Standard-Based Automation: Scalability, Flexibility and Exchange for Long Term Missions

Workshop on Simulation for European Space Programmes (SESP)
24-26 March 2015

ESA-ESTEC, Noordwijk, The Netherlands

Nieves Salor Moral[(1)], Dionisi Simone[(2)], Massimiliano Mazza [(1)]

[(1)] *Vitrociset Belgium*
Huygensstraat 34, Noordwijk, 2201DK, Netherlands
Email: n.salor_moral@vitrocisetbelgium.com, m.mazza@vitrocisetbelgium.com

[(2)] *Vitrociset Belgium in Germany*
Lise-Meitner-Strasse 10, 64293 Darmstadt (Germany)
Email: s.dionisi@vitrocisetbelgium.com

INTRODUCTION

Although each year several new missions begin, several more can be found in different stages of their life cycles. Nevertheless, all of them have to communicate with the ground stations in order to perform tasks, calibrate instruments, position themselves, etc. These processes are critical and a small failure may increase hugely the costs and throw away months of work. This complexity multiplies exponentially when several spacecrafts work together in a multi-domain environments, multi-satellites systems, etc.

The main part of the problem comes from the involvement of different tools for each mission and/or different parts of the complete lifecycle of spacecraft activities (e.g. creation, testing, operational, maintenance, storage).Therefore, the reusability and interoperability between systems/missions is somewhat a problem creating the need for new software to be added each time subsystems integration is required instead of scaling the software and unifying the needs and interfaces.

After interviewing several stakeholders and assessing the lessons-learned from previous projects, the main requirements of new and future automation space systems became clearer for being scalable and maintainable. Automation systems need to rely in the design of a formal component-based architecture which can be plugged and ported directly to other systems without risking inconsistencies or needing to develop extra components.

However, when trying to unify automation techniques with space standards, another issue appears. Nowadays, there are different systems that try to improve the MMI experience when creating and maintaining activities of type procedure. However, not only different versions of systems and procedures create problems of reusability and harmonization, but also the root understanding of what a procedure can do and cannot do, varies as the handling of the activities and commands.

Besides, most projects are born with heritages, both technological and operational/procedural. Project stakeholders or simply commercial partners want to reuse their know-how and background experience as-is (e.g. developed with non-homogeneous and very different tools), generating problems of interoperability between stakeholders. For example, the same activity may be defined twice for different missions, or the same functionality to be applied in different systems are incompatible, hereby losing time and resources and increasing the costs of new missions that could be avoided by standardizing the procedure preparation methodology.

In order to alleviate this bottleneck and based on our expertise in automation systems as a joint effort between the European Space Agency (ESA) and Vitrociset (VTCB), procedures will be set as the system automation core. The new system, named ASE5, tries to establish the foundations for future developments as a proven and validated system in the hopes of building expertise prior to the participation within the European Space collaboration known as European Ground-Segment Common-Core (EGS-CC) explained in [3].

In order to be considered for the EGS-CC, the new system had to follow all the ECSS Engineering standards, including [1] and [2]. The first one describes the procedure language to implement, while the second the data model known as Space System Model (SSM) which contains the spacecraft information to be stored. Furthermore and to increase the number of future users, the system should merge the best points of the most commonly used procedural environments (e.g. MOIS [4]) and those of the previous releases of ASE into the unified system that constitutes ASE5 to improve the user-friendliness and easiness of the procedure creation.

In summary, ASE5 provides a complete environment for the preparation, simulation, management and operation of activities (overall procedures compliant with the standard 70-32) around the influence of the populated SSM. Each of the functionalities to be implemented, based on the requirements validation, is completely decoupled and self-contained. This way, customers can adapt the required deployment to their specific needs being able to upgrade functionalities at any moment. In order to improve the interoperability of the procedures, the system follows a formalized approach based on the complete formalization of the requirements, the data and business models of the system and, in consequence, the standards of the SSM (70-31) and the procedural language (70-32).

SYSTEM CONCEPTS AND REQUIREMENTS

The information contained in the data model structure named Space System Model (SSM) represents the functional decomposition of the space system as a tree-like structure for elements of both the space and ground segment. The SSM contains data structures identified as objects. The recognized object types are System Elements (SE), Activities, Reporting Data (RD) and Events.
The SSM elements definitions are:

- System Elements (SE from now on) correspond to the elements resulting from the functional decomposition defined in ECSS-E-00 and ECSS-E-70.
- Reporting data (RD from now on) comprises parameters and compound parameters. It is associated to the SE.
- Activities are the monitoring and control functions. The term refers, in general, to procedures, telecommands and any function provided by the EMCS. An activity is associated to a SE.
- Events are associated with SE, RD and Activities and are occurrences of set of conditions that can arise.

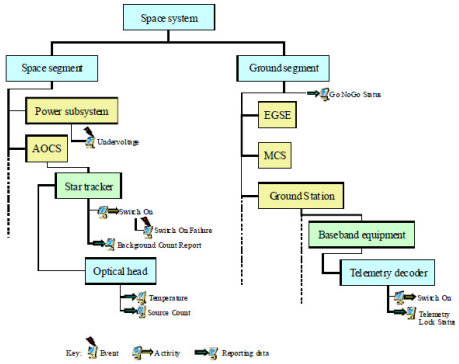

Figure 1.SSM Hierarchy Example

The relationships between the former data structures are based on containers, where a SE contains the activities which can be executed within; the information describing the SE is called RD and the events to be handled during an execution of an activity are also contained within the container SE. Any SE can comprise other SEs as sub-elements.
In the scope of the SSM, an activity is understood as a space system monitoring and control function implemented within the EGSE or any other mission control system. The possible implementations of activities are telecommands, procedures compliant with the European standard ECSS-E-70-32 and operating system commands.

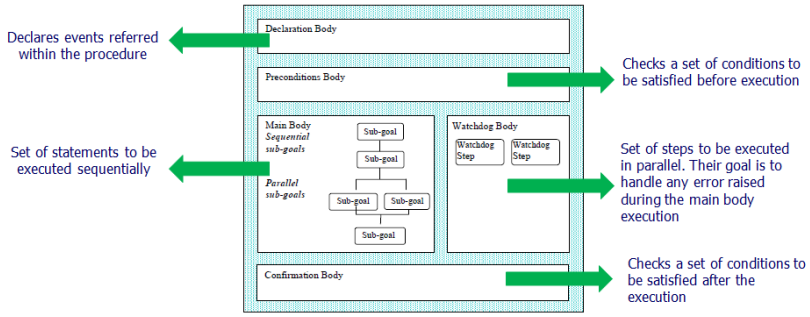

Figure 2 Procedure Structure

Procedures, as the core of the system, will need to be compliant to the semantics of the standard. They have a well-known structure (see Fig.2) to ensure the activity goal is satisfied and a control management can be performed to define the process flow. In the system, two different languages are accepted for defining procedures. As such, several other input methods can be allowed to improve the user experience by accepting other procedure formats while the respective mapper between the input format and the 70-32 is provided.

SYSTEM ARCHITECTURE

ASE-5 is planned to be the baseline product of VTCB and to give support to different external applications. In order to achieve these high-expectations, the system is scalable, portable and flexible while secure, safe and robust in its design.

Although in principle these properties seem to oppose each other, new technologies allow them to merge and take their advantages for a common-goal.
In order to be scalable and flexible, the design has made use of a tailoring of the Service Oriented Architecture by using some principles of the SCA[5]. The main idea of these architecture patterns is to allow new applications/services to be added without any need to modify already existing ones. Also in case one already existing component changes, it will not impact the rest of the system due to the encapsulation of its logic.
Unlike SCA, the portability of the system is achieved via three manners:

- The use of the Java programming language in the implementation, hence providing OS portability.
- The use of a light-weight ESB[6] as linking and central point of the architecture. It will allow for several types of deployments depending on the client needs (e.g. from a local deployment in one machine to a completely distributed and remote one).
- The use of open-source portable tools and libraries, when required but based overall in the de-facto standards EMF and XML for RCP applications, so data models and architecture wise

The architecture design based on standard architecture patterns and the use of already tested and approved libraries and tools ensure the system safety and robustness. However, besides those measures, the system will provide a component for the secure management of the information and its safety which can be updated with new security mechanism and/or rules depending on the platform/devices it is intended to be used.
Regarding the dependencies with the data sources of the system, the architecture has been designed to access the data sources independently for the services calling the request. This decoupling design decision has been taken due to the highly possible need to change or modify the data model or the database engine in the future. In this way, services do not need to know which database engine is being used or how the model is organized; only the interface of the data source and the route to be followed will be configured, based on the deployment requirement and secured by encryption.

System Modules

The diagram below shows a logic representation of the system architecture. In this architecture the system core is composed by the SSM Product and the Activity Execution Compiler and Engine. The management of the space system model data and its consistency is the responsibility of the first one, while the automatic activity execution and procedure compilation is managed by the second one. It is undeniable there is a dependency between them since procedures are part of the SSM and internal references within a procedure to other SSM elements exist.

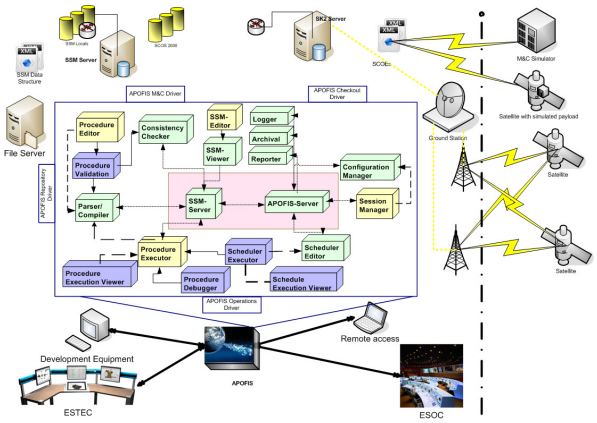

Figure 3 ASE-5 System Modules

In order to create light applications which later can be ported to the Web and other devices, most of the heavy tasks are performed in the server machine(s) where the databases, data model engines and the activity executor engine are located. This decoupling between client and server requires constant communication between them to pass data. This information stream will also have to be shared between all clients which are connected in real time. The sent/received information not only has to be managed in a secure way but has also to be guaranteed to the users to not lose any data packet. The architecture allows this communication by the implementation of web-services.

With maintainability, consistency and improvement capabilities in mind, all views have been created following the RCP mechanism through the eclipse SWT library. As the system has to answer to the preparation and operations of automatic activities, the views have to be user friendly, oriented to the goals to achieve (i.e. SSM browser, procedure edition and activity execution viewer) and to not distract the attention of the users while improving their performance.
Therefore, the main client components offered by the systems are:

- SSM Viewer/Editor displays the SSM data model in a tree-like structure while allowing partial views for deeper details (e.g. properties view per elements). This view will communicate in both synchronous and asynchronous way with the SSM Server. Synchronously whenever there is a specific request for information (e.g. in a refresh call or when asking for further details) and asynchronously for the receiving the change notifications from other users.

- Procedure Editor allows the edition and display of the procedure scripts in different formats and grammars. The editor provides content-assist, syntax highlight, semantic validation, folding and collapsing, formatting, etc. Its use has to be straight-forward with a small learning curve so it does not require a deep knowledge of the procedure language. Therefore the editor allows textual and graphical definitions sharing the same procedure data model which will be escalated in the future with more formats (e.g. tabs or tables) and also adding different languages (e.g. JAVA, C or ADA).

- Activity Execution Viewer displays the execution trace of an activity so an operator can not only verify the result of the execution but also modify the flow sending signals to the execution (i.e. abort, pause, resume) and browse through the execution in detail and check the timings, execution and confirmation status of each task executed. For an activity of type procedure, the view contains the full decomposition of a procedure in its smaller tasks, so the operator can see the specific process flow.

- Scheduler Editor/Viewer allows operators to create and display activity schedules. This display communicates with the schedule executor and the activity executor for retrieving the execution data and to prepare an execution plan.

The previous views are fed by the data provided from the functionalities implemented in the following modules in the server:

- SSM Product implements the Space System Model and the management mechanism of the data contained in the different source formats.

- APOFIS Product is responsible for preparing, compiling and executing activities contained in the SSM product. It also allows scheduling already prepared activities. The APOFIS component can be further decomposed into the following components:

    - Activity Editor/Executor is responsible for preparing and automatically executing simple and complex activities.

    - Schedule Editor/Executor offers the possibility to create and execute activity schedules

    - Parser / Compiler is responsible of parsing activities of type procedure, notifying all lexical, syntactic and semantic errors, and in case there are no errors, generating the activity executable code. As well as translating from one input procedure language into another.

- Reporter manages the creation, edition and display of reports based on the gathered information of the system.

- Logger provides the functionality of logging relevant information about the system working process and user interactions during the system execution time.

- Consistency Check Product performs data consistency check. To that end, it interacts with others components such as the SSM, in order to assess the existence and the reachability of individual space system objects and to check the consistency of the element referred in activities of type procedure. The consistency check is accomplished according to the checked object: i.e. a database, an SSM or an activity of type procedure. The database consistency check is performed verifying that the values in the table enforce one or more constraints, whereas, the SSM consistency check is carried out verifying that all the referred information exists in the data model and the execution contexts are correct. The activity of type procedure is consistency checked against only the SSM since the SSM contains all required data definition needed by the execution environment.

- Session Manager Product provides means for creating, selecting or archiving an operational configuration per session.

- Configuration Management Product provides a flexible and intuitive entry point to define the desired configurations relevant to the operations to be run.
- Authentication & Authorization product manages the user roles by which the system determines the level of system access in accordance with user privileges.

Procedure Harmonization Management

In the European Space Agency, the automation is performed through procedures and although there is not a unique format for the procedures used, the official standard for ground segment procedures is the 70-32. Thus, this system automation is performed through the execution of procedures compliant this standard.
Based on Vitrociset expertise in the automation field and the opinions of some final users, one of the major issues with the implementation and use of this standard is the complexity and understanding of its grammar. Before implementing the procedure editor and the execution engine, its whole behavior had to be clarified and had to match the one the final users are expecting. As such a preparation process (i.e. [7]) was performed by creating formal data and process models of the procedure standard. After receiving verification approval from ESA, a procedure meta- model was generated (see snapshot in the image below).

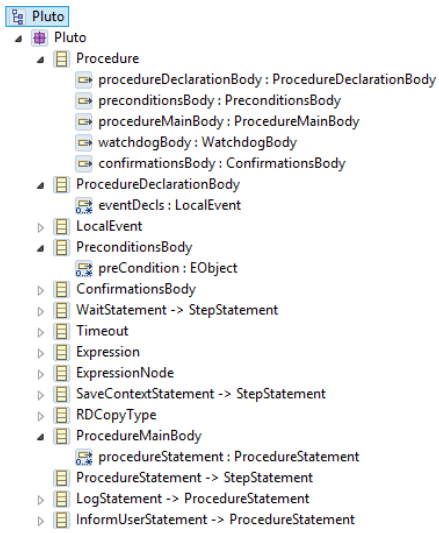

Figure 4 Procedure Meta-Model

This meta-model has been implemented with the EMF technology to create java data beans These data beans are used during execution to perform the approved process model through the object oriented approach instead of working directly with database objects. The population of the beans is performed through the allowed editors.
By default, the system comes with two editors integrated. These two editors are implemented through the latest technology named Xtext. This technology offers the possibility to automatically generate an out-of the box editor by defining just the grammar. As it is also based in the EMF technology, the grammar easily instantiates the existing procedure meta-model by specifying one to one the mapping among the beans fields and the procedure input. Besides, cross-references to the SSM Model are also part of the grammar to invoke other activities or to get properties from reporting data, system elements or events.
For each grammar, the system has to manually implement only the specific validation requirements, formatting rules, outline views, user templates and information scoping. The rest is either shared between grammars or automatically generated (e.g. label and content providers, lexical and syntax checks or consistency checking).

The grammars integrated are the meta-grammar, which is based on an XML format for improving its interoperability and exchange of procedures, and the PLUTO one, which is the example and most commonly known for the standard uses and is based on natural language.
The user can define the procedures in any of the two grammars as they are fully compliant with the meta-model. This approach allows the automatic translation from one editor to the other and the synchronization of the information among them so the user can edit in the preferred editor although originally has been defined with the other. This feature permits to reuse all procedures already created for existing missions (like VEGA) without any further effort.
In the future, new grammars can be plugged-in the system if they are compliant with the standard, thus with the created meta-model. The translation between grammars is automatic since the instance of the beans only need to use the content-providers automatically generated when defining the grammars.
However, users may want to also reuse those procedures created in no compliant languages (e.g. TOPE/MOIS, C, Java or JavaScript). For this use case, the system will need to create only external drivers to be plugged from those languages to the compliant ones. This effort is currently being tested with the assessment of including OTX automotive standard to the space sector.
Once the beans are populated, the input format of the procedure is no longer a decisive factor. Thus, only one execution engine has been implemented instead of several ones. This approach has allowed the system to decouple completely the view from the logic, thus reducing the impact if the grammar is modified according to user needs and properly test both parts.

SYSTEM USER CASE

Having a complete, unique system providing every feature to every space user (i.e. operator, architect, scientific or tester) is the goal ASE-5 tries to achieve. Nevertheless it has to be usable, efficient and reliable. A typical use case where the system may be applied is the connection of a user for preparing one or several procedures for a test campaign and their simulations to assure the results are the expected ones prior to upgrade them into operational status. The system allows creating/editing procedures in the language of the meta-model or in the PLUTO language defined in the standard.

System Access Modes

ASE-5 is an environment for procedure/activities preparation and execution with two main operational modes (preparation and execution which can be in:

- Stand-alone mode (without connecting the application to the target SSM Database Implementation, e.g. the ESTEC Database reference facility) which will behave as simulation.
- Integrated mode (connected to the target SSM Database Implementation).

When a user connects to the procedure manager, his role determines the operational mode. While in preparation mode, activities and schedules can be edited and the user can run procedures for testing. However, commands cannot be sent to the operational system. Some Dry Run Servers simulate the behavior of the device where the procedures run in case the user wants to test the procedures in stand-alone mode. In both modes, the user will have the same experience since there is no discontinuity of the user interface.

SSM Perspective

The first view of the system after connection is the Space System Model viewer. This is the single view where the user can access all the space information and can verify its consistency.
The central view shows the structure of the current Space System. A tree navigation system (i.e. a tree that allows the user to expand / collapse nodes) is offered to explore the Space System Model.
Moreover the system offers the user the possibility to have a direct way to assess the Space System Model while browsing it, selecting the elements on the tree and, according with their type, executing some functionality (i.e. editing, importing, exporting, validating).

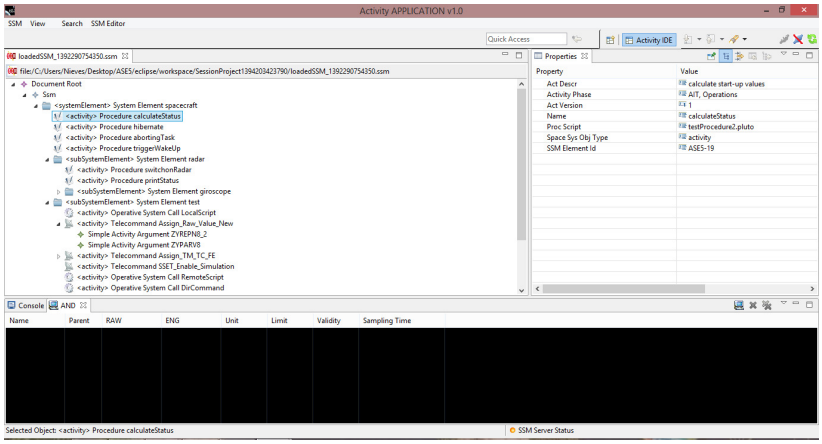

Figure 5 SSM Perspective HMI

At the bottom of the application view the user has the status bar with the status of the SSM connection with the database or with information about the element selected on the central view

Procedure Edition/Creation

After the user has connected to a Space System Model, the user can select the activity of type procedure to be defined and the procedure editor will be opened. Depending on the extension of the script the system will load the specific editor and the content will be automatically loaded.
Synchronously with the edition the system loads the semantic structure of the procedure also known as AST in the Outline view. As soon as the user modifies/creates an allowed and correct structure, it will appear in the AST.

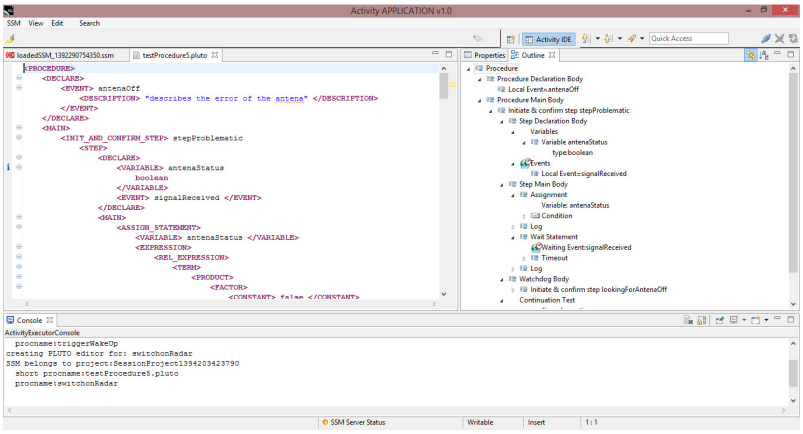

Figure 6 Procedure Edition Perspective

To improve the user experience during the edition, the system highlights the syntax that belongs to the selected language (PLUTO or metadata) and provides different functionalities like auto-completion and content-assist or auto-formatting of the text.
The user can decide to compile the procedure at any moment and when there are compilation errors the system highlights them on the editor with a tooltip describing the error found. The system also executes automatically a consistency check controlling if the SSM elements contained into the procedure have existing and accessible references to the working SSM.

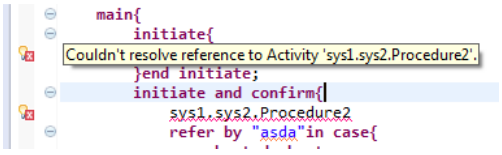

Figure 7 Editor capabilities

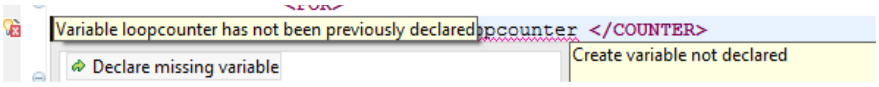

Figure 8 Quick fixes provided

For some kind of errors (e.g. missing declaring variables and events), together with the error symbol, there will be a proposed quick fix icon (i.e. a bulb) which will automate the solution.

Activity Execution/Simulation

When an activity has been prepared and it is considered valid (in case of a procedure, this means compiled without errors); the user can decide to execute an activity on the SSM. In that case he/she should right click on the activity and select the menu option "Execute ….." (The name of the option changes accordingly with the activity type).
If the activity is a procedure, the system will compile it automatically and it will return a message to the user in case of compilation errors. Otherwise the system will check if any argument is needed for the right execution of the activity and if so, the system will request the user to provide the

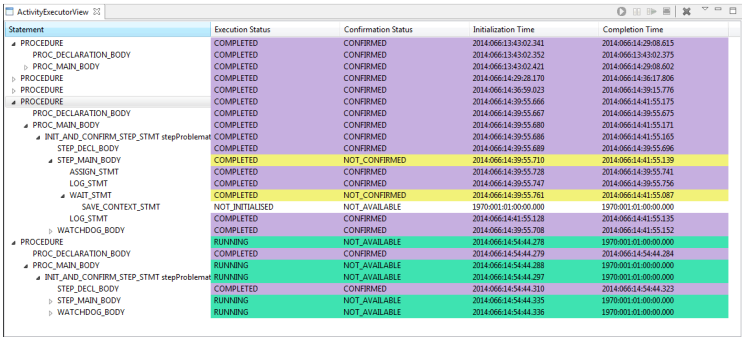

Figure 9 Activity Execution View

information prior to execution and validate it depending on the argument type.
Once, the verification of the activity is performed, the system will load the activity in the Activity Executor View and it will update its execution and confirmation status according with the status received from the other application (i.e. SCOS 2000 in case of TC, scripts or command execution result in case of OSC or the procedure engine otherwise.).
During the running of the activity the user can decide to abort / pause and consequently resume the execution of the activity (if the external application allows this kind of operation).The system will highlight the different statuses of execution. After the execution of an activity is completed the user can remove the execution instance from the view.

Alphanumeric Display

Another aspect of the activity execution is the monitoring of the received telemetry (or reporting data as it is understood in the system). This information can be used to alter the action flow or perform some tasks.

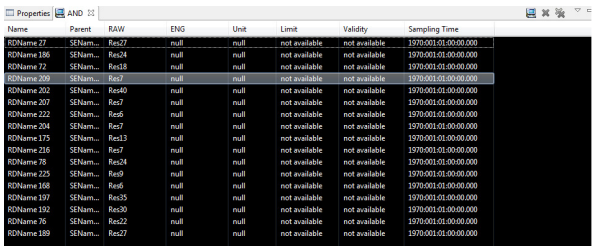

Figure 10 Alphanumeric Displays

The AND display shows in real time the engineering value of the reporting data present in the display. To add a reporting data to be monitored, the user has to select it in the Space System Model perspective and drag and drop the element onto the AND view. As soon as a change in its value is received in the client, it will be modified in the view. For improving the monitoring purposes, the view will highlight with colors the status of the reporting data according to its limit and validity checks.
The view also gives to the user the possibility of removing a single reporting data, remove all if the monitoring is no longer important and to save or load a predefined list of reporting data for a specific test/operation campaign.

CONCLUSIONS AND FUTURE WORK

In this paper the most important features of the newly developed automation system named ASE-5 has been explained. In its design and development an important effort of working with space standards have been applied. To increase its validity and long-term durability, the architecture developed has been implemented with the latest technologies and based on model instead of long-existing ones and hard-coded features.
Although Vitrociset has gambled that the automation in space is going to be performed through procedure and formal models, the system wants to also be used by current users and missions. This is achieved by allowing instead of a single/future format we want to achieve a non-closed set of formats including programming languages, non-European formats, etc. with little manual effort without having to modify at all the execution engine.
Although this release still needs further work and improvements, the base of the functionality is already set and deeply tested.

REFERENCES

[1]ECSS-E-70-32C - Ground systems and operations-procedure definition language; issue 2.0,July 2008
[2]ECSS-E-70-31C - Ground systems and operations-Monitoring and control data Definition; issue 3.0. July 2008
[3]Walsh,A; Pecchioli, M; Charmeau, MC; Geyer,M.; Parmentier, P; Rueting, J; Carranza, JM; Bosch, R; Bothmer, W.; Schmerber,PY.;Chiroli,P. “The European Ground Systems Common Core (EGS-CC) Initiative”; 11-15 June 2012. SpaceOps2012.
[4]Heinen,W.,Reid.S,,Varadarajulu.S;“Automation through On-Board Control Procedures: Operational Concepts and Tools” 25-30 April SpaceOps2010.
[5]Laws,S., Combellack,M., Feng,R., Mahbod,H., Nash,S;”Tuscany SCA in Action”; February 2011; ISBN: 9781933988894
[6]Rademakers,T.; Dirksen,J.; “Open Source ESBs in Action”; Manning Publications Company, 2008
[7]Salor Moral, N; Dionisi,S.; Mazza, M.; ”Model Conceptualization of a Procedural Standard for Improving Interoperability”; SpaceOps2014 “in press”.