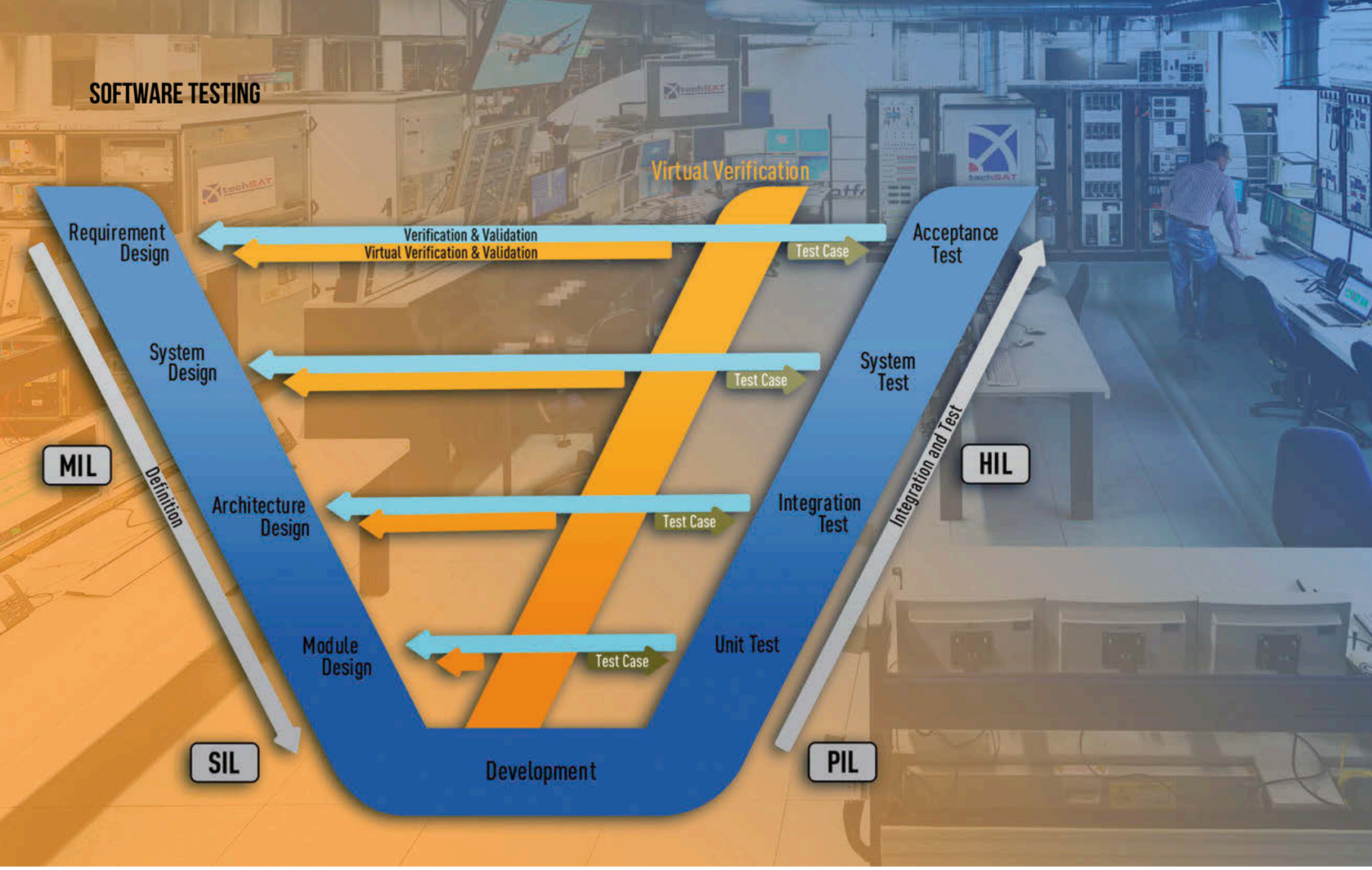


STANDARDIZED AND SCALABLE TEST SYSTEMS

Improving efficiency and reliability in the integration and testing of cyber-physical aviation systems through a standardized test environment

// **OTTMAR BAUER**

Integration, testing and verification of avionics systems is often a grueling battle when it comes to configuration and tools. Tests at different stages of development are separated from each other, the setups of the different manufacturers of tools may be incompatible. Schedules cannot be adhered to and costs are unnecessarily driven up.

This is reason enough to develop an open standard for test systems and tests. The Virtual Hybrid Testing Next Generation project (VHTNG), has seen the German and French authorities, manufacturers and users of avionic test systems join forces to develop such an open standard. They are supported by publicly funded projects in Germany by the Bundesministerium für Wirtschaft und Energie (BMWi) and in France by the Direction générale de l'aviation civile (DGAC).

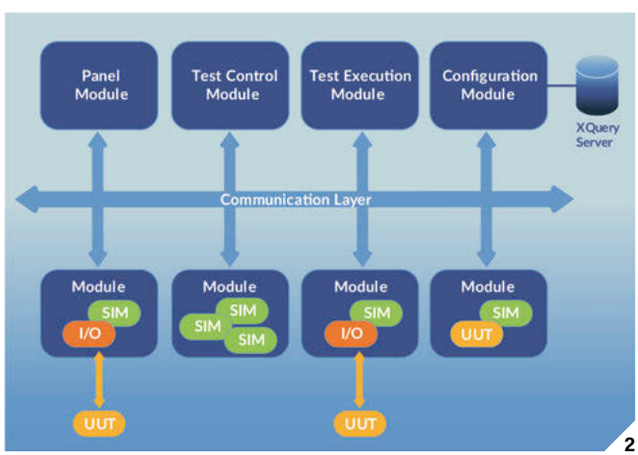
This standard covers the configuration of test systems, the control of the systems and the data exchange between the systems, as well as a uniform simulation interface model and a standardized test script specification. The requirements include the possibility of virtual testing and hybrid testing with reusability of the tests at all development stages from model-based testing up to system verification and validation. Fault insertion, real-time data exchange, scalability and integration with product lifecycle management tools are also on the agenda.

The standard is not too narrowly defined. It limits itself to defining the minimum information that is required for the test means interconnection in order to not restrict the creativity of the test equipment manufacturers.

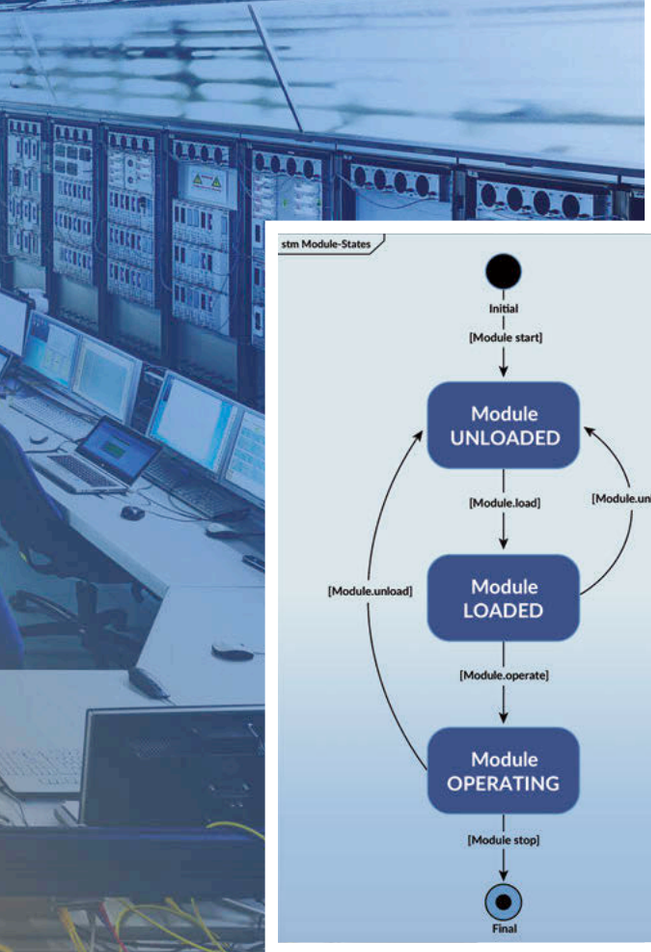
The standard defines a common model of a test system organized around a few entities and a simple behavior.

MODULES AND OBJECTS

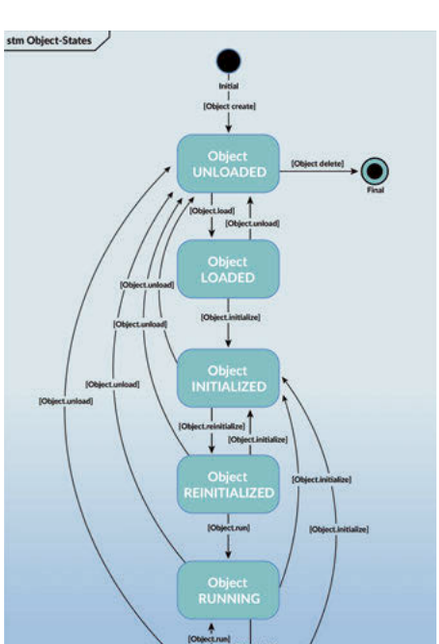
Modules are the basic building blocks of a test system. They communicate with each other and they hold objects in a hierarchical structure, which implement the function. A module can be provided by a single computer but also a complete function integration bench with several real-time computers and display computers can be handled as a module. There are specialized function modules, such as a



2



3



4

1

1 // Better testing tools by using standards to simplify certification through virtual testing and reuse of tests

2 // The structure of new test system standard with key modules and objects

3 // The module states are used to control the system from startup to operating phases

4 // The well-established simulation states are applied to all object types

test control module, test script control module or configuration module, as well as generic modules that provide simulation models and interfaces for connecting hardware as objects with standardized interfaces. Every module must provide a common control interface and pass the commands to its objects.

The test control module lets the user select a configuration and loads the modules included in this setup by sending the configuration URL. The modules to load fetch their referenced configuration from the configuration module which provides access via the XQuery language.

Modules that host simulation model objects or offer I/O interface objects run on headless real-time computers. Besides these statically configured modules, there are also test execution modules and test runtime objects as well as fault insertion objects all of which can be instantiated dynamically as required.

The modules and objects operate according to module and object state-machines. The module state reflects the load and operation state. The variable exchange mechanisms are functional if all modules are in operating state. The objects within the modules may be commanded to have synchronous state changes but it is also possible to command the states individually.

The configuration of a set of modules in order to run a test is a key aspect of this standard.

CONFIGURATION

Test benches, now modules, from different providers shall be configured by a single configuration. For the configuration data exchange format, XML with well-defined schema descriptions is chosen. All test system configuration items are modeled with UML.

The input of the configuration is given by the system hardware description, the Interface Control Document (ICD), the description of the test means, the simulation model descriptions and a test setup description, which connects these components with each other.

To avoid OEM specific formats a new Interface Control Document format was introduced. The complete aircraft configuration is held in a single XML tree. Each piece of equipment is described with functions and parameters and the used ports. The hardware interfaces are sorted by type and referenced by the ports with data direction to provide a non-redundant, consistent setup.

The Test Bench Supplier Input describes the properties of a configurable module. This includes the resources of the computers, CPU computing power and memory, and their I/O equipment and connection endpoints. To keep the standard manageable and still support manufacturer-specific features, there are named capabilities and proprietary configuration blocks.

The Model Supplier Inputs are held in zipped structures like the Functional Mock-up Unit (FMI/FMU) ZIP files. The structure contains exchange signal descriptions and model configuration as well as sources, documentation and binaries. At present, legacy simulations as well as Simulink, FMI and AMO models can be integrated. In addition to the simulation models, panels for user interaction can be configured.

All of the above are referenced in the Test Bench User Input, which represents the actual test configuration and signal mapping. The test setup modules are listed there, and the signals of the models and the test bench I/O channels are linked to one another.

A configuration tool finally creates a loadable Test Bench Configuration from the above information.

A special feature is that each configured object has its own set of variables or signals in its namespace. Variables are virtually wired together during the explicit configuration of the test setup. These links are preferably generated automatically, based on a naming convention, but can also be created manually. This procedure allows a simple exchange of real components with their virtual counterparts.

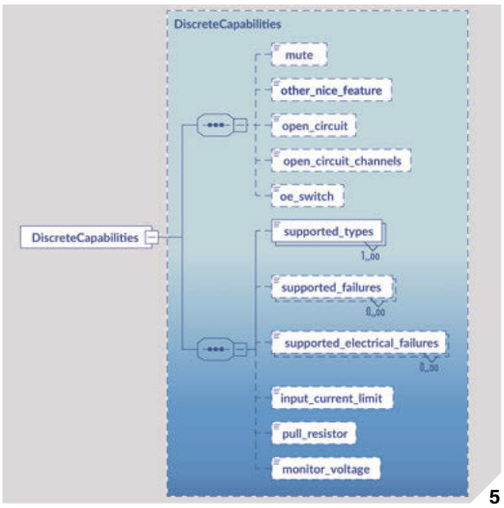
The configurations are offered by an XQuery server operating in the background of the configuration module. Only data relevant to the configured module needs to be transferred, the module itself perform the queries.

CONTROL

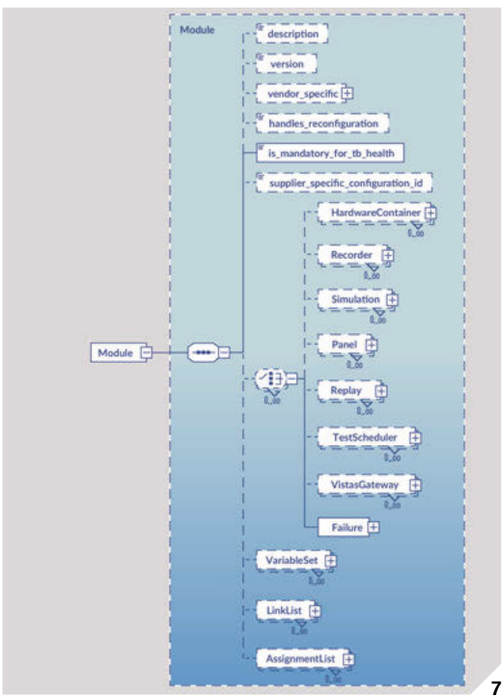
A discovery service on the Test Control module can find each module and offer it for loading a configuration referenced by an URL. The standard supports basic commands, such as load, initialize, reinitialize, run, hold and unload, which can be addressed to a complete module but also to a single object hosted on a module. Non-statically configured objects can also be created and deleted, which is usually needed for manual tests and for test scripts.

HEALTH MONITORING AND OPERATION LOGGING

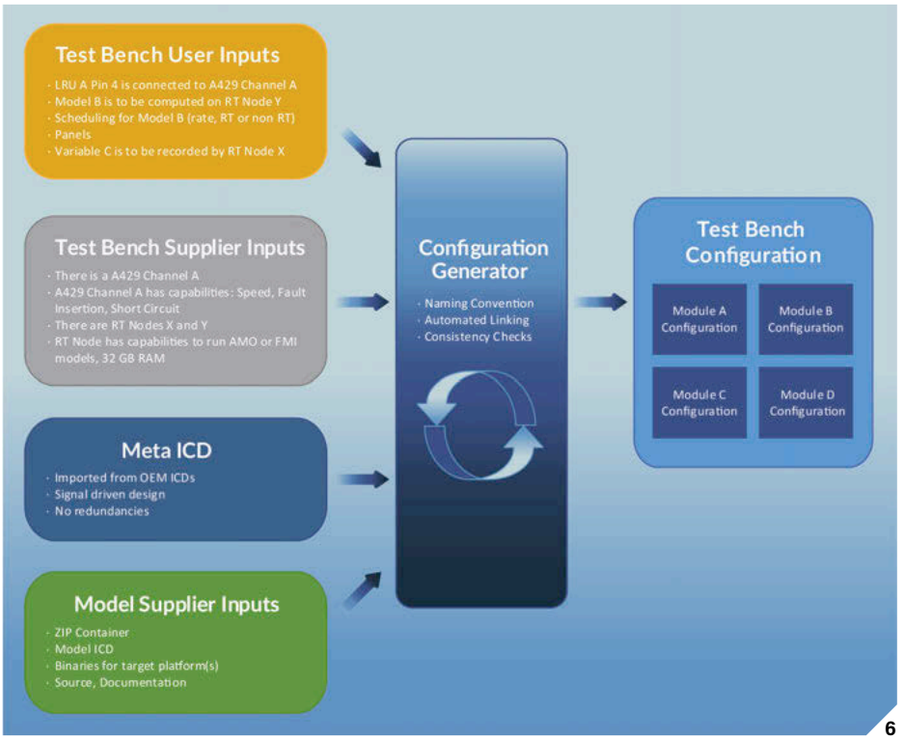
Each module collects its health and run state as well as the health and run state of the objects hosted by the module to report this information periodically. Any other module can receive these states and display them to the user or take appropriate actions. At the same time, each module sends operation log messages of different severity levels to track processes and support troubleshooting. Usually, the Test Control module with user interface takes over the task of collecting and displaying



5



7



6

this information or writing it to a file.

COMMUNICATION

Communication between the modules is handled by two protocol standards. Commands and non-real-time data exchange are handled via XML-RPC, which fits best to the request-response pattern. For the exchange of mass data in real time the Data Distribution Service (DDS) protocol was selected, following the publish-subscribe pattern. The use of the XML-RPC protocol allows simpler applications to participate in the communication, while the connection via the DDS interface provides the required data rates and quality of service for real-time applications.

A Communication Layer Reference API written in C++ is provided. This allows users to participate in communication with their own module implementations with little effort. The reference API itself is not part of the standard. It can be built either against the RTI Connext DDS or against the Adlink Vortex OpenSplice DDS implementations. The API covers variable, command and status & health exchange in such a way that the user doesn't need to know details about the XML-RPC or the DDS transmission protocol.

5 // The features of the different systems are controlled by the capability attributes in the manufacturers' system descriptions

6 // Standardized inputs are assembled to a single, consistent test system configuration, which can be used by all test systems complying with the standard

7 // The test bench configuration XML tree contains the complete setup of all modules in a test setup

TEST SCRIPT STANDARD

In addition to the standard for the test systems, a standard for test script description and execution is also being developed. The tests are written according to the State Chart XML (SCXML) specification. Signal names and execution steps are formulated generically and processed in an instantiation step for the selected test platform. This makes it possible to provide reusable tests for the different development stages (MIL, SIL, HIL, V&V) and on execution platforms of different suppliers. Another advantage of formulating test cases generically is the easier use of automated test generators from models or recorded scenarios.

The test scripts are structured into a single test procedure, the only instance which shall stimulate the unit under test, and a hierarchy of test observers. All of them can exchange events to synchronize test execution. The observers run in parallel and can be reused in a variety of test cases.

The test of avionics equipment must meet various requirements. One major requirement is the number of individual signals. A system test for a twin-aisle aircraft can easily reach over a million. Many of these signals must be transmitted in real time between dozens of computers. Demonstrations with a specially developed performance test suite show that these requirements can be met with 30,000 floating-point signals sent by each module at an exchange rate of 10 milliseconds with 10 modules involved.

In addition to the performance test suite, a compliance test suite is also being developed to test the modules and objects against the standard.

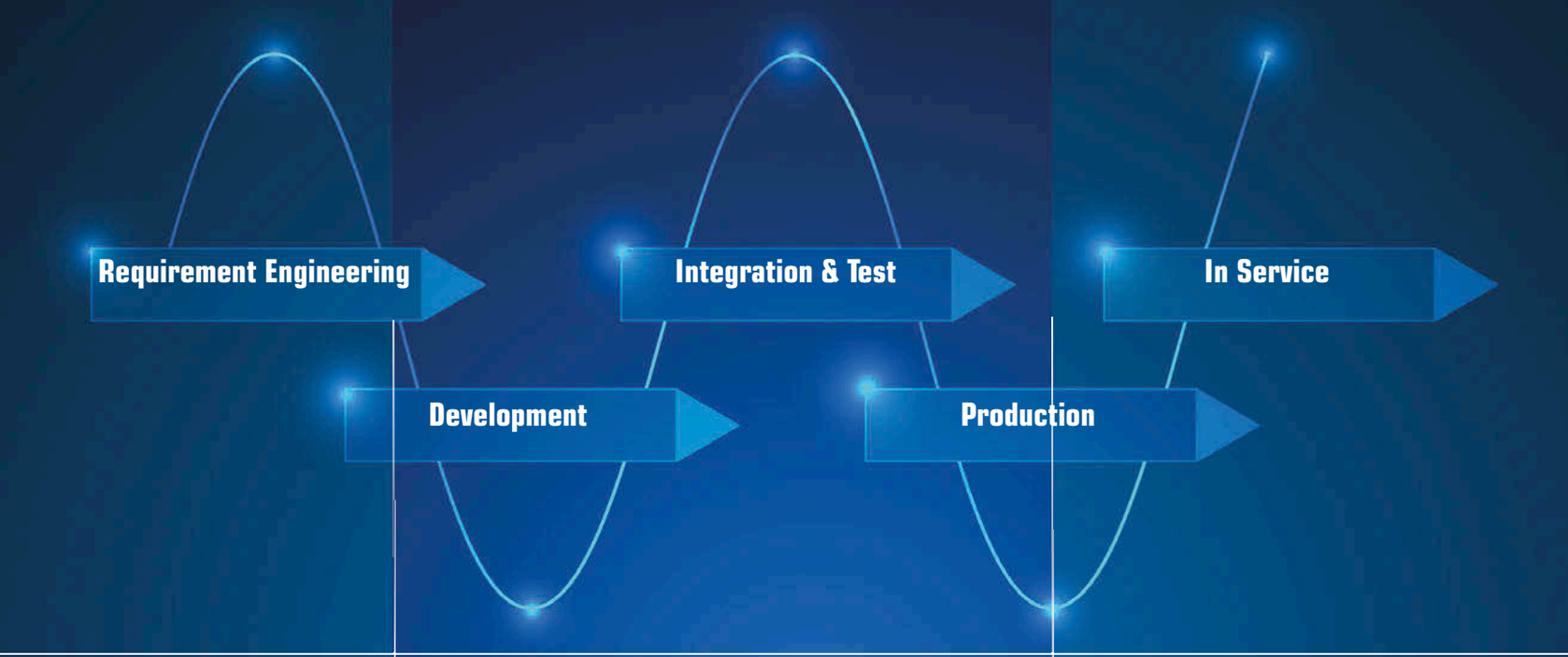
FUTURE DEVELOPMENTS

The practical use of the standard has already been demonstrated on test systems in production. The test system standard is to be released at the end of 2020. Hence, it is not too early to address the new safety challenges for commercial and general aviation posed by the development of drones and flying taxis.

The resulting standard enables the operation of highly scalable test environments. It connects systems from single-board computers, to fully-fledged function integration benches with multiple real-time computers and display stations, which have been produced by many different suppliers.

Companies from the automotive sector are also interested. In the meantime, they have become associated partners. \\

Ottmar Bauer is R&D project manager at TechSAT



MDVS Model-based Development & Verification System

VSIB Virtual System Integration Bench

SDIB Single Device Integration Bench

SIB System Integration Bench

FIB Functional Integration Bench

FAL Final Assembly Line Tester

PAT Production Acceptance Tester

IST In-Service Test Bench

The MAYA family

powered by ADS2 & TPM

We provide you with optimized solutions for integration, verification and validation for all phases of your product life cycle

ARINC • DO-160 • automated testing • ARINC • DO-330 • IV&V • DO-178B/C • DO-160

ARP4754 • DO development • safety critical • ARINC

• DO-160 • based • DAL A/B/C/D • model-based • DO-330

IV&V • DO-17 54 • development s sed

DAL A/B/C/D B/C • DAL A/B/C/D 254

development • automated testi /&V

DO-178B/C 330 • IV&V • DO-17

techSAT

www.techsat.com

Test & Integration Systems | Products | Software Solutions | Service & Support