Ariane 6 Simulation Infrastructure to Validate the Automated Procedures Software

Workshop on Simulation and EGSE for Space Programmes (SESP)
26 - 28 March 2019

ESA-ESTEC, Noordwijk, The Netherlands

Eduard Huguet[(1)], **Sílvia Navarta**[(1)], **Sergi Inglada**[(1)]
Raphael Methot[(2)]
[(1)]***GTD Sistemas de información SAU***
Pg. Garcia Fária 17, 08005 Barcelona, Spain
Email: eduard.huguet@gtd.eu, silvia.navarta@gtd.eu, sergi.inglada@gtd.eu

[(2)] ***Centre national d'études spatiales (CNES)***
2 place Maurice Quentin, 75 039 Paris Cedex 01, France
Email: Raphael.Methot@cnes.fr

INTRODUCTION

The concept of "Control Bench Family" (A6-CBF) was agreed by ESA, Ariane Group and CNES as the most efficient way to develop the Ariane-6 checkout and control command systems and comply with high level requirements of the Ariane-6 programme while focusing in development and recurrent costs reduction.

In the development framework of Ariane 6 Control Bench Family (A6-CBF) a new need was expressed concerning the Automated Operations Procedures (AOP) software validation means. These are the software procedures in charge of the automation of the validation and launch operations procedures and called Application Software. More precisely, Application Software Procedures validation platforms are to be delivered together with certain validation tools, providing automation and easing and guaranteeing the validation process.

In order to deal with this need, a specific bench named VADOR was designed. The new bench is mostly composed of pre-existing "Core" components such as the Process Simulator (developed in the frame of A6-CBF), coupled with some new specific components providing the additional features required to deliver a complete, fully-automated validation platform for the Automated Operations Procedures software development.

With the objective of keeping space launch operation costs to a very minimum, the new functionalities have been provided mostly by integrating into the bench some existing, well-tested open source automation tools (*RobotFramework*, *Jenkins*...), with only a minimal part of them requiring specific software development. With the same objective in mind, the new bench also makes extensive use of virtualization technologies, and it integrates all non-real-time dependent components into a single virtualization platform, thus significantly cutting hardware costs while fitting all the components required into a single bench cabinet.

Moreover, the VADOR bench does not include any physical console station for operator work, as all the components are designed to be operated remotely from the customer's own workstations through their network infrastructure.

This paper describes the technical solutions used of the new bench (along with the underlying modular architecture of the A6-CBF project that made it possible), while focusing on the innovative aspect of using common, well-established software development techniques and tools (Source Code Management, Automated Testing, Continuous Integration...) for the development and validation of launch operations automation procedures.

THE A6 CONTROL BENCH FAMILY

The "Control Bench Family" defines a product family concept by setting common design principles to all the Ariane 6 programme benches on the basis of next design constraints:

- Lessons learnt from the Ariane 5 Control Benches during the development and maintenance phases;
- The need to coherently support development and exploitation of the Launch System throughout its lifecycle, covering both the test facilities and the operational ground facilities;
- The need to prepare and reproduce in Europe operations performed in Guyana with appropriate representativeness, and inversely, for investigation purposes;
- The need of a Functional Test Facility in Europe as a reference platform of both ELA4 Control Bench and the Launcher System;
- The need of including Simulators in the development of the family;
- The need of a centralized, configuration controlled and system-oriented data management system to support collection, traceability and exchange of consistent data sets throughout the actors, locations and activities involved in delivery of the A6 launch service;
- A new concept adapted to the current A6 context, imposing a core architecture and related technology;
- Recurring Cost as a development driver.

The CBF defines a full-equipped bench, the "Core", which is intended to cover all A6 benches requirements.

"The Core" is thus built on the basis of up to several functional blocks catalogued within 6 sub-systems: acquisition & command, operations control, configuration & data exploitation and simulation. Depending on the functionalities required for a specific bench, a set of blocks is assembled. Up to 8 Control Benches are identified today, each of them with a specific assembly configuration of a set of the"Core" functional blocks:

- Control Bench for Launch Pad (CBLP) for operations on the Launch Pad ZL4 in Kourou
- Control Bench for ESR Production (CBSR1) for AIT operations in Aquitaine
- Control Bench for ESR Production (CBSR2) for AIT operations in Kourou
- Control Bench Replica (CBREP) representative of CBLP and that can be configured in other AIT configurations for exploitation and training in Les Mureaux
- Control Bench for ULPM (CBUL) for ULPM AIT operations in Bremen
- Two Control Bench for LLPM (CBLL and CBLL1) for LLPM AIT operations in Les Mureaux
- Control Bench for P5.2 (CBP52) for ULPM electrical operations (avionics checks and firing test) at the DLR test bench, Lampoldshausen
- Core Control Bench for Core validation in Barcelona.

Core Architecture

As previously introduced, "The Core" is composed of up to several functional blocks which are called First Level Core Components. Each First Level Core Component: it is instantiated at some CBF bench (or all of them), it may be instantiated several times on a given CBF bench and it is self-contained and individually validated.

Component instantiation does not involve any specific development. All Component instances are identical. Software of Component instances includes specific configuration data as for example the IP addresses. Hardware of instances are physically distinct, but are produced using exactly the same definition and integration design.

First Level Core Components are decomposed into Lower Level Core Components which may be:

- A Hardware Component composed exclusively of other Hardware components
- A Software Component composed exclusively of other Software components
- A Hybrid Component composed of a mix of Hardware and Software components, and maybe other Hybrid Components.

One of the most important issues of a First Level Core Component is their interfaces which are to be precisely specified and validated:

- Software ports: define how components exchange data;
- Hardware ports: physical hardware connectors;
- Interface definitions: explain what data is exchanged between software components and in which format.

This specification is crucial as all integration and validation strategy depends on it.

Subsystems

Once components are validated, they are assembled in order to obtain the different subsystems which compose the CBF benches. Subsystems define a functional perimeter within the bench constrained by specific RAMS requirements. The most important subsystems are:

- The Operational subsystem;
- The Software and Data Office subsystem (SADO);
- The Simulation subsystem;
- The Administration subsystem.

The **operational subsystem** is in charge of providing services and tools for operating the bench regarding launcher and ground processes. It is the only one constrained by Failed Operational criteria. More concretely, the operational subsystem is the command and monitoring system which is built around a central server connected to the process via deported frontends. Frontends provide the required wiring, power, communication interfaces with the Launcher and other ground processes such as the fluidic. The interface between the central server and the deported frontends is EtherCAT, which provides a deterministic real-time protocol for commanding and acquisition. The main First Level Components that compose the operational subsystem are:

- **Operations Control (OC),** which is the central server responsible for the overall real-time treatments involving acquisition, commanding, internal calculations, reaction to process, level 3 software execution (as explained later), data broadcasting to SADO and other external systems and functional bench monitoring.
- **TTE Frontend,** which guarantees the Time Triggered Ethernet communication between the launcher and the bench. It permits to set up the launcher avionics configurations and to exchange data with them.
- **WIO Frontend,** which provides a wiring interface with the launcher in order to send electrical commands for arming/disarming electrical and optical security barriers, commanding electro valves or acquiring launcher statuses.
- **Power Supply Frontend,** which provides a wiring interface with the launcher for powering the launcher equipment.
- **DCS,** which provides an MMI to operator for configuration deployment and startup of the bench.
- **Operator Stations,** which are providing the MMI to operators in order to interact and monitor the process. Monitoring and commanding is done via the Level 3 software which defines: views, process commands, process statuses, process measures, AOP (explained in further chapters), automatons, calculation equations.
- **Operational Networks,** providing connectivity between all the previous components: LAN and EtherCat networks.

The **SADO** subsystem is in charge of providing services and tools for archiving and exploiting all data produced as well as archiving and editing all the configurations required. The main First Level Components that compose the operational subsystem are:

- **PESTO,** providing to operators the required tools and services for exploiting all data generated by the bench (basically analysis of logs and analysis and graphical representation of acquired data).
- **GitLab,** responsible for bench configuration management.
- **LN3 Editors,** permitting edition of bench configuration regarding Level 3 software).
- **SADO Networks,** providing connectivity between all the previous components.

The **Simulation** subsystem provides ground process simulation as well as other external CBF interface simulations. The main First Level Component that composes the operational subsystem is the Process Simulator, which is detailed later as it has an important role on AOP validation.

The **Administration** subsystem is in charge of providing administration tools and services to the control bench. It consists of two administration servers. One of them is dedicated to network equipment (switches, firewalls) administration and the other one to system administration (servers).
The servers are independently connected to the corresponding equipment via an out-of-band network which is physically independent for network equipment and over a VLAN on the operational network for system equipment.

Depending on the required functionalities, each control bench composes the required subsystems by assembling the corresponding First Level Components.

Automated Operations Procedures

The CBF software is decomposed in Level 1 software, Level 2 software and Level 3 software:

- Level 1 software (L1) includes software specific to hardware (mainly drivers and operational system software), as well as standard commercial programs (COTS).
- Level 2 software (L2) corresponds to programs developed specifically for the Control Bench, and providing services required for the application programs (L3 software) and operators (MMI).
- Level 3 software (L3) encloses all the operational data and products that are specific to the processes under control and to the operations performed through the Control Bench. These data and software are specific for each control bench. They can be updated in order to adapt the bench to specific campaign requirements. The L3 software is thus loaded into the Control Bench upon operator request and processed in real time.

A crucial part of the L3 software are a set of Automatic Operations Procedures (AOP): these are "software modules" created by the operator within a dedicated edition environment which, once loaded into the bench as part of the L3 software, are used for automatic chaining of control operations for ground facilities and launcher. More concretely, an AOP is a set of instructions (acquisitions and commands), sub functions and modifiable parameters that interact with ground process subsystems.

Execution of AOPs is performed by an operator on MMI. When an AOP is being executed, a FRPA file is created in order to record log information of the AOP. Once the execution is finished (regardless of whether or not an error occurred), the AOP can be either run once again or be unloaded. On each execution, a new FRPA is created. FRPAs can be later analyzed by an operator on a dedicated log exploitation environment, provided within the bench perimeter, in order to assess the correct execution of the AOP. Moreover, validation of AOPs is to be performed on the bench itself. The edition environment provides integrity and coherence control, but not functional validation.

In order to provide automation to this validation process, a process simulator has been developed together with an automation manager, both assembled into an L3 validation bench, the VADOR bench.

Process Simulator

The so-called A6-CBF Process Simulator enables simulating the whole Launcher Ground Process by emulating the EtherCAT real-time networks and their connected frontends. It can exchange real-time data with the Operation Control, as well as other external simulation, under several conditions and test configuration, for validation and training purposes.

The simulator can reproduce not only the nominal behavior of the simulated equipment, but also emulate the degraded modes of the Ground process equipment, in order to validate the automatic procedures behavior against anomalous situations.

Technology

The A6-CBF Process Simulator is based on TwinCAT 3.1, a Beckhoff commercial solution, which provides a customizable Real-Time environment where simulation runs. The figure below (Figure 1) describes the blocks composing the A6-CBF Process Simulator:

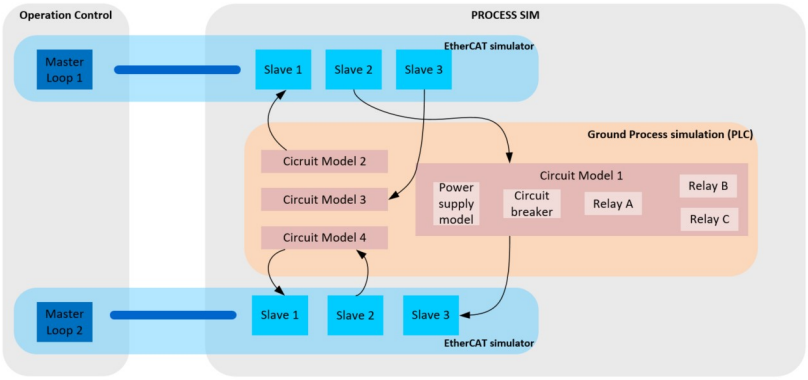

Figure 1: Simulation project block diagram

As displayed in the diagram above, simulation project is composed of the following elements:

- The *simulated EtherCAT segment*:
  This is a TwinCAT functionality that reproduces the EtherCAT features for an EtherCAT network. The EtherCAT Network Information (ENI) file is used to configure the mirrored process image. It is linked to the simulation elements in order to provide the simulation environment with the real-time data.
- The *Process simulation*:
  This a PLC project, it follows the same A6-CBF modelling approach, it contains a collection of low level elements like sensors, actuators, electric relays, power supplies and valves that are instantiated in order to build high-level circuits elements that reproduce the Ground & Launcher Process equipment used on the launch site.

With the objective of satisfying the different validation and training necessities, the simulation projects can be edited inside the Process Simulator, in order to compose a simulation solution including both the desired set of EtherCAT segments and an associated Ground Process simulation featuring the circuit elements being simulated. This configuration can be easily modified in order to simulate any specific bench.

Simulation Scenarios

The simulation project provides the Operation Control (OC) with the nominal behavior response to any commanding and acquisition action, performing in the same way as the real Ground Process equipment would perform.

However, with the aim of testing and validating the degraded-case management from the Operation Control, it is possible to act over the simulation models to provoke an expected malfunctioning into the simulation; and this way, providing the Operation Control with an unexpected behavior response which enables the validation of automatic procedures. This malfunctioning is directly applied into the low-level elements, and action consequence is forwarded to rest of the circuit which belongs. Several actions can be grouped into simulation scenario files, where sequences of actions are scheduled to be automatically performed at their predefined time.

External Simulators

There is also the possibility of expanding the simulation scope by exchanging some simulation parameters to an external simulator, which allows the testing of alternative or complementary behavioral models without the necessity of modifying the bench configuration. In this situation, the CBF Process Simulator simply acts as a “bridge” to the external simulator, which is the one providing the real simulation data.

THE VADOR BENCH

As mentioned, the purpose of a VADOR bench is to be able to perform manual and automatic validation of the AOP procedures and scripts, which are grouped under the common name of “L3 Operational Configuration” and stored into the bench GitLab-based repository.

AOP procedures are currently being developed in parallel by the A6-CBF final users (ArianeGroup, CNES), in preparation for the initial AOP configuration to be used for the first Ariane 6 launch (planned for 2020). After this, AOP configuration is expected to be evolved for each launch mission, either because of adjustments applied after previous launch experiences, or to adapt it to the specific particularities of the mission.

As these AOP procedures and scripts are, in the end, nothing more than software procedures and modules, not very different than any other software product. The basic idea is to be able to use the existing standard software-validation tools and patterns that have been successfully been used for years in software development. Such tools and patterns are, for instance, *RobotFramework* for test automation and a *Jenkins* server for Continuous Integration build.

Operating principle

The operating principle of VADOR relies on the usage of *RobotFramework* (RF), a very popular generic test automation framework for acceptance testing and Acceptance Test-Driven Development (ATDD). It provides a very powerful and flexible test execution engine, including a rich ecosystem around it consisting of various generic test libraries and tools that are developed either as part of RF itself or as separate projects. RF is open-source software released under Apache 2.0 License.

Moreover, one of the key features of RF, in the context of the VADOR bench, is that its testing capabilities can be easily extended by creating new higher-level keywords from existing ones (using the very same syntax as the one used for test cases), or even through external test libraries implemented using high-level programming languages such as Python or Java.

This means that as long as A6-CBF bench components can be externally controlled through any kind of software API, it is possible to create a dedicated controller RF library providing test keywords for it, and then write automation tests making use of these new keywords to write specialized tests suitable for AOP validation. It then becomes possible to fully automatize the testing of the whole AOP suite and integrate this testing process into a Continuous Integration & Delivery chain using a Jenkins server.

Test Libraries

In order to automatize AOP software validation, the testing engine needs to be able to operate the bench autonomously, simulating the way a human operator would interact with the bench when doing manual testing.

To accomplish so, a number of RF libraries have been written to provide the RF engine with the necessary interfaces to fully control the bench. These libraries provide the RF keywords that are later used by the tests scripts to simulate the operation of the bench.

The table below (Table 1) summarizes the test libraries specifically created for VADOR bench at the time of writing, their purpose, and the language used for coding them (the list is not closed, as new libraries might be added in a future in case they are found necessary):

Test Library	Language	Purpose	Status
OSI Library	Python / RF	Handles Operator Station Interface daemon, in order to send operator commands to the OC.	Done
TM Library	Python / RF	Handles bench time manager: start / stop / set TD clock.	Done
PSIM Library	Python / RF	Handles Process Simulator: start / stop simulation, set simulation scenario.	Done
DSC Library	TBD	Handles bench start / stop and operational configuration deployment.	Planned

Table 1: Test libraries created for VADOR bench

For all VADOR test libraries, Python has been used for low-level keyword coding (like TCP socket access, buffer manipulation, i.e.), while higher-level keywords have been coded using RF language (usually by combining calls to lower-level keywords coded in Python).

Test Files

RobotFramework has easy-to-use tabular test data syntax, and it utilizes a keyboard-driven testing approach. However, even if the RF language can be considered fairly simple and intuitive, which can be helpful from a developer's perspective, the RF isn't a truly helpful tool for less-technical personnel such as the intended target users.

Moreover, given the complexity of control bench operations (where many steps need to be performed in the right order and at the right time in order to assure the correct behavior of the bench), it was considered that letting the users directly write down RF test scripts based on the test libraries provided would pose an unnecessary risk, given the intrinsic difficulties inherent to test case specification.

In consequence, it was decided that the best approach would be to define a test case format specific for VADOR, which is XML-based and editable through a tool specifically developed for this purpose (*VADOR Test Editor*).

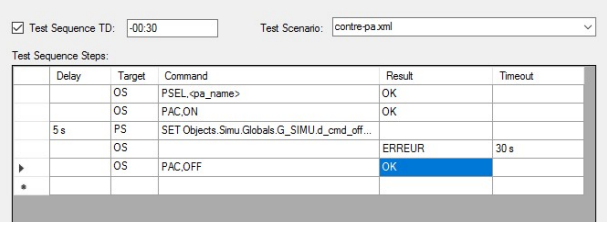

Figure 2: VADOR Test Editor screenshot (fragment)

With the help of this tool, test designers can easily build AOP tests by simply selecting the conditions to apply (TD startup, simulation scenario…), plus the commands to be sent to the OC using a simple form-type GUI, and without the need of struggling with the quirks of RF test case syntax. The *VADOR Test Editor* has been coded in C#.

AOP test sequences are built using the same commands that are used on the "real" Operator Stations, and for each sequence it is possible to set up different environment conditions such as the TD time to set bench clock to, or the process simulation scenario to use for degraded case testing. The possibility of the customer writing directly RF test scripts based one the test libraries provided is left open as an "advanced use case" possibility, however.

Test Execution

VADOR test execution works by automatically converting the XML test files, created using the *VADOR Test Editor* tool described in previous point, into generated RF test scripts that make use of the test libraries also described in previous section. The figure below (Figure 3) describes the workflow of VADOR test execution:

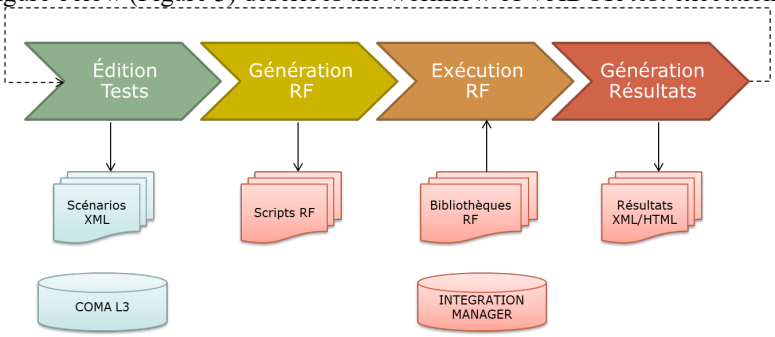

Figure 3: VADOR test execution workflow

Test execution is performed by the *VADOR Integration Manager* component, which takes each VADOR test file passed as an argument and generates an intermediary RF test script (using regular RF syntax) that performs the test operations configured by the user.

This intermediary test file is based on the VADOR test libraries described in previous section, and it performs all the steps required to operate the bench and perform the test in the appropriate order.

This RF test file is executed through *RobotFramework* engine, which generates report and log files in XML and HTML formats. These output files (specifically the XML output file) can be later picked by any external tool or plugin supporting RF (i.e. the Jenkins RF plugin) in order to build a completely automated AOP validation platform.

Operator Station Interface

The *VADOR Operator Station Interface* (OSI) is a key component of VADOR that acts as a simulated operator station for *VADOR Integration Manager*. It attaches to the *Operations Control* (OC) and interacts with it in the same way a real human operator would do through the physical Operator Station.

It is implemented as a Java daemon that receives the commands to execute through a socket channel, and forwards them to the OC using the core-level interface between the OC and the *Operator Stations*. Command results from the OC are also picked up and forwarded back to the *VADOR Integration Manager*, which validates the correct execution of the commands.

Use Cases

The following is a compendium of the different use cases identified for the VADOR bench.

Automated Nominal Case Testing

The primary use case for VADOR tests is validating the behavior of the AOP procedures against the model implemented by the *Process Simulator* and connected to the OC via EtherCAT links. Without any simulation scenario applied, this model behaves like the real bench control equipment behaving nominally.

Automated Degraded Case Testing

Degraded cases in the AOP procedures can be tested through the use of "simulation scenarios" on the simulator, which override the default behavior of the model in order to simulate different scenarios, as for example bad behavior in a component (i.e. a valve not opening when commanded to).

Automated Real Equipment Testing

As VADOR is in fact instantiating the very same components that are used in the "real" benches, it is actually possible to get rid of the simulators, and connect and test instead real hardware components by attaching them to the OC, either directly through the EtherCAT links or through the TTE frontend.

The latter is actually the use case in operation for the VADOR bench located in Ariane Group facilities, where AOP procedures will be tested against real TTE equipment on VADOR.

Automated Regression Testing

VADOR test scripts are meant to be stored along with the AOP procedures into the GitLab L3 repository, in order to make them evolve in parallel to the AOP software, in the same that way unit & functional tests are usually developed with application code in regular software development. In the long term, this means that it will be possible for VADOR customers to maintain a battery of tests for the whole AOP software suite, which will allow them to quickly run regression tests whenever changes and evolutions are made to the AOP baseline.

Continuous Integration

All previous use cases can also be simultaneously combined by using either the Jenkins CI server deployed in the bench or another external CI server: VADOR test tools are intentionally made command-line based, so they can be scripted and easily integrated into a CI chain. Moreover, test output files are regular RF output files that can be easily integrated into most popular CI engines through the use of plugins, for easy display and decision-making.

Manual Testing

Last but not least, VADOR features 5 virtualized Operator Stations with the same application software that the real, physical operator stations, so it is possible to perform all sorts of manual tests like it would be on a real bench.
Also, it is possible to use the virtualized operator stations as a "supervision" platform during the execution of the tests, by displaying i.e. the synoptic associated with the controls under test.

CONCLUSIONS

Current space simulation tools and environments do cover most of the needs of space systems simulation. However, in some cases they do not fully fit with specific simulation needs in the space industry. This is the case of launcher process simulations that typically need specific developments. This project demonstrates that based on industrial standard tools, COTS and environments these needs can be covered. The use of industrial COTS drastically reduces costs and scheduling. The space industry can take advantage of development investments managed by much bigger markets and also because the learning curve is drastically reduced and then open competition to many industries in the space field in benefit.

The CBF and VADOR developments have demonstrated the feasibility of using industrial COTS and integration of existing components to develop a very powerful testing and validation environment fully compliant with launcher simulation needs (launcher and ground process), and this principle may be certainly extended to other equivalent domains with specific needs. This new bench, in turn, was made possible because if the flexibility provided by the modular architecture of the CBF project, as it was not included in the original plan.

The VADOR bench, in its turn, presents a really innovative and fresh approach to Automated Procedures software development, as it delivers a very compact, flexible, remotely-operated turnkey bench solution, providing both development environment and fully-automated mission validation capabilities at minimal cost. In addition, the same principles demonstrated in the CBF and VADOR systems could be extended to other cases where such specific needs are to be addressed.