Model Based Operations Validation System

C. Laroque[1], D. Lucia[2] and M. Götzelmann[3]
VEGA Space, Darmstadt, Germany

S. Haag[4]
European Space Agency, European Space Operations Centre, Darmstadt, Germany

Operational procedures are typically initially designed at the spacecraft manufacture site by the system/AIV engineers, tested with the spacecraft database, and then published in the flight operation manual. The Flight Control Team (FCT) takes the delivered procedures to build the LEOP, commissioning and routine operation procedures. These procedures however are not always sufficiently validated, due to lack of time. Testing and validation of procedures is often left as a low priority task performed late in the development process.

Once developed, there is the need to allow easy (re)validation of the procedure in an automated or semi-automated manner. This validation is not only necessary for checking the consistency of the procedures during their development, but also to verify that the procedures achieve their objective, are safe, and do not endanger the integrity of the spacecraft. Moreover, automatic regression testing needs to be carried-out whenever the S/C database is updated, Mission Control System (MCS) or operational simulator versions are updated, or when Mission Planning System (MPS) outputs need validation.

In order to improve the level of testing and reduce associated effort and costs, ESA has conducted a study the prime objective of which was to develop innovative concepts and tools to automate the validation of operations procedures for current and future missions.

The developed concepts allows automatic validation of procedures at all stages of their development: initial development, upgrade to accommodate a new database or On-Board Software (OBSW) version, modification resulting from test campaign activities, and regression testing. Tools using validation models have been designed, prototyped, and evaluated with extensive involvement of perspective users. These tools can be shared between industry, spacecraft manufactures, the operations teams and any party for the validation of procedures such that the effort for procedure development and maintenance is reduced, and that procedures developed by one party can be provided and re-used by the other parties with minimum effort.

I. Current Practices

OPERATION procedures are typically designed at the spacecraft manufacture site by the system engineers. The system engineers work in coordination with the Assembly, Integration and Verification (AIV) engineers who encode the procedures in a specific Operation Language (OL) that can be executed. The procedures are tested with the spacecraft database. Once successfully tested, the procedures are then published in the flight operation manual.

The Flight Control Team (FCT) takes the procedures delivered with the spacecraft flight operation manual (spacecraft handbook) as input to establish the Flight Operations Plan (FOP). These procedures however are not always sufficiently validated due to lack of time, the testing and validation of the procedures is often left as a low

[1] Delivery Manager, Ground Segment Systems, Technology Division, VEGA Space GmbH, Europaplatz 5, D-64293 Darmstadt, Germany.
[2] Software Engineer, Ground Segment Systems, Technology Division, VEGA Space GmbH, Europaplatz 5, D-64293 Darmstadt, Germany.
[3] Principal Consultant, Technology Division, Europaplatz 5, VEGA Space GmbH, D-64293 Darmstadt, Germany.
[4] Software Engineer, Ground Systems Engineering, European Space Operations Centre, Robert-Bosch-Str. 5, 64293 Darmstadt, Germany.

priority task performed late in the development process. The procedures are not easily machine readable, they may be provided on paper, or in an electronic format that does not allow easy exchange. Finally, procedures are often based on procedures developed for AIT activities, and are therefore not easily reusable as they were developed for different purposes.

Based on this input, the FCT develops procedures for the commissioning and routine operations, and for LEOP. Two types of procedures are prepared: Flight Control Procedures (FCP) describing the nominal spacecraft operations, and Contingency Recovery Procedures (CRP) describing the operations in order to recover from errors occurring at spacecraft or ground level. Once developed, the procedures are tested at two levels to ensure that

1) *What is described in the procedure correctly executes*: the commands described in the procedure are properly uplinked and executed, the checks implemented in the procedure pass. It is not checked that the commands or the checks defined in the procedure are the proper/correct ones.
2) *The procedure does what it should do*. For instance, if a procedure is designed to switch-on equipment on the spacecraft, it is checked that the equipment is switch-on and remains on, that it consumes power, that it influences the thermal characteristics. If the procedure concerns a spacecraft manoeuvre, it is checked that the amount of consumed fuel on specific tank is the expected one, and that the spacecraft behaves as expected (check spacecraft dynamics). Typically, these checks need to be performed over a certain period of time. These tests also ensure that there is no degradation on other spacecraft units, and no reconfiguration.

The testing of procedures at both levels currently is a manual activity which is very time consuming. It requires a large amount of manual checking on the involved systems to verify correct execution of the procedure. Investigation of potential problems can also be difficult and time consuming.

In the first place the simulator is used for the test because the spacecraft is usually not available and untested critical procedures cannot be tested directly on the spacecraft. Moreover, it is not possible to test all procedures with a spacecraft on ground. In order to increase the fidelity of the testing, the procedures are in some case re-tested using the engineering model and/or the spacecraft when time allows.

The current practices show the following problems and area for improvements for procedures development and testing:

1) Development and validation of operation procedures currently requires a major investment of effort by the spacecraft Assembly, Integration and Test (AIT) and Flight Control Teams.
2) Late testing.
3) Effort spent on spacecraft manufacturer site to perform initial testing does not benefit to the FOP testing, since different tools, systems, and testing approaches are used.
4) Regression testing often not performed due to lack of time and budget.
5) Manual testing still predominant. The manual activities, although well understood, still require significant human effort by skilled engineers; they lack repeatability and result in duplication of work.

II. Objectives

IN order to improve the level of testing and reduce associated effort and costs, VEGA Space has conducted a study for ESA the prime objective of which being to study the concepts enabling automatic validation of operations, and to develop a Model Based Operation Validation system (**OPSVAL**).

The new concept enables automatic validation of operations without or with minimum human involvement in an automatic manner at all stages of their development: initial development, upgrade to accommodate a new database or On-Board Software (OBSW) version, modification resulting from test campaign activities, and regression testing. The concept maximizes the reuse of procedures throughout all development operation phases in order to avoid duplication work and re-testing of the same or nearly identical procedures by different parties. Industry, spacecraft manufactures, the operations teams and any

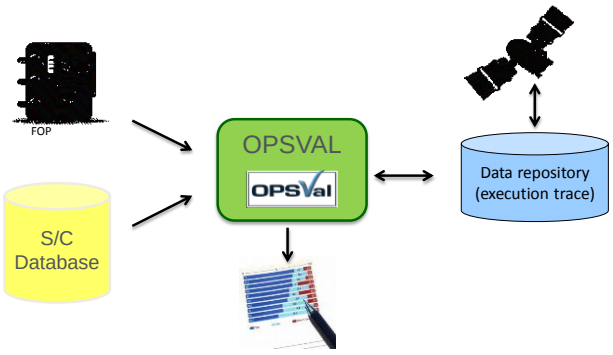

Figure 1. OPSVAL Objective - *provide the FOP, S/C database, and data repository containing the TM/TC data, OPSVAL will validate and provide test report*

involved sub-contractor can share a new common tool for the validation of procedures such that the effort for procedure development is reduced, and that procedures developed by one party can be provided and re-used by the other parties with minimum effort.

At the time the concept was established, an important part of the activity was to check and exploit the potential synergies with the other systems involved in operations. The automatic validation of procedures is not only interesting for the procedure developers. It can also help identifying space & ground segment malfunctions, increase the level of fidelity for the testing of operations, Operations Control Centre (OCC) systems, and simulators.

Once the concept was established and the synergies defined, investigations have been performed to check how the concept could be applied using the current infrastructures at the European Space Operations Centre (ESOC) and also on industry side. In particular, it was checked which existing system could be upgraded to fulfill the concept, what missing function should be added, and what missing element should be specified and developed from scratch.

The following sections describe into more details the detailed objectives of OPSVAL.

A. Scope of Procedures to be Validated

The prime objective of OPSVAL is to develop concepts and tools to automate validation of operations procedures for current and future missions. It cannot be expected that existing missions will rework all procedures to allow execution by an automation system and therefore the OPSVAL concept must aim at automation of validation without requiring that operational execution of procedures is automated as well.

OPSVAL shall not be constrained to validation of procedures designed for "manual" operation and shall ultimately provide means for validation of automated procedures as well. This would allow for future mission to ease repetitive testing and gain full benefit of the concept.

B. Validation Scope

OPSVAL should not only verify that a procedure achieves its objectives, but also that it is safe and does not endanger the integrity of the spacecraft. For this purpose OPSVAL shall monitor invariants and shall check conditions that shall not be violated under all circumstances. In particular OPSVAL shall check that all telecommands used in the executed validated procedures are actually verified by the control system or EGSE system and shall produce warnings for any out of limit conditions unless these explicitly specified as expected during execution of a specific procedure.

C. Execution and Validation Environment

OPSVAL should be useful for validation of procedures in various contexts, including spacecraft operations and AIV. This implies that the actual validation function must be as independent of the procedure execution environment as possible and only depends on data that have been collected during execution. While OPSVAL will have to rely on data generated by a ground control system, its dependency on a ESA MCS mission control system should be minimized. Ideally it should be possible to collect data used by OPSVAL for validation also on an EGSE system.

Independence of the OPSVAL validation function from any specific execution method or execution environment also implies that the validation must be an offline process, using data collected during execution of the procedures and stored in a database.

D. Use Cases

Setting up of OPSVAL for automated validation of procedures will not be without cost but it is expected that the biggest benefit of automated procedure validation will be gained for (non-) regression testing whenever updates are made to the spacecraft database, the simulator, the mission control system, or the procedures themselves. Therefore the first priority use case for OPSVAL is automated regression testing. In this use case, OPSVAL is actually not only useful for validation of the flight control procedures, but for the complete assembly of MCS, Simulator, Spacecraft Database, and procedures.

Availability of automated procedure execution, collection of validation data, and offline validation is expected to be of benefit also during validation with data collected during SVTs in order to make best use of the limited time available to test procedures with the real spacecraft. SVT operations are therefore a second use case to target.

OPSVAL should also provide support for procedure testing during procedure development. In contrast to the previous use cases, testing during development will typically deal with individual procedures, require a minimum of overhead for test configuration, and require minimizing turnaround times by validating results very quickly after procedure execution.

A final use case identified for the OPSVAL is the validation of MPS output.

E. Ease of Use

The prime objective of the OPSVAL initiative is to reduce the workload on Flight Control Teams and therefore OPSVAL must not require more work for configuration, set-up, and operation than absolutely necessary. This not only requires a user friendly man machine interface but also implies that the FCT must not be required to re-enter specifications that are already in the procedure specifications. As far as possible, test and validation instructions shall be extracted from the procedure specifications.

F. Configuration Control

Validation results are meaningful only for a well-defined configuration of the space element (Space craft or simulator), ground control system, spacecraft database, and procedure definitions. Strict configuration control of all test specifications and test results identifying also the configuration of the execution and validation environment, is therefore a core requirement for OPSVAL.

III. OPSVAL Concept

BASED on the objectives defined above, the OPSVAL concept has been defined in close collaboration between VEGA Space, ESA operations and flight control teams, and industrial partners.

The flight control procedure validation concept is based on checking of data collected during procedure execution according to a set of specified rules executed by a rule based data processor.

The overall validation process can be broken down into the set of sub-processes shown in "Fig.2". Each sub-process except the last generates data sets that one or more other processes use as input. Most of the sub-processes also need to use the spacecraft database which is not specifically shown.

Most of the sub-processes can be implemented in different ways with different systems and varying degree of automation. Defining the sub-processes as separate isolated activities that are only coupled by the output/input data sets allows combing different implementations according to the available environment and the preferences of the user community.

The following sections describe each of these sub-processes to more detail. They outline the general concept of the sub-process in a manner that is as independent of any implementation as possible and do not intend to describe implementation of an OPSVAL system but rather identify options for implementation of the concepts through cooperation of existing systems and new OPSVAL components, without attempting to precisely specify the borderline of OPSVAL.

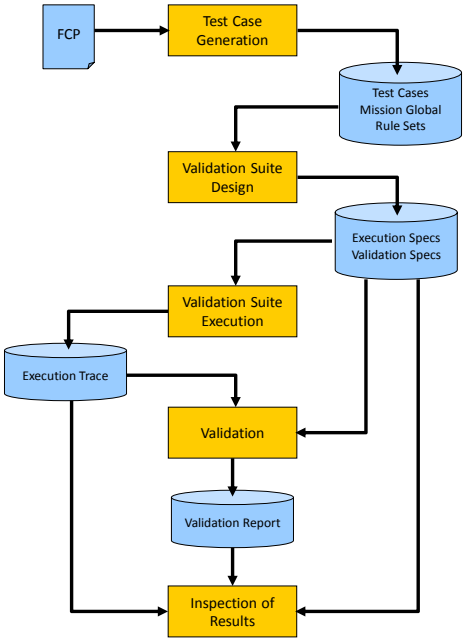

Figure 2. OPSVAL Validation Process

A. Test Case Generation

The task of the process Test Case Generation is to prepare a specification of how an individual procedure can be validated. A test case identifies the rules that shall be applied for checking of the execution trace of the procedure and correlates them with the command sequences and individual commands that are executed as part of the procedure. This concept is illustrated by "Fig.3".

One can distinguish implicit rules that can be automatically applied by OPSVAL and explicit rules that must be specified as part of the Test Case Generation process. An example for an implicit rule is that every telecommand must be completely verified according to the verification specification in the database. An example for an explicit rule is the verification that one or more specified telemetry parameter values conform to a specified condition.

In cases, where a command sequence "invokes" a command sequence defined in another procedure, the validation rules must be extracted from the defining procedure and included into the test case specification. When a complete procedure is called up, a reference to the corresponding test case specification must be included which can then be expanded when creating a validation suite.

In the execution trace a command sequence can be identified by its first provided the command sequence in which the command was executed is identified in the command history. When the same command sequence is executed more than one the time order of the invocations will still allow uniquely identifying the invocation.

Beside the procedure specific test cases one or more Mission Global Rule Sets need to be defined which define 'constraints' or 'invariants' that should be always respected unless explicitly disabled in the course of a procedure. Also for the mission global rules one can distinguish explicit and implicit variants. Examples for explicit rules are 'both transmitters cannot be ON at the same time" and "batteries shall never discharge", of course expressed by appropriate telemetry checks that the rule processor can understand. An example of an implicit rule is that no alarm (out of limit condition) should occur during execution of a procedure unless expected out of limit conditions have been explicitly specified.

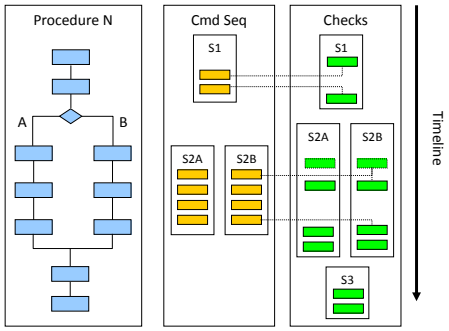

Figure 3. Test Case Specification Concept

Mission global rules may be violated by specific procedures and therefore it must be possible to disable specific global rules for the duration of a procedure or command sequence execution within a procedure specific test case specification. As an example, a procedure test case specification might need to allow one or more telemetry parameters to go out of limits because of the operations invoked by the procedure.

All test case specifications as well as the mission global rule sets must be stored to a special repository and kept under configuration control.

B. Validation suite Design

A Validation Suite is a set of procedures which are executed for the purpose of validation combined with instructions and rules for their execution and validation. As procedures depend on certain preconditions that may be influenced by other procedures the chronology of procedures within a validation suite is important. Executing all procedures sequentially will simplify the analysis of validation results but concurrent execution of procedures performing activities on unrelated subsystems should be possible to support the validation of mission timelines generated by mission planning systems.

Where procedures include branches, the branches to be executed must be selected. The same procedure can be included into a validation suite several times with different branches selected for each invocation of the procedure. When a command sequence defines formal parameters and the values of these parameters are not specified by a "calling" procedure, the values to be used for the validation suite must be specified by the validation suite designer.

What are actually executed are the command sequences that are extracted from a procedure. Parallel branches of a procedure are typically implemented by independent command sequences and therefore the output of the process is a "sequence of command sequences" with associated timing that can be executed within a suitable execution environment. We refer to this output as the Execution Specification for the validation suite.

The second output of the process is the Validation Specification for the validation suite. In essence, the validation specification consists of a set of rules that specify how the command sequences are identified in the execution trace and what checks shall be executed for the command sequences. These rules can be extracted from the test case specifications for the procedures included in the validation suite. In addition the validation specification will have to identify the mission global rule sets to be applied for the validation suite.

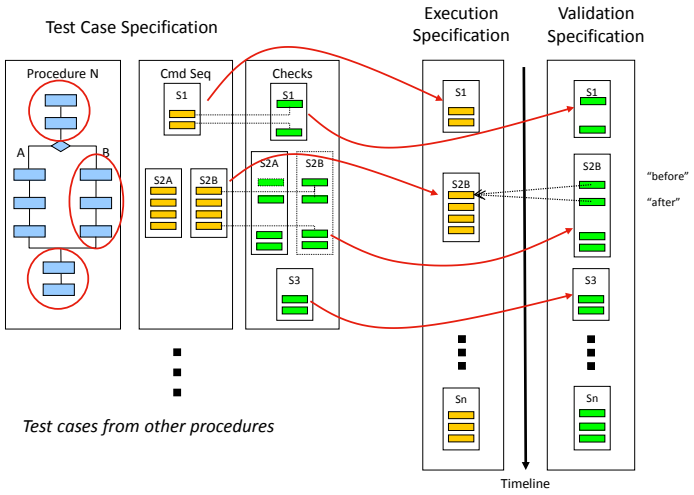

Figure 4. Validation Suite Concept

The overall process of designing a validation suite is illustrated in "Fig.4".

Both the execution timing for command sequences and the checks to be performed can be extracted from the test case specifications but may need tuning for the specific validation suite. For the relative timing of command sequences defaults may be applied but there should be means to fine tune the timing based on these defaults.
The execution specification and the validation specification must be stored to a special repository and kept under configuration control.

C. Execution Environment

The anticipated execution environment includes a ground control system, the space element, and a validation data collection and storage facility. The ground control system can be an MCS or an EGSE. The MCS may be connected to a spacecraft simulator or the real spacecraft in the context of a System Validation Test (SVT).

In general the command sequences specified could be executed "manually" by loading them on the stack and up-linking the commands to the space element for execution. Especially for regression testing purpose automation of the execution is an essential requirement and therefore only implementation options including a minimum degree of automation are considered. These implementation options include:

1) Generation of an automatic command stack (auto-stack) from which commands are released by a MCS according to the timeline defined by the execution specification;
2) Generation and up-link of a mission timeline that is executed onboard according to the timeline defined by the execution specification;
3) Generation of suitable PLUTO procedures and possibly schedules for execution by an automation system that load and uplink the command sequences according to the timeline defined by the execution specification.

Use of these execution options is illustrated in "Fig.5".

When using an automation system, several options are available for execution:

1) The system could just be used to configure the control system and the simulator and to load and start a command stack that contains the actual "validation timeline";
2) The system could in addition be used to uplink individual command sequences as specified in the execution specification;

In principle the automation system could also uplink individual commands and check conditions in the telemetry, but that may have the effect that the command sequence is not identified in the command history and the proposed validation approach relies on that information.

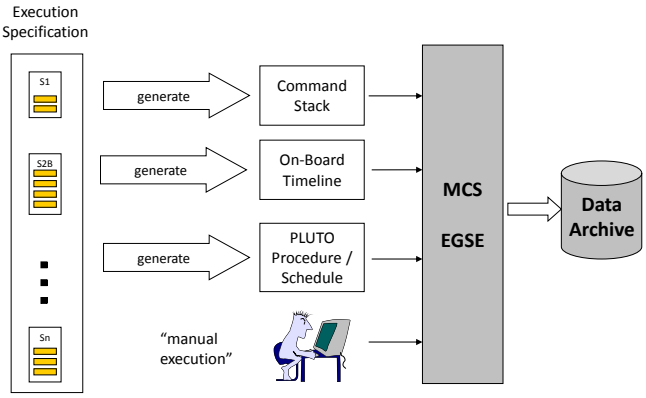

Figure 5. Procedure Execution

In all cases it is envisaged to use a Data Archive (DARC) to collect and store data for later offline validation. In order to limit the amount of data that need to be extracted and stored it might be necessary to derive focused data specifications from the validation specification of the applicable validation suite.

For the correct execution of the validation suite the space element might have to be brought into a well-defined start condition. For the simulator this can be implemented by restoring a previously stored breakpoint. If such activities shall be automated by OPSVAL a Validation Manager Component will be required that can control both the simulator and the ground control system. This component would set the simulator into the start condition by restoring a specified breakpoint and then configure and initiate execution of validation suite execution depending on the implementation option selected. In case of an SVT the required start condition will have to be agreed with the spacecraft manufacturer.

D. Validation Concept

Validation is performed by a tool capable of reading the validation data for a session from the data archive and executing the rules specified in the validation specification for the validation suite. This tool has to use the execution specification to locate the command sequences that were executed in order to apply the rules to the correct data. This process is illustrated in "Fig.6" which outlines the two steps to be performed for every command sequence: searching the command sequence in the execution trace and then performing the checks defined in the validation specification.

In order to locate the data in the execution trace, the start time of the validation session has to be identified to the validation process, as also indicated in the figure.

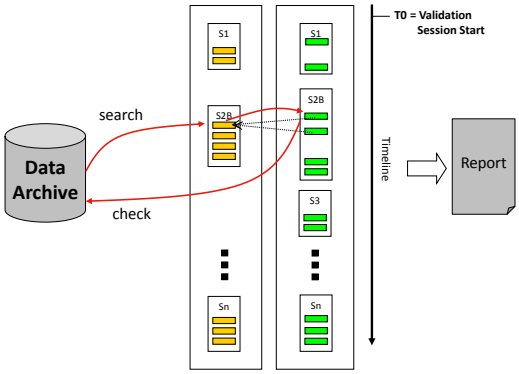

Figure 6. Validation Concept

The result of the validation is documented in a report which provides as much information as possible for any detected anomaly, i.e. violation of any of the specified rules.

IV. OPSVAL System

A. Overview

The OPSVAL system includes three main components as illustrated "Fig.7":

1) *Test Preparation System*

The test preparation system is used to define the test suites and test data than will be run to validate the procedures. This system allows definition of test suites indicating which operational procedures shall be run, allows definition of the rules that will be used to validate the procedures, and allows visualization of the operational procedures. The test preparation system generates 2 main outputs, the execution specifications and the validation specifications. From the execution specifications, the preparation system is able to generate command stack file for manual execution of command and command sequences, or PLUTO scripts for automatic execution of command and command sequences.

2) *Validation System*

The validation system is an offline system which validates the operational procedures after they have been run according to the execution specification. The validation system reads data collected during execution via the execution trace extractor, which can interface to the OPSVAL Data Archive system or other systems. The validation system is composed of a validation engine (rules based engine), a validation model composed by the rules defined by the validation specifications, the execution trace extractor, and a reporting component generating reports for the end users.

3) *Data Archive system*

The OPSVAL Data Archive collects during test execution the data needed for the validation, and makes them available for later inspection. The following data are stored and available for retrieval: telemetry parameters, packet reception notification, event notifications, command status, and log messages. Two ESA data archives are supported: the DARC and MUST.

The execution system is not considered as an element of OPSVAL. OPSVAL relies on the fact that an external system will run the procedures according to the execution specifications.

The complete system has been designed as a modular system, and provides adapters allowing easily adapting the system for usage of different archive system, spacecraft database system, or execution system.

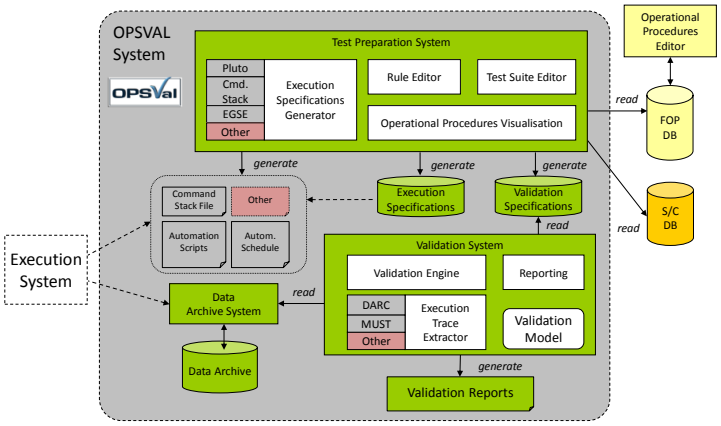

Figure 7. OPSVAL System Overview

B. Test Preparation System

The responsibility of the test preparation system is to prepare a specification of how an individual procedure can be validated. To accomplish the task, the procedure (or set of procedures) must first be imported. OPSVAL can import procedures stored in SOLM (Spacecraft Operations Language Metamodel) XML format and extract all relevant information, i.e. telemetry checks, sending of commands, packet reception checks.

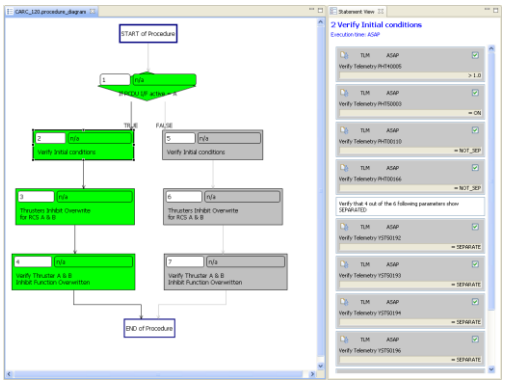

Figure 8. OPSVAL Preparation System

While the procedures are imported, OPSVAL checks their validity, and in addition check their consistency with the spacecraft database. When the procedures are imported, it is possible to defined test suites, i.e. define a set of procedures to be scheduled for execution and validation, choose which branch from each procedure must be executed, provide the values of input parameters if any, and automatically generate validation rules out of each single procedure, according to telemetry and packet checks. The validation rules can be fine-tuned by the user, to achieve a higher level of precision in the validation results.

For each procedure selected for a test suite, validation session "set-up" and "shutdown" script can be configured, such allowing to set the runtime environment in a specific state before the test and after. This can be used for instance to set-up a simulator in a specific state before running a procedure.

At the end of the process, execution and validation specifications are generated out of a test suite specification. The former are used by the execution system to generate an execution trace; the latter are used by OPSVAL Validation component to enforce selected rules against a provided execution trace.

C. Validation

OPSVAL performs validation using a validation model which is defined by validation rules. After the execution trace has been generated and stored, validation can be carried out. At the beginning of the validation, validation specifications and execution start time and end time are provided. The execution trace is then extracted from the archive, and processed. OPSVAL offers the possibility to run several times the validation session on the same execution trace, to allow for instance changing the validation rules and re-validating them on the same execution trace.

1. Validation Model

The OPSVAL validation model is centred on the ECA (Event Condition Action) concept. Event Condition Action is a short-cut for referring to the structure of active rules in event driven architecture and active database systems. Such a rule traditionally consists of three parts:

1) The *event* part specifies the signal that triggers the invocation of the rule.
2) The *condition* part is a logical test that, if satisfied or evaluates to true, causes the action to be carried out.
3) The *action* part consists of updates or invocations on the local data.

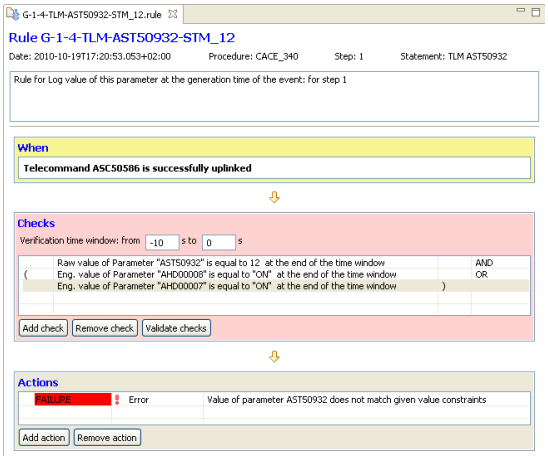

Figure 9. Validation Rule

The following events can be detected by OPSVAL from the execution trace retrieved from the data archive: command acceptance, command uplink, command execution, parameter value changed, packet reception, on-board event notification, out-of-limit, log message.

An example of validation rule, shown "Fig.9", is: When the telecommand <xxx> is successfully uplinked (*action*), check 10 seconds before the command is uplinked that some telemetry parameters have specific values (*condition*). If not, raise an error (*action*).

The validation process inspects all supported events between a given time window to see if any of the defined rules can be triggered. All rules triggered by the event are evaluated, i.e. the condition checks are performed, and the actions are eventually executed.

During the validation process, OPSVAL automatically loads (and un-loads) the rules that need to be checked according to the procedures/steps being validated. OPSVAL uses the JBoss Drools rule engine. The results of all validated rules are stored for later reporting (the system does not stop validation upon rule violation).

2. Validation Rules

A validation rule is a check of one or more conditions, which must be verified when a particular event is detected. Validation rules are specified during the test preparation phase and, at the end of the test suite preparation, are exported in the validation specifications.

Validation rules are divided in two main groups according to their validity level: rules valid at global level, i.e. valid for every procedure defined in every test suite, and rules valid at procedure level, i.e. valid only for a specific procedure. Within these two groups, the rules are further divided. Global rules are divided in implicit and explicit rules:

1) *Implicit rules* are automatically generated by OPSVAL and always apply. They enforce basic checks for all procedures (OOL check, log messages having severity state error or alarm, onboard events having severity state error or alarm). Implicit rules cannot be modified by the user, but may be de-activated for some operational procedures.
2) *Explicit rules* are mission specific rules and must be explicitly defined, automatically or by the user. Except for their definition, they are used by the OPSVAL system in the same way. They can be created, modified or deleted at any time.

Also procedure rules are divided in procedure-global rules (can be added by the user to a specific procedure defined in the test suite, and are valid throughout the entire procedure validation) and per-statement rules (rules attached to a specific statement of a procedure - Per-statement rule can be triggered only during the execution of the command sequence the statement belongs to).

D. Validation Results and Reporting

At the end of the validation process, the validation report is generated from the validation results. This report can then be inspected by the user in two forms, a summary report providing the list of procedures which have been validated and the validation status, and a detailed report providing the details of all the detected events, all rules validated, and possible errors. The detail reports allows the user to check in details all the TM/TC data collected by OPSVAL from the execution trace within the timeline.

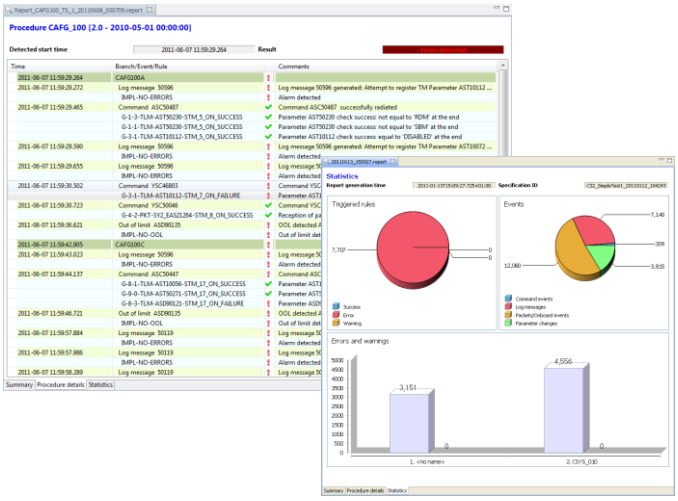

Figure 10. OPSVAL Reporting

From the reports, the system provides the ability to navigate through the managed data; For instance it is possible from an error reported in the report to navigate back to the violated rule and the corresponding procedure step via the validation specification. Also the system provides the ability to edit the reports for instance to insert annotations to set an investigation result where applicable. Statistics are also provided, and the report can be exported in html format.

OPSVAL also provides report filtering capabilities since the collected data may be huge. The filtering allows for instance to select which events shall be displayed, which rule execution status, or to filter on specific string.

Finally, report comparison is provided allowing comparing summary reports between two summary reports.

V. Results & Benefits

The OPSVAL system development was completed early 2012 and used in operations for concept and system validation. Based on these tests, a first conclusion could be drawn which need to be confirmed in the next months, and which is presented below.

Table 1 – OPSVAL Results & Benefits

Objective	OPSVAL Results & Proven Benefits
Common tool used and shared across Agencies and Industry.	OPSVAL modular architecture allows easy integration in different context and adaptation to specific systems. Development of specific adapters to a different archive systems and execution systems have been implemented and optimized in very short time. The system can be run in complete independence from the execution system. Dependencies between the OPSVAL components are reduced by the definition of well-defined interfaces (execution and validation specifications). OPSVAL import/export functions allow interfacing with external systems via standard well-defined interface which are not OPSVAL specific. It is expected that OPSVAL need adaptation to interface to specific data archive,
Usability.	Close cooperation with the first users and an Agile development process allowed providing a clean integrated and optimized OPSVAL MMI in order to ease the operations of the system. The MMI uses standards Eclipse mechanisms (projects, views, editors), such allowing users familiar with Eclipse application easy operations. Configuration of OPSVAL was streamlined and reduced to the minimum, with automatic configuration where possible (automatic rule generation for instance). Installation of the system is very easy, the system is platform independent. The validation engine provides useful explanation on violated rules, such allowing providing explanatory reports. There is a clear separation between the data used for validation, and the rules. Therefore, the rules can be established and maintained relatively independently from the validated data. For offline validation, the system can be used with general automatically generated rules. In such cases, configuration and operations of the system is straight forward and very easy. The system is flexible and allows the users to define their own rules.
Use for current flying missions.	OPSVAL could be used and tested with several missions flying since several years, for offline validation of operations. In this context, OPSVAL can be used with or without procedures. In this later case, OPSVAL allows validation of offline data and detection of any problem using specific rules entered by the FCT. Import of procedures not prepared for automation is problematic, since most of the procedures are formally not correct. Therefore, these procedures cannot be used for automatic rule generation or even cannot be exchanged via external standard file format (SOLM). Cleaning these procedures is a tedious work.
Use for new missions.	Usage of OPSVAL for the development and validation of the operational procedures should be decided as early as possible to really fully benefit from the provided function. This enforces writing correct procedures at development time, and allows performing regression testing (including testing of operational systems of the ground segment) at all stages of the development process. This allows not only testing the FOP but also the systems constituting the ground segment (MCS, MPS, simulator …) and their configuration (including the spacecraft database).

Objective	OPSVAL Results & Proven Benefits
Increase the level of testing.	OPSVAL allows to really test the procedures effects and its correctness via definition of powerfully rules, and to check for un-expected effects. Not only the checks defined in procedures are tested, OPSVAL allows testing that the procedure performs the intended actions over time periods, and always checks for un-expected conditions. The OPSVAL validation engine - Rete algorithm used within Drools - has good performances - the system can validate 60000 facts (parameter updates, packet receptions, command releases/executions, log messages) per minute - and is well suitable for operational procedures testing.
Reduce time and cost spent on FOP development and testing.	It is acknowledged that using OPSVAL adds some effort in order to install and configure the systems, and to define the validation suites for the mission. However, it is expected that this will pay off in the future by allowing automated testing and performing much more regression testing than usually done.
Maximizing the reuse the existing software infrastructure.	OPSVAL reuse existing systems and can interface to them, for instance the data archive, execution system, procedure editor. OPSVAL is independent from these systems.

In addition, out of the initial testing, the following recommendations are made for the future

1) *Procedure Editor*.
   Editor of operational procedure should be drastically improved in order to enforce provision of syntactically and formally correct procedures, and to allow easy exchange of procedures between the different parties involved in the development of a mission.
2) *Data Archive System*
   Effort should be made to enforce usage of a single archive system across several missions within each entity. A common generic system should be used for several missions, and should store the same basic set of data for all missions. Initial testing has shown that not the same set of data is recorded by the different missions, and that different archive systems with different internal structure and API where used.
3) *Integrated Tools*
   There is a need for integrated tools able not only to test operational procedures but also the systems building the ground segment. These tools are today becoming mandatory to support the increasing effort spent on regression testing, and to ease testing with systems of increasing complexity. Automation systems will become more and more important, should be easy to use and configure, and should provide efficient interface allowing controlling them from the outside - from OPSVAL for instance.
4) *OPSVAL System*
   OPSVAL is a new system already providing satisfactory function, and the first validation results show its usefulness. It is expected that further usage of the system will allow identifying improvements that need to be implemented in the future.

Appendix A
Acronym List

AIT	Assembly Integration & Testing
AIV	Assembly Integration & Verification
CRP	Contingency Recovery Procedure
DARC	Data Archive
ECA	Event Condition Action
EGSE	Electrical Ground Support Equipment
FCP	Flight Control Procedure
FCT	Flight Control Team
FOP	Flight Operations Plan
LEOP	Launch and Early Operations Phase
MCS	Mission Control System
MPS	Mission Planning System
OBSW	On-Board Software
OCC	Operation Control Centre
OL	Operation Language
OOL	Out Of Limit
OPSVAL	Model Based Operation Validation System
SVT	Spacecraft Validation Test

Appendix B

References

[1]JBoss Drools – The Business Logic integration Platform, Drools Expert (rule engine), Software Package, Ver. 5.0.1, URL: http://www.jboss.org/drools