

Communication Infrastructure for Hybrid Test Systems - Demands, Options, and Current Discussions

Andreas Himmler, Lars Stockmann, and Dominik Holler
dSPACE GmbH

ABSTRACT

The application of a communication infrastructure for hybrid test systems is currently a topic in the aerospace industry, as also in other industries. One main reason is flexibility. Future laboratory tests means (LTMs) need to be easier to exchange and reuse than they are today. They may originate from different suppliers and parts of them may need to fulfill special requirements and thus be based on dedicated technologies. The desired exchangeability needs to be achieved although suppliers employ different technologies with regard to specific needs. To achieve interoperability, a standardized transport mechanism between test systems is required. Designing such a mechanism poses a challenge as there are several different types of data that have to be exchanged. Simulation data is a prominent example. It has to be handled differently than control data, for example. No one technique or technology fits perfectly for all types of data. There are certain requirements that have to be fulfilled. For example, the mechanism for data exchange needs to have adequate performance to satisfy the demands of the industry. Another requirement is that it must rely on well-established standards to ensure stability and be future-proof. This paper describes the architecture of hybrid test systems, with special emphasis on the communication infrastructure. On the basis of this, it then analyzes the requirements for realizing a suitable communication infrastructure of hybrid test systems. Finally, it presents a proposal of how to devise a standardized interface to this communication infrastructure.

CITATION: Himmler, A., Stockmann, L., and Holler, D., "Communication Infrastructure for Hybrid Test Systems - Demands, Options, and Current Discussions," *SAE Int. J. Aerosp.* 9(1):2016, doi:10.4271/2016-01-2051.

INTRODUCTION

Traditionally, test setups for the aeronautics industry are engineered from ground up. This means that it is common to design every required laboratory test mean (LTM) for a specific aircraft program with a particular setup in mind. This holds specifically for integration test systems. This approach hinders the sharing of test resources between different tasks of the same program, even more 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 needed for a given test task) and connecting different test systems to a larger test system.

Nowadays, even a simple sharing of test resources is difficult, due to the required re-engineering and maintenance costs [1]. The reason for this is that the communication between the involved systems is realized using specific protocols and software interfaces. It is not possible to simply exchange one LTM component or reuse a test that has not been explicitly designed for this setup. A costly adoption is most certainly required.

Thus, there is a need to make integration test systems for the aeronautics industry more modular, scalable and flexible than they are today. This demand is also essential due to the high level of system integration that is characteristic of new aircraft designs. This

dramatically increases the complexity of verification, just like it also increases the complexity of the design. And the demand becomes even more urgent with every new aircraft program, as the system complexity and the level of software complexity rises with each new program. A common measure to visualize this is the number of lines of code required for electronic controls on board an aircraft (ref. Figure 1, [2]).

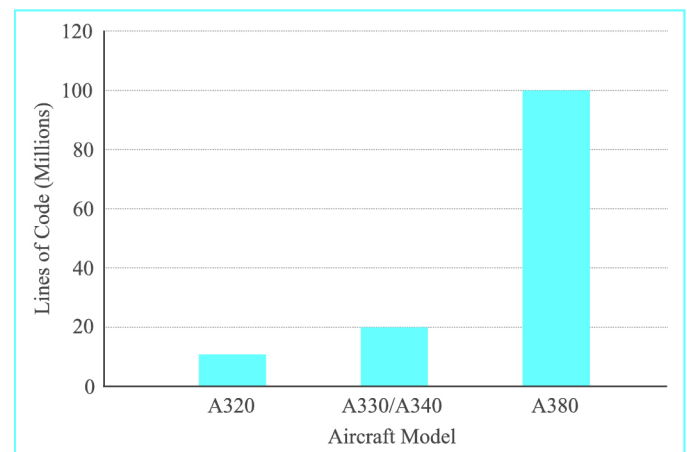


Figure 1. Increase of software complexity on aircraft [2].

The demanded modularization or partitioning of integration test systems (also referred to hereinafter as hardware-in-the-loop (HIL) systems) requires (1) hardware and software interfaces between HIL subsystems (optionally spatially distributed, with or without real-time coupling), (2) software interfaces between HIL systems and PCs to control the HIL systems, and (3) model interfaces. All of these interfaces need to be standardized in order to achieve the above-mentioned goals.

Parts of these required interfaces are already under discussion and even in use and supported by modern test technologies [3, 4], like software interfaces to control HIL systems, model interfaces, and real-time coupling of test systems over large distances. This paper focusses on the communication infrastructure required for modularized test systems, taking into account the architecture of this infrastructure.

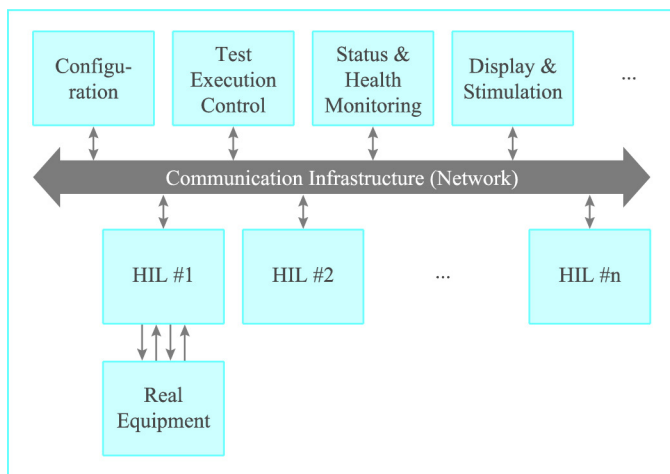


Figure 2. Network-oriented LTM architecture [1].

A modularized test system as named above will have a network-centric architecture, as shown in Figure 2. Depicted are different exemplary LTMs that are connected to the communication infrastructure. These LTMs are referred to hereinafter as modules. The bold double-headed arrows symbolize network communication that has to be realized on a module. It can be assumed that real equipment (e.g., the system under test, such as an electronic control unit) cannot directly be connected to the communication infrastructure as it cannot be expected to offer the required network interface. The smaller arrows symbolize generic I/O (e.g., analog, discrete, bus communication, etc.).

This paper focusses on the communication infrastructure of a network-oriented integration test system. First, the requirements for the communication infrastructure are analyzed in the section “Requirements Analysis”, which discusses a layered architecture together with suitable interface layers. After that, the section “Technologies for Data-Centric Communication”, identifies reasonable technologies for implementing the communication infrastructure.

REQUIREMENTS ANALYSIS

The most obvious requirement of the communication infrastructure is that it should be possible for each single module to use the infrastructure to communicate with other modules. As stated earlier, each test system currently uses technology-dependent ways of communication. To achieve interoperability, some kind of standardized interface is needed.

There are actually two kinds of communication in a test bench. One can be classified as being data-centric and the other as being control-centric. Each class has different properties and thus its own requirements regarding the communication mechanism. For example, data-oriented communication, like simulation data, is generated continuously and may be interesting for many participants in a test setup. Control-oriented communication on the other hand is directional. A communication infrastructure must be capable of communicating both kinds of information. Hence, the final communication infrastructure will at least have two standardized interfaces.

The second requirement, which may seem also fairly obvious, is that the interface has to be defined in a way that makes it possible for every module provider to implement this interface. This means that the interface needs to be compatible with different test system technologies and relatively easy to adopt with regard to its functionalities. This includes that it should be realized on top of a potentially existing implementation.

Layered Architecture

The more complex the test benches get, the more important it is to have a suitable architecture that allows a separation of concerns. This enables distributed development and simplifies the integration process. It needs to be possible to test the software and hardware modules for hybrid test systems independently and reuse them.

A hybrid test bench as under consideration in this paper requires at least three layers, one of which is the communication infrastructure. It constitutes the base of the communication. It is interfaced and accessed by a communication infrastructure abstraction layer, which interfaces the communication infrastructure on protocol level. On top of this infrastructure abstraction layer is the application software, which interfaces the abstraction layer on API level.

The application software 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 (ref. Figure 2). As already stated above, this application software may be developed independently from the actual module by suppliers or by different company divisions (e.g., a centralized modelling 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, it is yet another requirement that application software should be allowed, if not required, to be agnostic of inter-module communication. Hence, its data must be processed by the module prior to transmitting it through the communication