant with the European Cooperation for Space Standardization (ECSS) standard ECSS-E-ST-70-32C, which defines *"the capabilities of the language used for the definition of procedures for Space system testing and operations"*. Let us remark that this standard contains a non-normative annex defining the “Procedure Language for Users in Test and Operations “ (PLUTO) language [3], which can be considered as is the main ancestor of the ATOP language.

The ATOP language is formally defined by:

- the “building blocks” that constitute procedures and the role that each of these building blocks plays in achieving the overall objectives of the procedure,
- the dynamic aspects of procedures i.e. the execution logic of each building block and execution relationships between these blocks,
- the syntax (grammar rules) and semantics of the language itself.

The syntax and keywords are closer to natural language than mainstream general-purpose programming languages. For instance, all forms of calls to subsidiary activities begin with the verb "`initiate`" and composite object references use the English prepositional form—incidentally close to the French form—, e.g. "`status of sensor_14 of payload`".

ATOP language is closely coupled to the concepts of the SSM, as defined in ECSS-E-ST-70-31C [13]. While known as SSM, it is used in EGS-CC under the term Monitoring and Control Model (MCM). The concept of SSM is basically a very broad model of the whole Space system, including Flight and Ground segments, covering both hardware and software components. It is structured as a tree of *system elements*, each of which may contain *activity*, *reporting data* and *event*. These are very abstract concepts gathering several ideas. For instance, the name *activity* may refer to a Ground procedure, an On-board Control Procedure (OBCP), a telecommand, etc. An example of SSM's AOCS system element is sketched in Figure 1, excerpted from ECSS-E-ST-70-31C [13].

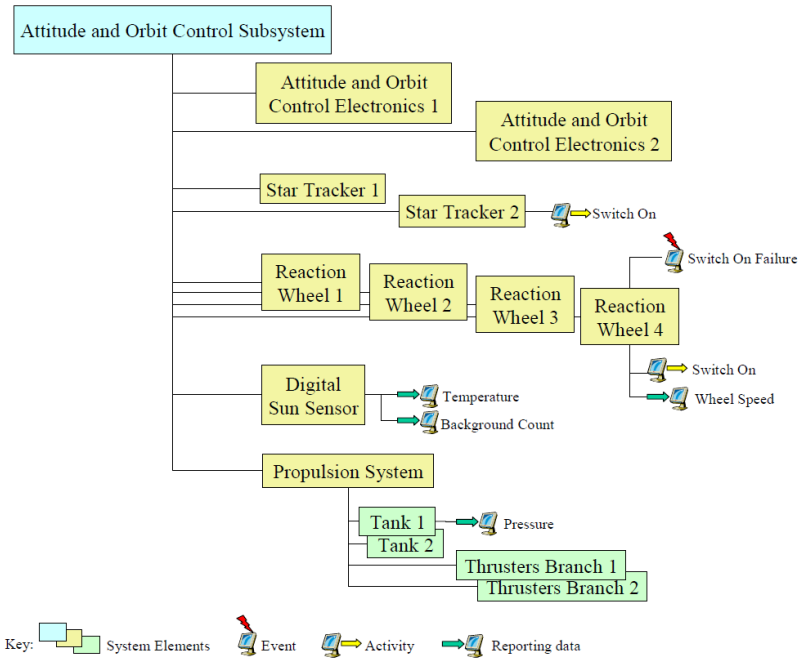

Figure 1: Example of an AOCS system element, part of a Space System Model (ECSS-E-ST-70-31C)

An ATOP procedure is an activity attached to some system element of the SSM, and which can interact with other system elements. The modes of interaction include reading a reporting data, initiating another activity, raising an event, etc.

ATOP is *goal-oriented*: it aims at reaching a given verifiable goal provided that given prerequisites are verified. For this purpose, the procedure defines one or several steps, which are meant to reach sub-goals in a similar way. Each procedure and step define *bodies*:

- a *declaration* body (optional), defining local events, variables and enumerated sets used in the step/procedure;
- a *precondition body* (optional), defining the prerequisite condition to be fulfilled before going further; clauses may be defined to define behaviour if not verified, e.g. waiting the fulfilment (with some timeout duration) or aborting the procedure/step;
- a *main body* (mandatory), defining the statements or inner steps to execute as soon as the precondition is fulfilled for reaching the assigned goal;
- a *watchdog body* (optional), monitoring the good execution of main body, taking the control a soon as one or multiple boundary execution conditions are detected, possibly recovering them, then resuming / terminating / aborting the main body or escalating to its parent as necessary;
- a *confirmation body* (optional), verifying the assigned goal by means of a post-condition; clauses allow to define what to do if the condition is not fulfilled, e.g. waiting the fulfilment (with some timeout duration), ending in failure or restarting the procedure/step, from the precondition body.

These concepts are illustrated in the following figure, excerpted from ECSS-E-ST-70-32C [14] ("Sub-goals" are referenced as “steps” within the language).

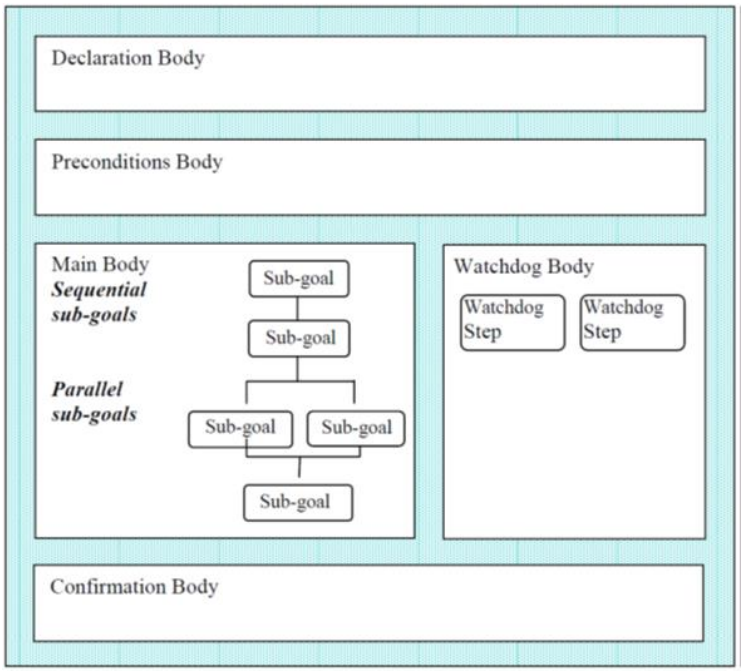

Figure 2: Structure of an ATOP procedure or step (ECSS-E-ST-70-32C)

ATOP follows the approach of statically-typed languages: all variables used shall be declared in the declaration body, with a name and a type; these are visible in all sibling bodies and within their inner steps, if any. ATOP's types include usual data types: integer, real, Boolean, enumerated, relative/absolute time. In the declaration body, one can also declare *local events* that can be raised in other bodies in case of detected contingency, for transferring the control to a watchdog step.

The action to take at the end of any step is controlled in a systematic way by means of *continuation test*. A continuation test is an optional construct that define what to do, namely the *continuation action* (CA), depending on the *confirmation status* (CS) of the step, which is:

- "confirmed": if the confirmation body has been fulfilled
- "not confirmed": the confirmation body has not been fulfilled
- "aborted": execution has been aborted (before execution of confirmation body)

The continuation actions are "terminate", "abort", "resume", "raise event", "continue", as well as "ask user" for letting the operator decide which CA shall be undertaken.

The above description of ATOP gives a taste of the dedicated constructs and execution behaviours specifically designed for testing and operations. Many more are specified by the language.

4 Concepts and Implementation

As mentioned in the previous sections, ATOP is a language specification. To leverage ATOP effectively in an operational setting, appropriate tooling must be developed to comply with the established specification and standards. This tooling acts as an intermediary, ensuring integration between the ATOP-based procedures and the broader operational software systems. By adhering to the prescribed framework, these tools enable efficient consistency checking, validation, execution, and management of test and operations procedures.

Two main areas are:

- Preparation – This phase encompasses the authoring and refinement of ATOP procedures. Procedure authors create, edit and verify these procedures within an integrated development environment (IDE)

equipped with specialized functionalities. The IDE facilitates the authoring process through syntax highlighting, consistency checking against the Space System Model, and semantic verification, ensuring compliance with operational constraints. Additionally, ATOP procedures are integrated with tailoring data, that are mission- or system-specific elements that adapt the generic procedures to particular operational contexts. This integration ensures that procedures remain flexible and reusable across different missions and operational scenarios.

- Automation – Once an ATOP procedure has been authored and verified, it undergoes validation and execution. The ATOP execution engine plays a crucial role in this process by compiling the DSL script into a format interpretable by the MCS. This compiled output enables direct interaction with the system, allowing for automated and highly controlled operations. The execution engine ensures consistency, traceability, and correctness of operations by following predefined workflows and verification steps. By automating these processes, ATOP reduces the risk of human error and increases operational efficiency.

4.1 Preparations: Authoring and Managing ATOP Procedures

The preparation phase of ATOP procedures involves a comprehensive environment for authoring, verifying, managing, and refining procedures. This is facilitated through the OPEN-M system [8], which extends beyond procedure management to encompass the full spectrum of EGS-CC tailoring data. Since Flight Control Procedures are inherently part of the EGS-CC tailoring data model (aka MCM, aka SSM), OPEN-M ensures consistency across all mission-specific configurations, including telemetry, telecommands, monitoring activities, and operational constraints. At its core, OPEN-M is built on the OPEN Framework, a shared software infrastructure that supports both Ground station tailoring (OPEN-S) and mission tailoring (OPEN-M). While both environments share fundamental functionalities, each is tailored to its specific domain. OPEN-M provides a suite of desktop and Web-based applications that work together, facilitating collaborative authoring, version control, consistency checking, and automated workflow management.

The procedure development environment within OPEN-M is centred around the ATOP procedure model, an intermediate representation that serves as the authoritative data structure for defining procedures. Rather than relying on a single editing approach, OPEN-M provides multiple views of the ATOP procedure model through the Multi-Page Editor, which enables users to seamlessly switch between different visual representations of a procedure. These representations cater to various user needs, ensuring accessibility for both technical and non-technical stakeholders.

Each of these editors directly modifies the ATOP procedure model, ensuring that the procedure remains consistent across different views, meaning changes in one editor are immediately reflected in the others without compromising data integrity. By supporting interoperability between these editors, OPEN-M ensures that users can switch between views at any time, choosing the most suitable perspective for their task. Whether writing precise script-based procedures, structuring high-level operational steps in a table, or visualizing execution flow, the ATOP procedure model remains synchronized and consistent across all representations.

4.1.1 Procedure Textual Editor

The Procedure Textual Editor serves as the primary authoring interface for ATOP procedures, providing an integrated development environment (IDE)-like experience tailored for procedure creation. It features real-time syntax verification, providing immediate visual feedback on potential errors and inconsistencies. Syntax errors are dynamically highlighted, accompanied by detailed error messages that guide users toward correct procedure formulation. This proactive approach helps identify procedural issues early, minimizing the risk of runtime errors. Advanced code navigation and refactoring capabilities allow users to quickly move between procedure sections, with features like outline view and cross-reference tracking. Semantic highlighting differentiates between various ATOP language elements – declaration bodies, precondition blocks, main execution bodies, and confirmation sections – enhancing readability and comprehension of complex procedures.

The tight integration with the OPEN-M environment enables verification against the mission's data model, ensuring that all referenced MCM system elements, activities, and parameters are correctly configured. Additional features, including code folding, hover-based information displays, and comprehensive search functionality, support an efficient editing process. Designed to go beyond basic text input, the Procedure Textual Editor provides a specialized authoring

environment that aligns with the ATOP language's objective of facilitating clear, systematic, and comprehensive operational procedure development.

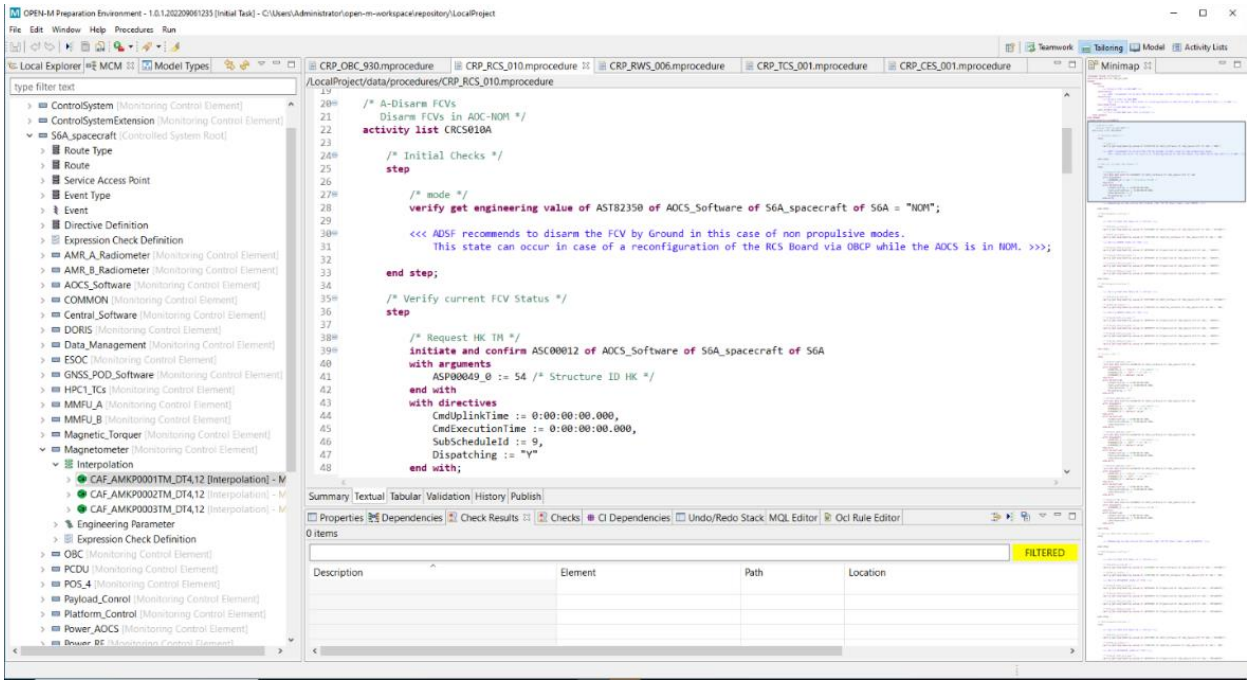

Figure 3: OPEN-M Procedure Textual Editor

4.1.2 Procedure Tabular Editor

The Procedure Tabular Editor (PTE) offers a comprehensive high-level overview of the procedure. It is designed to support all statements defined within the procedure language. To facilitate differentiation between statement types, each of those is uniquely rendered, featuring distinct colorization and formatting of text as illustrated in Figure 4. Beyond simply displaying elements in sequential order, the PTE organizes them into an expandable tree structure, enhancing both navigation and content awareness. No knowledge about the underlying ATOP procedure language is required, making it ideal for individuals who prefer a graphical interface or those looking to deepen their understanding of the script language by toggling between the graphical and script views.

The editor streamlines development by supporting standard operations such as copy-and-paste and drag-and-drop for individual or multiple statements, accelerating procedure development by facilitating the reuse of elements from existing procedures in new ones.

Each statement type can be modified via dedicated property dialogs, allowing users to create or edit content through a user-friendly interface tailored to each statement type. The PTE interfaces with the tailoring data to display relevant elements or to provide details about the selected element, such as data type or range sets. Additionally, it performs consistency checks on input values in relation to the MCM.

Special emphasis has been placed on handling ATOP Expressions, a fundamental component of the procedure language used for setting values or evaluating conditions. Given that expressions can be nested or concatenated, the editor offers a flexible interface for visualization and editing of all possible configurations. This is achieved through a dual approach: displaying portions of the procedure syntax as text and allowing for in-text editing within a specialized graphical interface. Figure 5 demonstrates this integrated approach of textual and graphical editing.

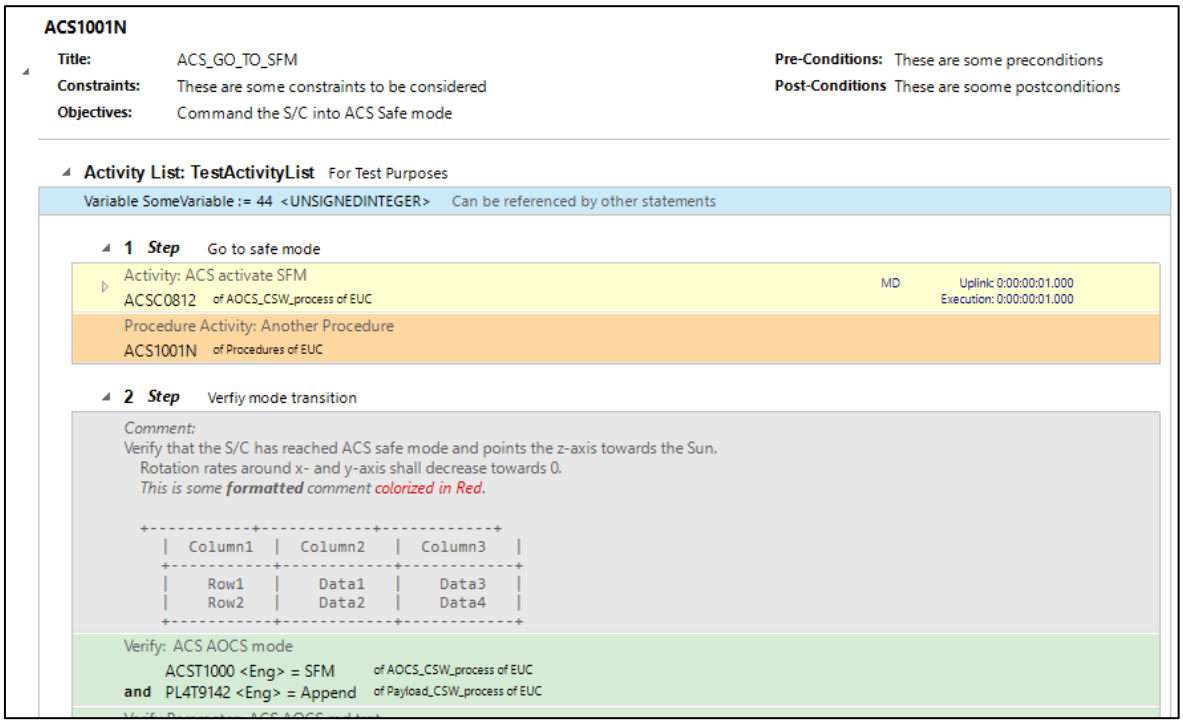

Figure 4: OPEN-M Procedure Tabular Editor

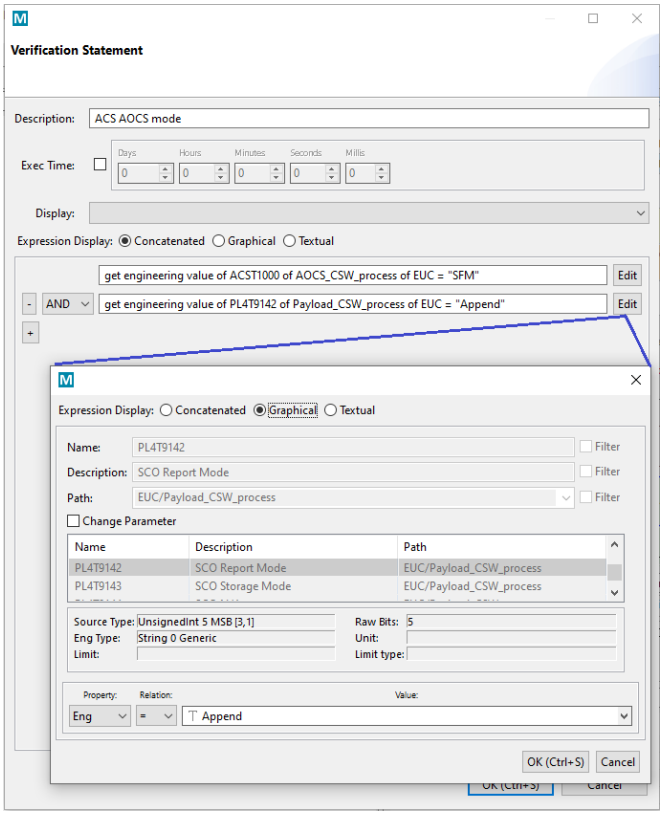

Figure 5: OPEN-M Expression Editor

4.1.3 Flowchart Editor

The flowchart editor is a specialized software tool designed to streamline the creation, visualization, and management of procedural workflows. It accomplishes this by offering a graphical interface where users can construct workflows through the arrangement of steps, conditions and connections. Each node within the editor represents a concrete statement of the procedure model that can be created, rearranged, altered or deleted.

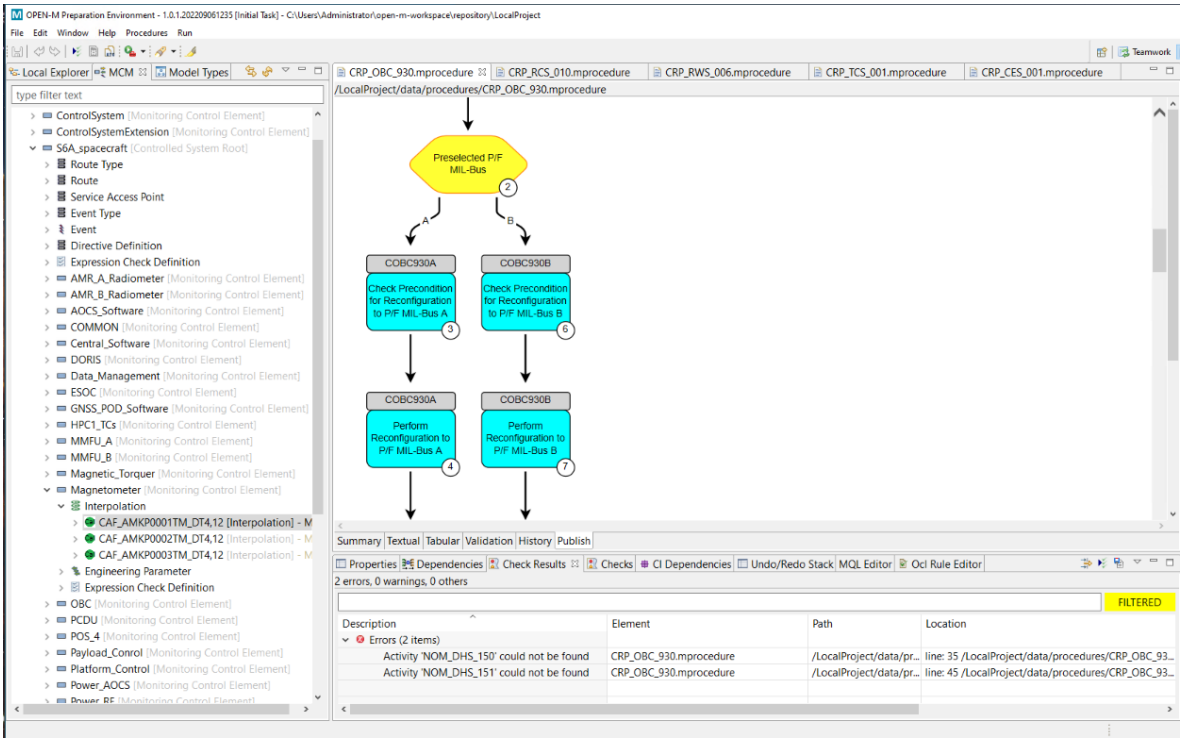

Figure 6: OPEN-M Flowchart Editor

One of the key features is the graphical representation, which allows users to create workflows visually. Steps are the building blocks of these workflows, and users can easily add, edit, and connect them to define the sequence of operations. Moreover, the editor facilitates data flow between nodes, making it easy to define how procedural flow is routed between different steps within the workflow. This aids in understanding the procedural logic and dependencies. A critical advantage is the error prevention. The editor provides only the operations that transform a valid workflow into another valid workflow. This prevents issues such as cyclic dependencies or invalid connections, enabling users to focus on their work instead of resolving issues. Additionally, the editor directly edits the in-memory representation of the procedure model directly from the graphical interface, providing immediate feedback and validation in the textual editor if used in side-by-side mode.

The flowchart editor offers several notable key features: Firstly, it enhances visual clarity by providing a graphical representation of workflows, making complex procedures more understandable and reducing the risk of misinterpretation. Secondly, it encourages modularity and reusability through node-based design, allowing for the creation of modular workflows that can be easily reused and adapted. The visual nature also makes procedural workflows accessible to a broader audience, including individuals who may not be familiar with coding languages, such as domain experts. Collaboration is facilitated, as team members can work together on a visual representation of the procedure, fostering a shared understanding and easing communication. Finally, procedural flowchart editors improve efficiency by streamlining the design and debugging processes, ultimately reducing development time and enhancing productivity.

4.2 Automation: Execution of ATOP Procedures

Automation plays a fundamental role in space operations, ensuring repeatable, reliable, and autonomous ground execution of procedures. It is critical for both the AIV phase and mission operations, enabling Space and Ground Segments to function with minimal manual intervention. At its core, automation execution is built on a structured procedure model that integrates with the MCS. It ensures that automation procedures are prepared, exchanged, and executed in a standardized and predictable manner.

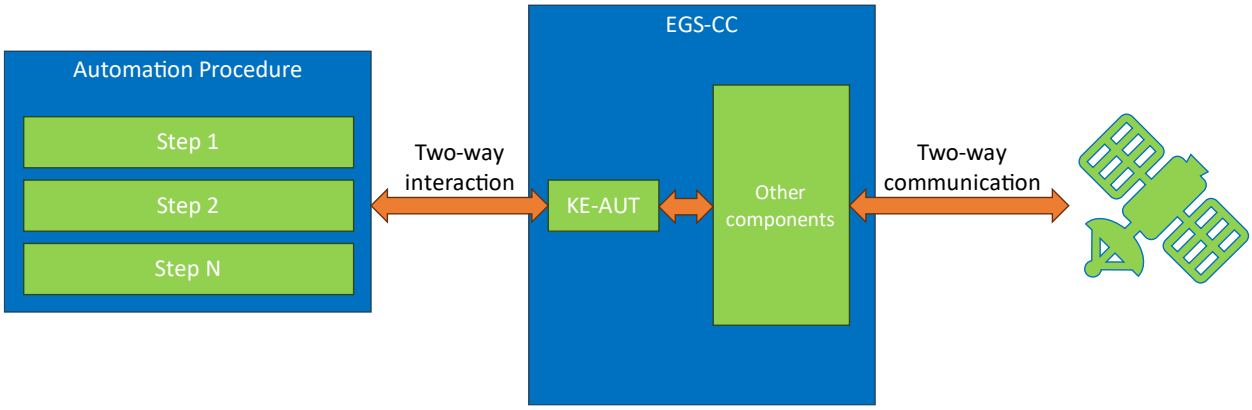

Figure 7: EGS-CC – Automation Procedure Execution

4.2.1 Modular Execution Framework

The EGS-CC Kernel Automation component (KE-AUT) is responsible for procedure execution. Designed with modularity and extensibility as core principles. It provides a flexible framework that supports multiple automation languages while maintaining a consistent and robust execution environment.

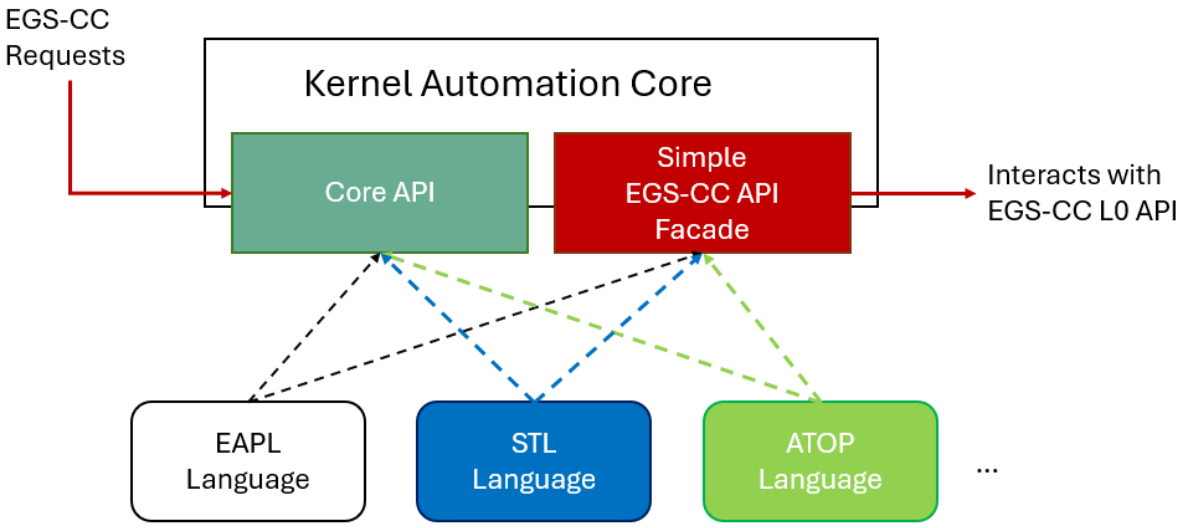

Figure 8: KE-AUT – Architectural Overview

The architecture of KE-AUT (see Figure 8) is built around a modular core containing essential EGS-CC service implementations. This core serves as the primary entry point for procedure execution, offering a standardized mechanism for processing and managing operational procedures across different language implementations. A key feature is the Automation Procedure Adaptation API, which provides an interface for various programming languages

to interact with core system functionality. This abstraction allows different language implementations to be integrated as "plugins," each encapsulated within its own bundle. The approach supports both syntactically compatible languages, like EAPL (which follows Java/Groovy conventions), and more specialized/custom languages like ATOP and the Station Tailoring Language (STL) (which is not part of the Reference Implementation) that may have unique syntactical requirements. The modular design ensures that each language extension exists as a separate bundle, such as “esa.egscc.ri.automation.lang.atop” for the ATOP implementation. This architecture allows language-specific details to be encapsulated without impacting core system functionality. Most language implementations can leverage the existing core functionality and Automation Procedure Adaptation API with minimal additional modifications. It also facilitates future extensions through a well-defined plugin mechanism. While most language implementations remain self-contained within their respective bundles, the system allows for core functionality extensions when necessary, typically in scenarios requiring unique language features or specialized interactions with the Automation Procedure Adaptation API. By employing this modular and extensible design, KE-AUT provides a robust solution for executing procedures across various Space mission contexts.

4.2.2 Procedure Execution

Upon executing a procedure, the execution system sets up the necessary environment for running the procedure. This environment determines how the procedure interacts with the system, influencing aspects such as isolation, debugging etc.. A procedure can run within a shared Space or an isolated process, with each approach offering distinct advantages. Shared execution allows interaction with system components but may expose the system to failures. In contrast, isolated execution minimizes impact, making it useful for debugging or containing potential errors.

The procedure follows a structured execution sequence. It begins by receiving input values, such as arguments and contextual MCM objects. If a precondition method exists, it is invoked to determine whether execution should proceed. Steps are then executed sequentially, concluding with a confirmation method if defined. If any step, precondition, or confirmation method fails, the procedure terminates prematurely. Execution flow can be externally managed with commands like pause, resume, stop, and abort. These commands take effect at specific points, ensuring ongoing steps are completed before modifications occur. The abort command is an exception, immediately stopping execution, though its effect varies based on the environment. The execution mode can be specified at invocation, ensuring appropriate control, safety, and debugging capabilities as needed.

4.2.3 Debugging and Live Reloading

In order to ensure efficient troubleshooting, procedures can be debugged in a dedicated mode. This launches a process with an open debug port, allowing remote debuggers to connect. Additionally, KE-AUT supports exporting code for integration with external debugging tools, offering flexibility in debugging workflows.

KE-AUT also provides the capability of live reloading of procedures during M&C sessions. A listener continuously monitors procedure definition changes, triggering a recompilation and cache update upon modification. This ensures that subsequent executions use the latest version without requiring system restarts, improving responsiveness to evolving mission needs. The ability to reload procedures dynamically makes automation execution more adaptable to real-time operational demands.

5 End-to-End Workflows for Procedure Development

The ESA-Procedure Management Environment follows a structured lifecycle from initial authoring to operational deployment, optimizing interactions and streamlining data management through DevOps principles. This approach enables continuous integration, delivery, and deployment (CI/CD) to enhance efficiency and reduce manual workload.

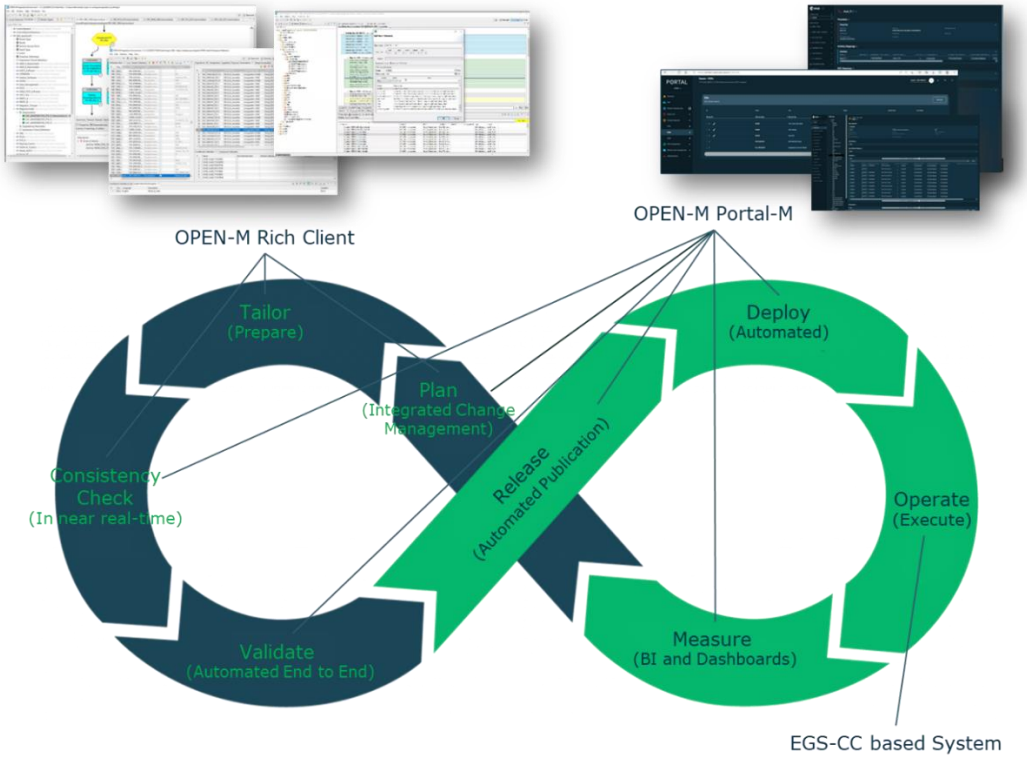

Figure 9: OPEN-M – Data Lifecycle

The process begins with the Flight Control Team (FCT) creating operational artefacts, such as Flight control procedures and spacecraft tailoring data, which are managed via a change management system. This system categorizes modifications (e.g., procedure updates, spacecraft data changes) and integrates them into a centralized mission repository. Verification and validation ensure artefacts meet operational standards before deployment, with proper versioning and integration into the operational environment for accurate execution.

5.1 End-to-End Workflow Integration within the OPEN-M ecosystem

Verification and validation are crucial for ensuring mission data compliance within the EGS-CC ecosystem. This process is designed for automation, aligning with DevOps principles. A continuous integration (CI) system verifies mission data throughout its lifecycle [2].

When a change request is initiated in Portal-M, a feature branch is created in the mission repository. Any modifications undergo an automated verification pipeline within the OPEN-M environment. Procedure Editors (presented in 4.1) in OPEN-M assist engineers in authoring operational procedures, automatically generating compliant operational data. Built-in syntax verification ensures consistency before changes are committed. Once submitted, the CI system triggers consistency checks to verify tailoring data against mission models, preventing inconsistencies early. Additional verification includes rule-based verification, cross-data dependencies, and simulated runtime testing to detect potential issues before deployment. Manual reviews by analysts and engineers complement automated checks, particularly for critical changes. Once approved, the feature branch is merged, generating versioned operational products that undergo final validation in the AIV environment before deployment. This structured, continuous delivery

approach ensures only verified and validated products enter the operational phase, minimizing risks and supporting reliable spacecraft operations.

6 Current and Future Technical Capabilities

The management of complex Space mission procedures requires advanced technological solutions capable of addressing the challenges inherent in modern Space operations. As described in previous sections, the OPEN-M/EGS-CC systems provides a structured approach to data and procedure lifecycle management, supporting the entire process from initial planning through final execution. By integrating preparation, verification, validation, execution, and monitoring, The OPEN-M system offers a comprehensive framework that enhances procedural management methodologies. The system's primary strength lies in its capacity to facilitate data integration across operational phases, utilizing a suite of tools that enable change management, real-time consistency checking, end-to-end validation, and automated deployment. This approach enhances operational efficiency while maintaining procedural integrity through coherence, traceability, and consistency between development and execution environments.

6.1 OPEN-MP Activity: Technological Advancements

The OPEN-M Procedure Management Extensions (OPEN-MP) activity has contributed to the development of ATOP's implementation, focusing on procedure preparation and execution. By addressing key challenges in procedure management, the activity introduced new configuration strategies within the OPEN-M framework.

One key advancement was the development of a transparent configuration management approach that enables the autonomous generation of spacecraft database items. This mechanism reduces manual intervention in procedure interpretation, minimizing human error and improving system reliability. The configuration management strategy incorporates runtime system integration, where procedural semantics are automatically translated into executable instructions. The development environment was further refined with intelligent assistance technologies, such as semantic highlighting and auto-completion, leveraging linguistic analysis to provide contextual guidance during procedure authoring. Additionally, integrated real-time consistency checkers and linters enhance verification by providing immediate feedback, ensuring procedural integrity throughout development.

6.2 Language Evolution and Future Perspectives

Upcoming activities will focus on expanding ATOP's linguistic and operational capabilities. Research objectives include enhancements to language specification, with emphasis on syntactic, grammatical, and semantic refinements. The extension of language capabilities to better support AIT and Software Validation Facility (SVF) use cases aims to broaden operational applicability.

A key challenge in language evolution is the development of a robust version management strategy. This involves mechanisms that support language evolution while maintaining backward compatibility – an essential requirement in domain-specific language design. The approach aims to ensure seamless procedure exchange across different language iterations, preserving historical procedural knowledge while enabling technological progress.

A Web-based preparation environment is also under development to improve procedural management interfaces. Designed with a lightweight, modular architecture, this environment functions as an independent procedure development tool while also allowing integration into runtime deployments. Its design prioritizes usability, providing authoring, verification, and management functionalities.

7 Outlook and Conclusion

7.1 Potential Research and Industry Adoption Pathways

Future developments of OPEN-M and ATOP will focus on technical enhancements and broader adoption within the industry. Research opportunities exist in procedural automation, with particular attention to intelligent technologies and collaborative methodologies.

7.1.1 Artificial Intelligence Integration

Integrating artificial intelligence into procedural management presents an area for further exploration. Machine learning techniques may contribute to procedure generation by analysing historical mission data, identifying execution patterns, and predicting potential contingencies. Natural language processing could support more intuitive procedure

authoring, enabling domain experts to generate procedures through linguistic interfaces that translate high-level intent into operational instructions. AI systems could also support predictive modelling, identifying spacecraft system behaviours and suggesting procedural modifications based on pattern recognition. This would shift procedure management from a reactive model to a more dynamic, adaptive framework that responds to evolving mission requirements.

7.1.2 Advances Verification and Validation

Automating the validation and verification pipeline will reduce human intervention throughout the procedure lifecycle, from initial development to operational deployment. A systematic validation approach would include automated syntax checks, semantic analysis, and cross-data consistency verification. The system could generate test environments, conduct rule-based validations, and verify dependencies to ensure that only validated procedures are deployed. Future developments may consider more advanced runtime verification mechanisms that operate across multiple abstraction levels. Beyond syntax checks, these systems could assess logical consistency and potential failure scenarios. Comprehensive simulation environments could enable exhaustive testing, providing digital twin models that replicate spacecraft system behaviours with high accuracy.

7.1.3 Industry Adoption

The adoption of ATOP within the Space industry requires a structured approach that considers the sector's conservative stance on new methodologies. Adoption efforts should focus on demonstrating clear benefits through comparative analyses and performance metrics. Developing structured migration strategies may help Space agencies and contractors transition to modernized procedural infrastructures with minimal risk.

7.2 Conclusion

The ATOP language and OPEN-M environment introduce a structured and goal-oriented approach to Space mission procedure management, addressing key challenges in automation. By enhancing clarity and precision through a dedicated DSL, ATOP reduces human interpretation errors while supporting automation. The OPEN-M environment further streamlines mission operations by providing an integrated framework for collaborative procedure development, verification, and deployment. Its flexible editing capabilities – encompassing textual, tabular, and graphical formats – cater to diverse user expertise and requirements. Additionally, a modular execution framework ensures interoperability across automation languages while maintaining alignment with the M&C model of EGS-CC.

Together, these advancements enhance the efficiency, reliability, and automation of mission operations. As Space missions grow in complexity, the methodologies introduced here may play an increasingly significant role. Future work will focus on refining these approaches and expanding their applicability within broader operational context of Space mission automation.

References

[1] F. Trifin, P. Choukroun, S. Gärtner, P. Hamacher, L. Schlag, “Joined ESA-DLR Procedure Management Environment” SpaceOps 2023, 17th International Conference on Space Operations, ID # 383 URL https://star.spaceops.org/2023/user_manudownload.php?doc=383__j8f403up.pdf

[2] Langs, A., Oertlin, J., Trifin, F.: “Applying continuous integration for operational products in the mission preparation environment”. In: Proceedings of the 17th International Conference on Space Operations (SpaceOps 2023), 5 (2023)

[3] B. Demeuse, S. Valera, PLUTO, a Procedure Language for Users in Test and Operations, 1998 DASIA Conference, 25/05/1998

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

[5] K. Davies, A.V. Payne, F. Croce, Automated ground control system operations for RADARSAT-2, 2004 SpaceOps Conference, 2004

[6] M.A. Seymour, The PLUTO operations procedure language and its use for RADARSAT-2 mission operations, 2004

[7] Walsh, A., et al. "*The European Ground Systems Common Core (EGS-CC) Initiative*", 2012 doi:10.2514/6.2012-1282732. URL https://arc.aiaa.org/doi/pdf/10.2514/6.2012-1282732

[8] European Space Agency. About OPEN. *OPEN Preparation Environments*. URL: https://open.space-codev.org/

[9] Beck, T., Schlag, L., and Hamacher, J. P., "*ProToS: Next Generation Procedure Tool Suite for Creation, Execution and Automation of Flight Control Procedures*" SpaceOps 2016 Conference, American Institute of Aeronautics and Astronautics. doi:10.2514/6.2016-2374 http://dx.doi.org/10.2514/6.2016-2374, URL https://arc.aiaa.org/doi/10.2514/6.2016-2374 .

[10] Trifin, F., Walsh, A., "Monitoring and Control Operations Preparation Framework for EGS-CC Based Environments", SpaceOPs 2018 Conference, American Institute of Aeronautics and Astronautics,. doi: 10.2514/6.2018-2711 URL https://arc.aiaa.org/doi/pdf/10.2514/6.2018-2711

[11] Trifin, F., Walsh, A., "OPEN: A community based preparation environment for EGS-CC based systems", Simulation and EGSE for Space Programmes, SESP 2019, URL https://atpi.eventsair.com/QuickEventWebsitePortal/sesp-2019/website/Agenda/AgendaItemDetail?id=33624499-2a59-471e-b755-b81e097a06f0

[12] Trifin, F., "The Next Generation Mission Operations Preparation Environment at ESOC", Simulation and EGSE for Space Programmes, SESP 2017, URL https://indico.esa.int/event/180/contributions/1369/attachments/1263/1488/1400_Trifin_-_Paper.pdf

[13] ECSS-E-ST-70-31C – Ground systems and operations – Monitoring and control data definition (31 July 2008) - https://ecss.nl/standard/ecss-e-st-70-31c-ground-systems-and-operations-monitoring-and-control-data-definition/

[14] ECSS-E-ST-70-32C – Test and operations procedure language (31 July 2008) - https://ecss.nl/standard/ecss-e-st-70-32c-test-and-operations-procedure-language/