Developments Targeting Hybrid Test Systems for HIL Testing

Dr. Andreas Himmler[1], Lars Stockmann[2], Stefan Walter[3], Sven Laux[4]
dSPACE GmbH, 33102 Paderborn, Germany

In the last years, the aerospace industry has recognized the growing need to improve integration test systems with regard to modularity, scalability, and flexibility, since new aircraft programs are becoming increasingly complex. At the same time, sharing test resources between different programs and introducing new test equipment from different sources and suppliers requires a strong effort. A generic and modular architecture for an open test environment can overcome these issues. A new architecture needs to support hybrid testing, this means testing of real and virtual systems under test and a mixture and exchange of both. Therefore, this type of architecture is used in ongoing projects aiming for new test technologies, not only in the aerospace industry but also in the automotive sector. This paper compares the requirements for a generic and modular test architecture as well as trends and solutions to achieve architectures for hybrid test systems used in the aerospace industry with the requirements for the automotive industry. This is done by investigating typical use cases of both industries. This paper also presents preliminary results aiming for a unified approach that is based on ongoing activities in both industries.

I. Nomenclature

HIL	=	hardware-in-the-loop
LTM	=	laboratory test mean
MBSE	=	model-based systems engineering
PLM	=	product life cycle management
SUT	=	system under test

II. Introduction

In the last years, the aerospace industry has recognized the growing need to improve integration test systems with regard to modularity, scalability, and flexibility. New aircraft programs are becoming increasingly complex. At the same time, sharing test resources between different programs and introducing new test equipment from different sources and suppliers requires a strong effort. The reason for this is interoperability issues caused by the high level of customization. Software and communication mechanisms are designed for a single environment.

A vendor-independent, generic, and modular architecture for an open test environment can overcome these issues (Ref. [1], [2]). Therefore, this type of architecture is used in ongoing projects aiming for new test technologies, not only in the aerospace industry but also in the automotive sector. To achieve a modular and open test environment, it is important to develop and use open standards for the interfaces of this type of architecture. The heart of the system is a standardized communication infrastructure. It forms the basis for combining test equipment of different suppliers and using laboratory test means of different aircraft programs. This not only improves modularity and scalability. It also lets developers use the best equipment for a particular task.

A generic and modular test technology needs to enable testing of virtual systems under test the same way like testing of real systems under test. This includes the need to test a mixture of both and an easy exchange of a virtual system under test by a real one and vice versa. Thereby it is possible to first perform tests on virtual systems under

[1] Business Development Manager Aerospace, Product Management.
[2] Senior Research Engineer, Product Development.
[3] Product Manager Hardware-in-the-Loop Testing Systems, Product Management.
[4] Research Engineer, Product Development.

test and replace them by real systems under test as soon as they come available, which enables early tests during development by using the same test equipment.

The functions required for configuring, operating, and controlling the test systems as well as for executing tests can be clustered into modules that are connected to the standardized communication infrastructure. This connection is established via standardized interfaces and by applying standardized data formats. The modules can be grouped into three categories: configuration and control, test execution, and test services.

Similar to the aerospace industry, the idea of having a standardized communication infrastructure has also sparked the interest of the automotive industry. This raises the question of whether the same solution and technology can be used for both. A unified solution has the potential to give users access to a widely supported, cost-effective technology. However, the technology must meet the requirements of both industries and it has to be customizable to match use cases of both. This is a prerequisite for acceptance and widespread use by manufacturers (OEMs) and suppliers alike.

This paper compares the requirements for the generic and modular test architecture as well as trends and solutions to achieve architectures for hybrid test systems used in the aerospace industry with the requirements for the automotive industry. This is done by investigating typical use cases of both industries. The paper also presents preliminary results aiming for a unified approach that is based on ongoing discussions in both industries.

Section III describes the strategic goals and high-level requirements for modular, generic test benches for integration testing. The following sections IV and V describe some specific use case that are relevant for a new test technology. The latter section refers to use cases of the automotive industry, which face similar problems than the aerospace industry. A vendor-independent, generic, and modular test architecture is described in section VI. A test technology that is currently under development based on this is presented in the subsequent section VII. This test technology is VHTNG – Virtual and Hybrid Testing Next Generation. It is the result of research projects that are led by Airbus and that involve several aircraft test bench suppliers (Ref. [1]). The goal of VHTNG is an open standard.

III. Requirements for Future Test Benches

This section describes the strategic goals and high-level requirements for future test benches. However, it first describes the current situation to illustrate the need for a new approach.

Traditionally, test setups for the aerospace industry are engineered from the ground up and specifically for one aircraft program or for the verification of specific aircraft systems (Ref. [1][2]). It is therefore common practice to design every required laboratory test mean (LTM) with a particular setup in mind. This holds specifically for integration test systems. Such specialized test equipment usually results in high development costs and high lead times because they are usually based on proprietary solutions. These solutions have a number of proprietary interfaces and data formats.

The interfaces and data formats also create issues. For example, test bench operators have to overcome severe obsolescence issues with today's test systems, which leads to high maintenance costs. It is also very costly to integrate test automation tools, connect test benches to each other, and integrate them into the company's product life cycle management (PLM) environments.

The same reasons make it difficult to share test resources between different tasks of the same program, and even more so between different test programs. It also complicates the unified configuration of test resources (e.g., using only those parts of a test system that are actually required for a test task) and connecting different test systems to a larger test system.

While the approaches to building test setups in the aerospace industry have not changed in quite some time, the complexity of aerospace systems is growing significantly. This holds for commercial aircraft as well as for military aircraft. A common way to visualize complexity is the number of lines of code required for the electronic controls on board an aircraft (ref. Figure 1, [3]). The many requirements for commercial aircraft of tomorrow are driven by demands for fuel efficiency, low noise emission, flight range, and economic efficiency. The need for networked aircraft systems also contributes to system complexity. Finally, aircraft developers and manufacturers must adhere to a variety of demanding standards and norms.

The above factors illustrate the need to make test systems for the aeronautics industry (specifically integration test systems) more modular, scalable, flexible and open than they are today. One driving force is the high level of system integration of new aircraft designs, which makes both design and verification much more complex. Open test systems become even more important with every new aircraft program, as the system complexity and the level of software complexity continues to rise.

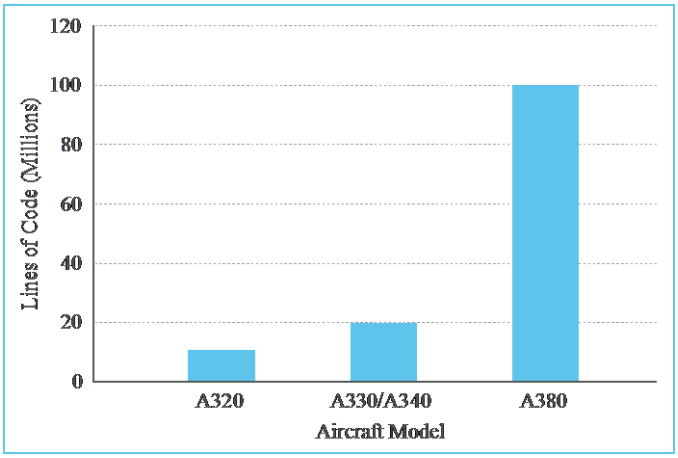

Fig. 1 Increase of software complexity on aircraft [1].

The requirements for a new test system architecture have been clustered into three categories by [1]:

1) Requirements derived from the system under test (SUT)
2) Requirements derived from the testing strategy
3) Requirements driven by commercial aspects

The systems under test range from components, subsystems and systems of different size and complexity to various system combinations. The test scope ranges from component tests at the systems suppliers up to full aircraft tests on the ground. Each test level requires a different level of instrumentation and fidelity in order to achieve the required test coverage. A future test technology must make it possible to use one test bench for the verification of different SUTs and to combine system test benches to multisystem test benches within minutes.

Test strategies can be very different: There is a clear need for fully automated tests (Ref. [4]), but manual testing also has to be possible. Additional requirements stem from other considerations, like using paper-based requirements or an MBSE approach, performing local tests or interconnecting remote test benches at different sites, using real or virtual equipment, etc. Future test technologies must provide a single solution for all of these demands.

A future modular, scalable, flexible and open test technology also has to meet commercial aspects from a customer (i.e., users of test systems) point of view. At the same time, it has to be beneficial for suppliers – There has to be a valid business model for both. Test bench customers want a wide variety of compliant test benches and test tools from various providers. Test system suppliers prefer to have a unified non-fragmented market. Therefore, a common standard-based solution for a large number of customers from different industries will gain widespread acceptance.

IV. Use Cases

This section describes use cases that have to be taken into account for a modular, scalable, and open test technology. This is in addition to the problems described in section III. The use cases described in this section are related to the consistent testing throughout the development process:

- Testing of virtual equipment prior to testing real equipment
- Intersite testing

Virtual equipment is a kind of simulation of the real equipment. It behaves in the same way, at least with regard to the functionality required for the intended test scenario. Virtual equipment is fundamentally a software model of the original device.

Testing virtual equipment is an answer to two common issues (use cases):

- The real equipment is not available yet because it is still under development. Virtual equipment can then be an alternative to build a test bench earlier, making it possible to perform verification and validation (V&V) in advance.
- The real equipment is too heavy and/or too expensive. Building a test bench would be too time-consuming and expensive and require electric power installations, dynamic loads, or other complex laboratory tools.

Both use cases call for a combination of virtual and real equipment. In the first use case, the virtual equipment must be replaced by real equipment on the same test bench. This must be as easy as possible for the test system operator. It should be possible by just changing the configuration data of a test bench to switch from testing a real system rather than the model representing the system (i.e., the virtual equipment).

Another use case coupling a test system at the aircraft manufacturer with a test system located at a system supplier. This enables functional tests in early stages of development by coupling the aircraft manufacturer's integration test system with a system suppliers test system located in a different plant. It also enables functional tests without the need to ship hardware back and forth. Functional testing can be done by coupling test systems at different locations because no hard real-time coupling is required.

For the second use case, it must be possible to combine different test systems to a new, larger test system. This process must be configured and operated from a single location, e.g., the main testing site or a remote site at a supplier. However, the coupled systems most likely have to be monitored from both locations. This coupling requires a secure network connection and the application of standard network protocols in order to avoid the costly installation of dedicated network hardware.

V. Use Cases – Automotive Industry

Similar to the aircraft industry, the automotive industry also has to address today's and future test challenges caused by the continuously increasing complexity of networked vehicle functions. The reasons are manifold: an increase in driving comfort through functions for partially automated driving; to vehicle safety with respect to other traffic participants, implemented by means of active and passive safety systems; to an increase of the efficiency of vehicle propulsion systems, which means reducing exhaust emissions and extending the range of electric vehicles. This section presents a typical prospective test scenario and the resulting requirements for a suitable test system.

The use case focuses on developing an efficient driving strategy for a hybrid electric vehicle while considering the surrounding traffic and the vehicle communication with an intelligent traffic infrastructure. The aim is to reduce the energy consumption and emissions of a hybrid electric vehicle by using intelligent driving algorithms. To obtain reliable results regarding the energy consumption and emissions of the vehicle, the results delivered by the design should be as realistic as possible. This requires a comprehensive approach in which the component and function development not only considers the context of the vehicle, but also the overall traffic and any intelligent infrastructure, such as the vehicle interaction with smart traffic lights. To provide a suitable development and test platform, a real-time-capable vehicle simulation on a HIL system is coupled with an integrated traffic and network simulation as a co-simulation. In addition to the actual vehicle movements of the surrounding traffic, communication signals of the infrastructure are also exchanged between the vehicle simulation on the HIL system and the traffic and network simulation. The traffic and network simulation can be considered a dedicated simulation, which ideally runs in a cloud environment and to which the HIL system can be dynamically couple when needed. The traffic and network simulation in the cloud runs in 24/7 operation and is continuously available to other potential test systems for use (Ref. [5]).

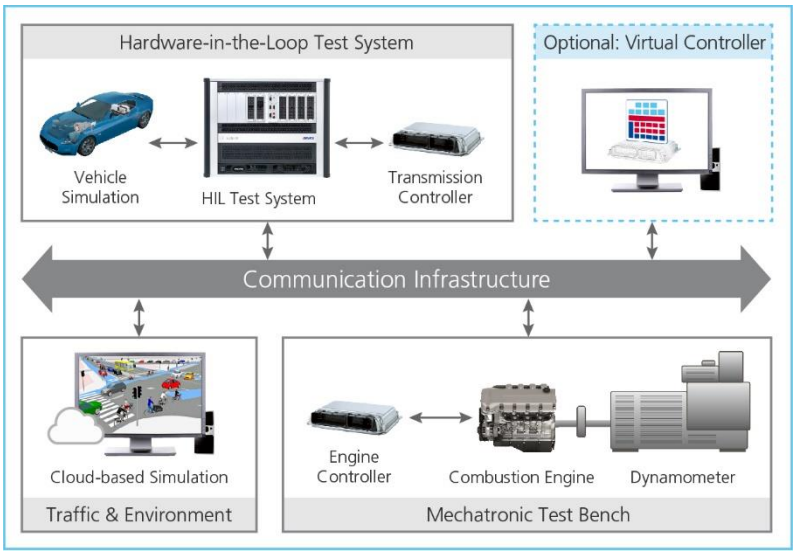

Fig. 2 Test bench setup for realistic laboratory-based emission tests.

To evaluate the impact of the driving functions on the fuel consumption and the emissions by means of measurements, especially in transient operation, the conventional part of the hybrid powertrain, the internal combustion engine, is integrated into the co-simulation on a test bench (Ref. Fig. 2). The electric part of the hybrid

powertrain, the electric motor, can either be integrated into the vehicle simulation of the HIL system or, to further increase the level of realism, it can be integrated into the co-simulation as a real component on a separate test bench. Controllers relevant to the driving strategy, e.g., adaptive cruise control, transmission control or hybrid control, can either be integrated into the co-simulation as virtual ECUs at early development stages or as real ECUs, connected to the HIL system via an electrical interface, in a later development phase.

All co-simulation participants, i.e., the engine test bench, the vehicle and e-motor simulation on the HIL system, and the network and traffic simulation in the cloud, are executed in real time. Due to the real mechatronic components on the test bench, the coupling between the HIL system and the test bench is subject to hard real-time conditions. The coupling between the network and traffic simulation in the cloud and the HIL system is only subject to soft real-time conditions. Since the network and traffic simulation in the cloud is executed at a slower task rate compared to the simulation executed on the HIL system, the missing intermediate simulation steps must be interpolated locally on the HIL system. In contrast to the rather soft real-time coupling of the HIL system to the cloud, it must be ensured that the coupling of the HIL system to the test bench is highly synchronous and time-deterministic with fast communication cycles.

The coupling described by this use case supports interactions between the different components of the co-simulation so that the impact of the simulated vehicle communication and the traffic on energy consumption and emissions can be measured as real physical values at the test bench. Complex driving functions can be tested in a laboratory environment while considering realistic traffic scenarios whose individual participants interact and communicate with each other.

To set up such complex heterogeneous test systems, special requirements for the individual subsystems as well as the necessary communication infrastructure must be considered. It must be possible to integrate subsystems, which potentially come from different time and application domains, efficiently and flexibly based on a communication infrastructure. This requires a modular system structure in which the overall system is ideally built up step by step from the individual subsystems (modules). In addition to the stipulated modular system design, it important to establish clearly defined interfaces between the subsystems. This is the only way to gradually replace individual subsystems during a development project in accordance with the project progress, from simulated components to real mechatronic components. For example, based on the use case described in this section, the mechatronic engine test bench will replace an engine simulation used in an early project phase. Ideally, the individual subsystems provide functional units whose interfaces are as independent of other subsystems as possible to avoid unstable system states and to relieve the communication infrastructure. The communication infrastructure itself must provide a standardized mechanism for exchanging simulation data between the different subsystems of the overall system. In addition to a standardized exchange of simulation data, it must also be ensured that the overall system can be centrally initialized, controlled, and monitored by means of functionalities provided by the communication infrastructure.

VI. Generic and Modular Common Test Bench Architecture

A generic and modular test architecture built on open, standardized interfaces and data exchange formats is per se beneficial to meet the specific demands for testing facilities of the aerospace industry. This is because it enables to solve the challenges the industry is facing today and to meet the strategic goals and high-level requirements for future test benches (Ref. section III).

This section describes a generic and modular test architecture along with trends and solutions to achieve open architectures. The approach is derived from the high-level requirements in section III and the use cases described in section IV. Technical and strategic aspects are also taken into account. The latter are related to topics that facilitate a cross-vendor approach.

This section also takes into account results from research projects in Germany and France that develop a modular and generic test architecture (Ref. [1]). Different test system suppliers, aircraft system suppliers, and an aircraft manufacturer participate in these projects.

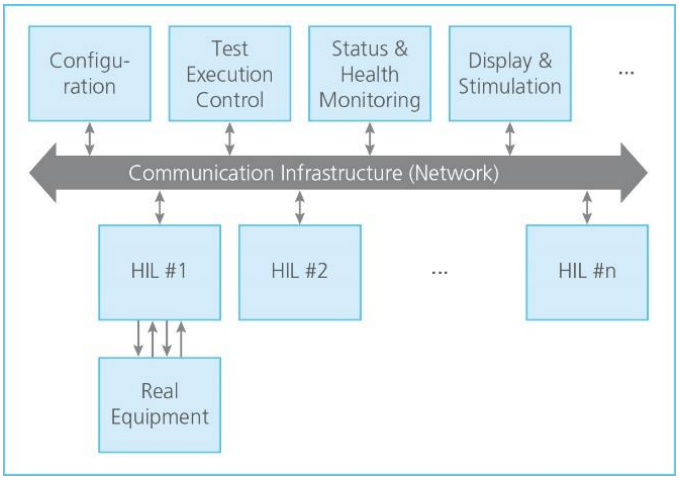

Fig. 3 Modular test bench with a standardized communication infrastructure.

A. Architecture

A generic and modular architecture for an open test environment makes it possible to overcome the limitations of traditional integration test systems of the aerospace industry. This type of architecture is based on open and standardized interfaces, letting developers use the best equipment for a particular task. The architecture of a modular test bench presented in this paper is shown in Figure 3. All functions to be covered by this test bench are clustered into modules that communicate through a standardized communication layer. There are essentially three types of modules: test execution, configuration and control, and test services.

Figure 3 displays modules required and used to configure and control a modular test system, as well as test services, above the communication layer. The list of services is not exhaustive.

The standardized communication infrastructure of the architecture in Figure 3 allows for communication between the different modules, which are the building blocks of a modular test bench, via standardized protocols and application programming interfaces (APIs).

The architecture must enable an economically viable adaption of different existing test technology from test system suppliers. This means that even upgrade paths of installed systems must be taken into account.

B. Standardized Communication Infrastructure

The heart of each open test system is a standardized communication infrastructure (Ref. [1], [6], [7]). It forms the basis for using and reusing laboratory test means of different aircraft programs and for combining modules of different types and from different suppliers. This not only improves modularity and scalability but allows developers to use the best equipment for a particular task.

There are essentially three types of functions to be fulfilled by the communication layer:

1) Exchange of simulation data while tests are being conducted on the test bench. This simulation data is predominantly real-time data. The exchange is synchronized with the simulation steps of the simulations executed on the test execution modules (HIL systems in Figure 3)
2) Exchange of commands and control data as well as status data. This data has to be exchanged during run time as well as before and after tests. It transmits command data for switching the states of the modules's state machines and the related respond data. It is also required for transmitting status and health information of modules. Command and control data is also required to enable the automated detection of modules that are connected to the communication layer. This data is not real-time-critical with regard to the conducted tests and real-time simulations. However, a command may have to transmit respond data within a specific amount of time.
3) The communication layer is also required to transmit file-based data. This is required to transmit test-related configuration files to the modules attached to the communication layer. The configuration data provides all required information about the operating conditions for each specific test setup (e.g., the configuration of I/O channels, simulation models). File-based data transmission may also be required to send log data to a central logging module after a test has been conducted.

The communication infrastructure has to be standardized at the protocol level. This way, suppliers can create standard-compliant modules without having to integrate third-party software. The key is to use an existing protocol

stack to ensure interoperability and add only one custom protocol layer (the data model). For example, the internet protocol suite is widely supported by commercial off-the-shelf network hardware and operating systems and can easily be extended. Thus, it is perfectly suited for the communication infrastructure.

The additional protocol layer then defines the semantics of the data that is exchanged between the modules of a modular test system and how it is organized. Therefore, its design depends on the specific use cases and corresponding requirements. For example, in [6][7] data models for distributing status information and control data are described.

C. Test Execution Modules

A test execution module hosts simulation models and I/O boards (real/virtual), provides the respective real-time scheduling mechanisms, and has standard communication, configuration, and control interfaces. A hardware-in-the-loop system with these interfaces is a typical text execution module. It provides the counterparts of physical interfaces of the system under test. A HIL system can be of any size. It can range from a single real-time processor with no I/O channels up to a multiprocessor system with several hundred I/O channels. A test execution module can also be a test bench with mechanical parts, e.g., to test actuators for flight control surfaces.

In Figure 3 different test equipment for test execution (i.e., hardware-in-the-loop simulators HIL #1, HIL #2, … HIL #n) is connected to the standardized communication infrastructure. The simulators can vary significantly in size and capabilities, and they can come from different suppliers. For example, one simulator may be dedicated to testing avionic systems. A second simulator may be used to test high-power electronics, and a third test system may be a test bench for an electric motor. It is also possible to have more than one simulator for an application (e.g., avionics).

Simulation models hosted on a test execution module are needed to simulate the environment of the system under test (e.g., to simulate aerodynamic loads for control surfaces). Simulation models can also be a kind of simulation of the real equipment. It this case, they are virtual equipment. Virtual equipment has the same functionality required for the intended test scenario as the real equipment. It is fundamentally a software model of the original device.

D. Modules for Configuration and Control

The modules for configuration and control have to enable the test bench users (i.e., the testers and test managers) to control the tests and monitor the test bench status, e.g., to load a configuration or to start and stop a test. In addition, the users need to know if everything works as expected and be able to take the appropriate actions should a problem arise.

To control a test bench, commands instructing the test bench modules about their next task must be sent to the modules. The commands are commonly implemented as remote procedure calls (RPC, cf. [8]). For example, every test system technology offers one or more commands with semantics for starting and stopping a test. In addition, commands can have parameters, such as a start time or a file name. The actual names or labels of the commands and their parameters as well as the correct sequence of their execution can vary, but generally the information is transmitted in a classic request-response pattern. Thus, a generic, modular and open test technology has to be compliant with this.

In addition, it should be possible to port existing APIs for testing to support the control mechanism. This is necessary to completely automate a test procedure. For example, the ASAM XIL API [9],[10], [11] is integrated in existing tools and test scripts. The standard is widely accepted and used in the automotive industry. It is also applied in the aerospace industry. XIL API support makes it possible to use established tools for testing without additional effort.

The configuration module has to allow for the automated generation of test bench configuration data. This requires standardized input data formats: aircraft configuration data, simulation model configuration data, module-related information, and information concerning the virtual or physical connection of the SUTs to the test bench.

The main requirement for status monitoring is to give testers access to the vital state of a module during all phases of a test. This means that the tester needs to know whether a module is available and operational or if an error occurred that has to be solved. If the health status of a module deteriorates, the tester needs more details about the issue. The tester must be able to access the log files of all modules from the operating terminal and might want a live view of log messages of the module that is currently under investigation.

E. Modules for Test Services

A generic and modular architecture for an open test environment requires more modules than those described above. Prominent examples are modules for test data monitoring and data stimulation to allow users to analyze and manipulate test data during a test run. Data recording and replay also requires appropriate modules. It must also be possible to configure and inject faults at different levels, e.g., logical transmission errors like bit flips or physical transmission disturbances like loose contacts, by means of a failure injection module.

Additionally, automated testing requires tools and modules to run automated tests (lights-out testing) on a generic and modular test bench (Ref. [1][4]). However, this is beyond the focus of this paper.

VII. A Unified Approach: Virtual Hybrid Testing Next Generation

The unified approach presented in this paper is the technology VHTNG – Virtual and Hybrid Testing Next Generation. It started a few years ago as a research project funded by the French and German governments. The project is led by Airbus and involves several aircraft test bench suppliers (Ref. [1]). The goal of VHTNG is an open standard as an answer to the new test equipment requirements.

Fig 3 shows the VHTNG architecture. The functions that are required to configure, operate, and control a VHTNG and to execute tests can be clustered into modules that are connected to the standardized communication infrastructure. These modules can be grouped into three categories: configuration and control, test execution, and test services.

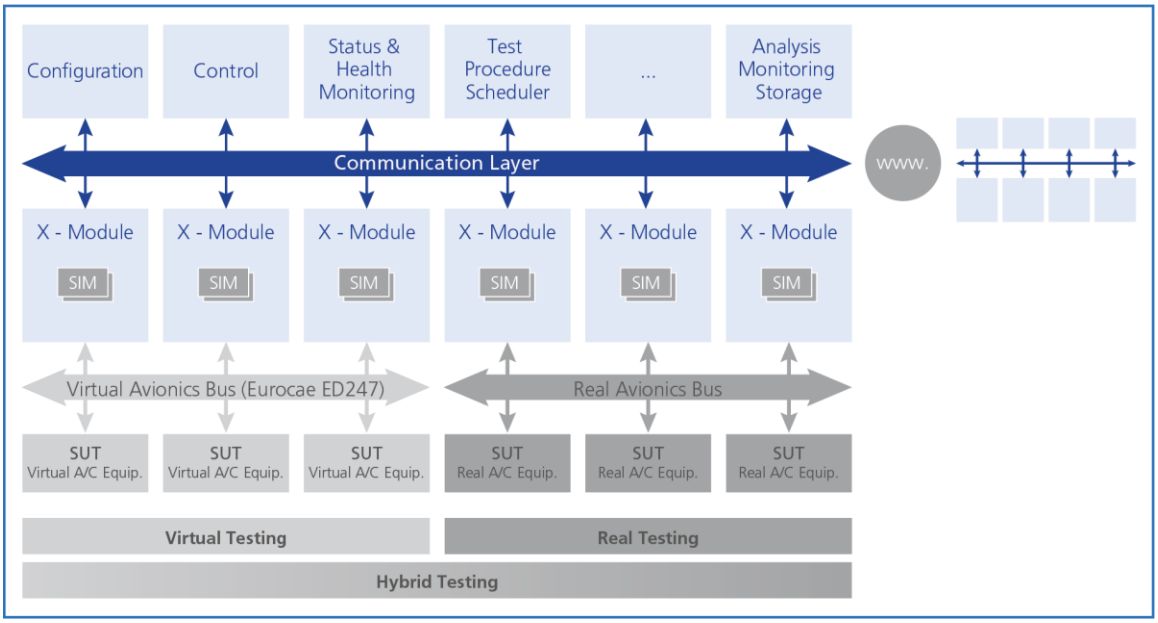

Fig. 4 Generic and modular test architecture VHTNG (Ref. [1]).

A. Standardized Communication Infrastructure

The VHTNG communication layer provides standardized communication between modules. It is enables real-time communication for variable exchange as well as a command and control bus. VHTNG uses a layered architecture, as is shown in Figure 4. This architecture enables standardization at the API level and the protocol level. The protocols, data models, and an application programming interface (API) are developed for VHTNG.

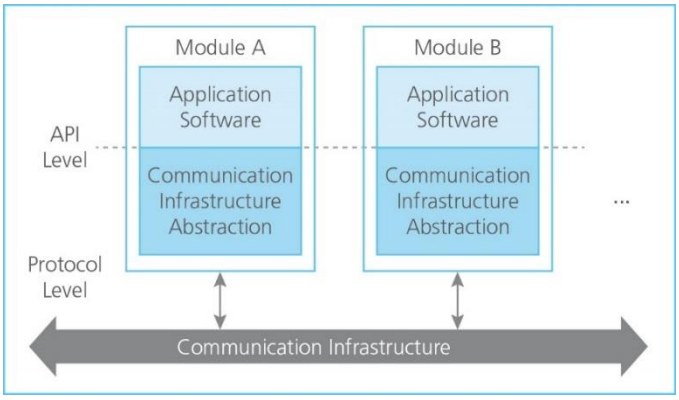

Fig. 5 Generalized layered architecture of a module.

The application software shown in Figure 5 consists mainly of simulation models running on the module (e.g., environment models, system models, or component models), I/O models (i.e., access to real I/O that interfaces with the system under test), and test automation scripts. This application software can be developed independently from the actual module by suppliers or by different company divisions (e.g., a centralized modeling organization). It emits and consumes user data that is technology-independent, such as simulated signal values or control commands. It is not aware of any other modules. In fact, another requirement is that application software must be allowed, if not required, to be agnostic of intermodule communication. Hence, the data must be processed by the module prior to transmitting it via the communication infrastructure. This is done by communication infrastructure abstraction, which decouples the application software from the communication infrastructure. From a technical point of view, it encapsulates the outgoing information of the application and extracts the information relevant to the application from the communication infrastructure.

The VHTNG standardization for variable data exchange and for the command and control bus is done at the protocol level (Ref. [6]). This way, suppliers can create modules that comply with the standard without having to integrate third-party software. The key is to use an existing protocol stack to ensure interoperability and add only one custom protocol layer (the data model.

VHTNG applies the internet protocol suite, as it is widely supported by commercial off-the-shelf network hardware and operating systems and can be easily extended. Thus, it is perfectly suited for the VHTNG communication infrastructure. Data Distribution Service (DDS) (Ref. [12]) is used as middleware (Ref. [6]).

B. Test Execution Modules

VHTNG test execution modules, named VHTNG X-modules, host simulation models and I/O boards (real/virtual), provide the respective real-time scheduling mechanisms, and they have standard communication, configuration, and control interfaces. Thus, X-modules are essentially test benches or hardware-in-the-loop systems. They can be of any size and capability.

The VHTNG concept enables the remote connection of test benches and the integration of legacy test benches, once they have the standard VHTNG interface.

C. Modules for Control and Status Monitoring

VHTNG uses modules for controlling a VHTNG system and monitoring its status and health state. Both rely on Ethernet communication (Ref. [7]). The role of the control module is to control the operating state of a VHTNG system, e.g., to load and unload configurations, to start, stop and reset a system. The aim of the status and health monitoring is to monitor the status of all modules of a VHTNG system and of all the entities within the modules.

The purpose of the VHTNG control module is straightforward. It sends modules of a test bench information about their next task. This information is implemented in the form of a remote procedure call (RPC, cf. [8]), which is a common approach. The control module uses a unified set of commands that every test system technology supports. This set of commands has been agreed by different test system suppliers. For example, every test system offers one or more commands with semantics for starting and stopping a test. In addition, commands can have parameters, such as a start time or a file name. The actual names or labels of the commands and their parameters as well as the correct sequence of their execution can vary, but generally the information is transmitted in a classic request-response pattern.

However, there is a limitation of the classic request-response mechanism that has to be overcome by VHTNG: Request-response mechanisms allow only one response per request, but multiple responses are beneficial if command execution takes a long time (e.g., several seconds or even minutes). Long execution times can become a problem for the requesting module because it does not have any feedback from the other module to know whether the other module is still properly processing the request or has not understood the command. The request might thus time-out without further notice to the requesting module.

With VHTNG, this problem is solved by the VHTNG status and health functionality. The response and command are used to inform the requested module that the command is understood and supported by the module. All subsequent status changes of the requested module are reported by using the status and health functionality (Ref. [7]). This is enabled by the fact that in many test environments, modules and simulations follow a certain state machine.

In VHTNG, the overall test bench, its modules, the module's I/O interfaces, and simulations are all entities that are monitored. Hence, they are called monitored entities. This makes it possible to deduce the status of the overall test bench hierarchically from the health statuses of its entities.

In humans, one good indicator for health is the heartbeat. The image of the heartbeat can be metaphorically transferred to computer systems in a network. In a VHTNG system, every module sends out periodic messages with all the information required to gain an overview of the current module status. If a module stops sending these messages,

i.e., if its heart stops beating, it is considered inactive. If the module is mandatory for a test, this is considered an error. There might be modules that are not mandatory for a test, for example, certain data monitors. These can start and stop sending messages at any time.

VHTNG modules can contain a hierarchy of monitored child entities that also have to be monitored. A solution based on simple sequences is applied for this. This approach works regardless of the underlying communication technology.

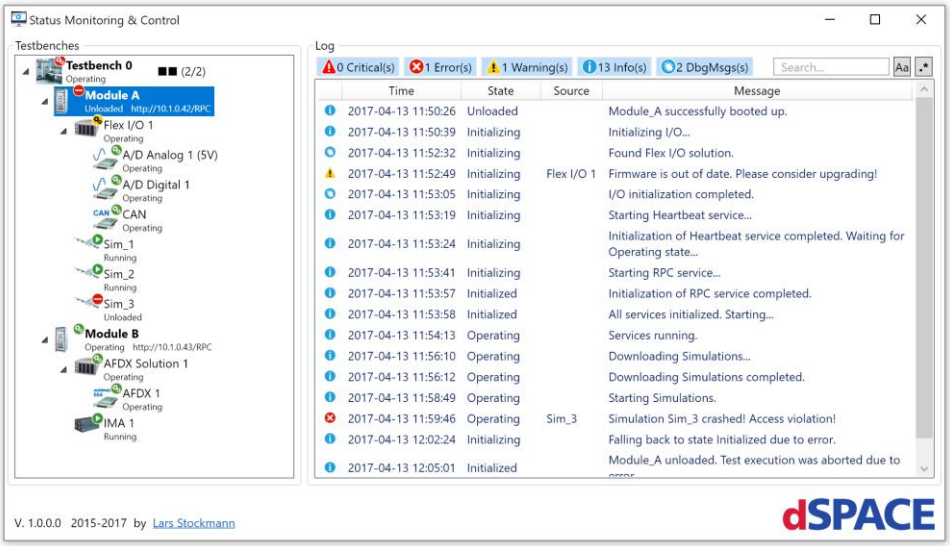

Fig. 6 Screenshot of a status monitoring and control tool (Ref. [7]).

Figure 6 shows the graphical user interface of a combined VHTNG status monitoring and control tool. It consists of two parts. The test bench hierarchy is shown on the left side of the user interface. In the example in Figure 6, the test bench consists of only two modules. The first module has a health status error because one of its simulations crashed. The crash was logged by the module. If the user selects the module, the status and control tool requests the log messages from the module and displays them in the right-hand pane. The source column confirms that the error originates from a simulation. The log entries can be filtered by severity or using a text filter. This way, it is easier for the user to find the relevant messages. Using the tool, the user can immediately see why the module's health status is in an error state and can take appropriate actions, such as unloading the module or reloading it.

The tool described has been successfully used in projects during the last years. In this time period, the underlying concepts have proven to be suitable. The tool has been used for diagnosing problems and controlling multiple tests in parallel.

D. Modules for Test Services

A test bench requires a number of test services, as has been briefly described in chapter VI. The modular setup of the VHTNG greatly eases the integration of test services. A test service can be hosted by a dedicated, configurable module connected to the communication layer. To give one example, a gateway module between the VHTNG communication layer and the VISTAS/ED-247 virtual I/O protocol (Ref. [13]) is considered here. In this specific case, the gateway functionality can be hosted by a dedicated gateway module or by an X-module. VISTAS is the acronym for Virtual Interoperable Simulation for Test of Avionics Systems. A first version of the standard was released in October, 2017. It standardizes the communication with virtual aircraft equipment.

The VISTAS communication has its own configuration files. For example, the ED-247 standard specifies the VISTAS network configuration file in detail. The VHTNG gateway between a VISTAS network and the VHTNG communication layer requires various configuration files, since it acts as both an X-module and a VISTAS node. Thus, it requires configuration files for both sides. At configuration time, the different configuration files are read and processed to generate run time data, which is then used by the gateway when it is set to the running state by the control module.

This can be seamlessly integrated into the VHTNG technology, as has been successfully demonstrated (Ref. [13]).

VIII. Conclusion

The strong need for a new test technology for integration testing in the aerospace industry has been identified. This motivated the development of the vendor-independent, generic, and modular test technology VHTNG – Virtual and Hybrid Testing Next Generation. It is the result of research projects that are led by Airbus and that involve several aircraft test bench suppliers. The goal of VHTNG is an open standard.

This paper motivates the development of VHTNG based on strategic goals, high-level requirements, and use cases. The latter result in specific requirements. It is relevant to note that the automotive industry faces similar problems related to integration testing as the aerospace industry, even though they work in different ways, to an extent. VHTNG can also meet the demands of the automotive industry, even though VHTNG has its origin in the aerospace industry.

Acknowledgments

This work was partly funded by the German Federal Ministry of Economic Affairs and Energy.

References

[1] Martinen, D., Lagalaye, M., Pfefferkorn, J., and Casteres, J., "Modular and Open Test Bench Architecture for Distributed Testing," SAE Technical Paper 2017-01-2117, 2017. doi: 10.4271/2017-01-2117.

[2] Delrieu, S., "Laboratory Test Means Scalable to the Test," SAE Technical Paper 2015-01-2546, 2015, doi: 10.4271/2015-01-2546.

[3] Wiels, V., Delmas, R., Doose, D., Garoche, P.L. et al., "Formal Verification of Critical Aerospace Software," *Aero. Lab Journal*, No. 4, 2012.

[4] Rasche, R., Himmler, A., Franke, M., Meyer, V., and Thoben, K.-D., "Interfacing & Interchanging – Reusing Real-Time Tests for Safety-Critical Systems," *2018 AIAA Modeling and Simulation Technologies Conference, AIAA SciTech Forum, (AIAA 2018-0123)*. doi: 10.2514/6.2018-0123

[5] Reinold, P., Meyer, N., Buse, D., Klingler, F. et al., „Verkehrssimulation im Hardware‑in‑the‑Loop‑Steuergerätetest", *VPC Simulation und Test 2018*, Hanau, 2018.

[6] Himmler, A., Stockmann, L., and Holler, D., "Communication Infrastructure for Hybrid Test Systems – Demands, Options, and Current Discussions," *SAE International Journal of Aerospace*, Vol. 9, No. 1, 2916, pp. 134–139. doi: 10.4271/2016-01-2051

[7] Stockmann, L. Himmler, A., "Monitoring and Control of Hybrid Test Systems," SAE Technical Paper 2017-01-2119, 2017. doi: 10.4271/2017-01-2119

[8] White, James E., "RFC 707: A High-Level Framework for Network-Based Resource Sharing," Proceedings of the 1976 National Computer Conference, 1976.

[9] Association for Standardisation of Automation and Measuring Systems (ASAM), "ASAM XIL," https://wiki.asam.net/display/STANDARDS/ASAM+XIL [retrieved May 2017].

[10] Brückner, C., Neumerkel, D., and Rasche, R., "Ready, Set, Go! Measuring, Mapping and Managing with XIL API 2.0," *Proceedings of the 7th ASAM US-Workshop*, Novi, MI, USA, 2014.

[11] Liu, P., Jürgens, J., "Collaborating in California: Dynamic Skip Fire Development Using HIL API," *Proceedings of the 7th ASAM US-Workshop*, Novi, MI, USA, 2014.

[12] Object Management Group, *Data Distribution Service for Real-time Systems Specification 1.4*, 2015.

[13] Hildenbrand, Y., "ED-247 (VISTAS) Gateway for Hybrid Test Systems", SAE Technical Paper 2018-01-1949, 2018.