applied sciences

MDPI

Article

Industry 4.0 Technologies as an Obsolescence Mitigator for Testing of Mechatronic Systems in Aviation

Konstantin Klein [1,*] and Klaus-Dieter Thoben [2]

[1] BIBA—Bremer Institut für Produktion und Logistik GmbH, 28359 Bremen, Germany
[2] Department of Production Engineering, Universität Bremen, 28359 Bremen, Germany
* Correspondence: kle@biba.uni-bremen.de

Abstract: Companies building products with a usage phase of 25 years and more, have suffered from obsolescence for decades. Software and hardware components inside of the supporting systems, such as avionics test benches, are particularly affected. They consist of components built for 3–7 years but expected to operate for decades. The lack of standardized, open and modular systems for special timing constraints increase the pressure and impede the interoperability in this field. Industry 4.0 systems offer exactly this, but do not support aviation standards. The research in this article aims to show that it is possible to use widespread Industry 4.0 standards and technology at the field level to achieve transferability and maintainability in the operation of test benches. This article describes an implemented and evaluated proof of concept of the implementation of an industrial ethernet standard EtherCAT for the ARINC 429 I/O for an aviation test bench. As a baseline, market available EtherCAT components were evaluated and selected. For the client development, standard components for the EtherCAT logic and ARINC 429 were used. Additionally, the full interoperability at the syntax level, was achieved by implementing a configuration approach of the I/O, in order to encode and decode the messages. The evaluation was performed using an Airbus A350-1000 test bench for secondary flight controls—namely the HighLift system.

Keywords: industry 4.0; obsolescence; design support systems; automation; testing

check for updates

Citation: Klein, K.; Thoben, K.-D. Industry 4.0 Technologies as an Obsolescence Mitigator for Testing of Mechatronic Systems in Aviation. *Appl. Sci.* **2022**, *12*, 11142. https://doi.org/10.3390/app122111142

Academic Editors: Tobias Larsson, Shaun West and Christopher Ganz

Received: 14 September 2022
Accepted: 31 October 2022
Published: 3 November 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Companies involved in the design, manufacturing and marketing of complex mechatronic systems in aviation are subject to different constraints in their implementation. One of the constraints is the industry-standard requirement that products—such as airplanes—have a designated life time of 25 years or more after refurbishment [1] (p. 39). Products with this planned life span are referred to as sustainment dominated products [2]. The current regulations for certification stipulate that these products must remain maintainable and recertifiable [3] (p. 105).

This constraint leads to consequences that are typical for modern mechatronic systems in aviation. Technical approaches that address the conditions are primarily implemented in information technology. Specifically, software development is considered the biggest driver in the development [2]. For them, correctness (of all developed components of a product) must be proven. The driving force in the correct development is not only the quality requirement of the manufacturer, but also the necessary government certification, as clearance for use in the legal space. For this purpose, test benches are used, which help the test engineer to execute requirement-based tests, to access the system under test (airplane systems) and to understand the results, as well as to store them for the entire life cycle.

In the integration test phase, as part of the certification, the subsystem of the mechatronic system and the test bench form a temporary system. All test operations are only possible in this constellation. For this purpose, the test benches must support the technologies used in the mechatronic system over the entire life cycle. Continuous product changes

and interim tests require the same lifetime from the test benches as from the actual product. It remains even when the product (airplane) is already past the certification phase (design phase) and is in the operation (usage phase) [3] (p. 105). Test benches are implemented from commercial off the shelf (COTS) components. They are designed for aerospace test benches only. The used components are produced in small series. They have an expected lifetime of three to seven years [2]. As a consequence, individual components in the test benches have to be replaced several times during their operational phase, in order to maintain the operational phase of the mechatronic system. These components are software, hardware and mixed forms of software and hardware. The affected components are no longer available on the market with the advanced service life. In order to mitigate the consequences of obsolescence, current approaches rely on hardware abstraction layers (HALs) [4]. This approach reduces the effort of the reimplementation in such cases. This article shows a proof of concept for a different approach, relying on the standardized I/O modules from the field of automation in production. An additional configuration approach will show the configurability of the I/O. This will enable the full interoperability and exchangeability of the I/O, not only at the system, but also at the syntax level. This approach enables the interconnection of test benches from different enterprises using different coding standards for variables inside the test benches, following the Industry 4.0 paradigm.

1.1. The Structure of the Work

The main part of this article starts with the description of the problem statement around the functional obsolescence. It describes the life cycle of a test bench with particular focus on the revision and explains the replacement of an obsolete component, namely the I/O cards. Within this process, the article explains the positions in hardware and software responsible for the obsolescence. Section 1.4 explains the objectives of the work. Additionally, the rationale of the chosen approach and its novelty will be stated out. The influence of technologies around the Industry 4.0 paradigm will be in focus.

Materials and methods starts with the explanation of the HighLift testing scenario. The HighLift system is a part of the flight control system—a reactive, safety critical real-time system [5]. These properties directly and indirectly influence the obsolescence of the software components of the HiL test benches. These relationships are explained in Section 2.1. From the perspective of the test bench, the I/O components are considered in particular. They consist of a software component for the data exchange and a hardware component that sends and receives data as an electrical signal. For the I/O component, the ARINC 429 interface was chosen. ARINC 429 is the most widely used data transfer medium in aeronautics [6] (pp. 4–31). The proposed concept of the standardized and syntactically configurable I/O is explained in complete, in the third section.

The implementation of the proof of concept and its validation in three scenarios are the core of the results in the presented work. The paper concludes with a discussion of the results and the outlook.

1.2. Problem Statement in the Context of Obsolescence

To enable a reliable test process, the HighLift components are manufactured as a single component or as a small series and tested as a black box at the system level. In the next step, those components are to be integrated and tested with the focus on the software within the slat flap control computer (SFCC). The test process of a SFCC emphasizes the application of two different test means. The first is the hardware in the loop test bench in which the real SFCC is tested within a simulated environment. The second test mean is a real HighLift system rig, depicted in Figure 1, which performs the slap/flap drive sequences and failure cases using real system components [7].



Figure 1. Airbus A350 Test Rig-Image from Airbus Operations GmbH.

Both test benches (see example in Figure 2) use I/O cards to access the system under test (SuT). Both test means represent large industrial equipment developed for a specific variant of a product. It makes it the **one-of-the-kind** test mean. It has an operation time for the complete product lifecycle of a specific aircraft variant. When the test campaign is completed, the test means are already ca. 5–7 years old. Within this time span, some of the components may already have become obsolete. To be able to guarantee availability over the complete lifespan, the periodical maintenance actions, such as the periodical refurbishment are mandatory [8].



Figure 2. A380 Test Bench-Image from Airbus Operations GmbH.

The usage scenario of a test bench is divided in three phases: pre-certification phase, post-certification phase and in-service phase. During the pre-certification phase, the test rig is used daily, and the availability is critical. Once the aircraft variant is certified and the production has already been started, the post-certification phase starts, and the availability is important. In this phase, the design of an aircraft is optimized and tested, regarding minor issues detected during the certification. During this phase, the test rig is used continuously. Once the post-certification phase is finished, the in-service phase of the airplane starts. During this phase, the test rig is only used if it is necessary. Common requests are when the aircraft has serious problems during its operation or in the occurrence of emergencies. During this phase, the test rig has ended its useful life, but its lifespan hasn't ended, and it has to be ready in cases of testing demands. If the company decides to no longer maintain the equipment, the FAA (Federal Aviation Administration) and EASA (European Union Aviation Safety Agency) can decommission the airplane type completely [3] (p. 105).

Obsolescence of an I/O card occurs normally within the mentioned periodic maintenance and refurbishment process performed by the supplying companies. The following two situations are responsible for most obsolescence problems in the maintenance of test benches:

1. The card can be found to be defective—the chance of the defect of an I/O card within first four years of the operation is at 50% [4]. Inevitably, the operating company will face a situation where the defective I/O card is no longer maintained, and a simple replacement is no longer possible. Since the I/O card consists of software and hardware (driver and PCI or VME card), the change of hardware requires the change of software and the reimplementation of all of the accessing proprietary components.
2. Following a necessary upgrade of the operating system of the test bench, the new version no longer supports the I/O card. In this case, the I/O card will need a replacement to a variant that supports the new version of the operating system with all the consequences of the first case.

The above-mentioned effects are amplified through the fluctuation in engineering teams. Once the test bench is designed and in operation, engineers change departments or companies. The knowledge of the test bench design vanishes. This applies to supplying companies as well. Those can quit the market completely [1] (p. 100).

In the literature, those cases are described as software obsolescence, even when the effect is caused by hardware [9] (p. 11). The interface where the obsolescence occurs is the deciding point, according to [10] (p. 25). More precisely, it is functional obsolescence, what occurs when hardware or software changes [11]. This leads inevitably to an interoperability problem at the system and syntax levels [12].

1.3. Design Methods against Obsolescence

This section shows the characteristics of the available design methods to mitigate the risks of obsolescence. These characteristics are introduced briefly before the known implementations are described.

1.3.1. Openness of a System

Articles unanimously cite system openness or technology transparency as the most important factor in minimizing the risk for software obsolescence [10] (p. 29). Open systems are, according to [10] (p. 29), systems with interfaces, based on generally accepted standards. It is assumed that this hardware and software design methodology ensures that a module can be replaced if necessary. The technology including interfaces used inside the component are not relevant. Access to the source code is to be preferred [10] (p. 32).

1.3.2. Modularity

Modularity is a concept that breaks down a system into pieces of components with standardized interfaces. Thus, modular systems are open systems. That allows the exchange of components inside the system. Modularized system architectures expose only the interface to the outer world and hide their internal implementation. The coupling of the software to the hardware is loose [13] (p. 11).

1.3.3. HAL

Modern test benches use the approach of the hardware abstraction layer (HAL) to mitigate obsolescence. HAL is a software layer what presents a common interface to test bench hardware. The applications access the test bench drivers only through HAL. The actual driver is hidden. When an I/O card becomes obsolete, only the HAL requires a programmatical adaptation, instead of all I/O accessing applications. That minimizes the costs drastically, but the core problem remains unsolved [4]. A proprietary HAL brings additional obsolescence risks when its software interface becomes updated.

Other approaches to mitigate the I/O obsolescence, include HAL relying on open standards ARINC 653 [6] (36-1) and AUTOSAR specification of the I/O hardware abstraction [14]. They are technically open HAL. So far, they have only been implemented and specified for the control computers themselves. They are insufficient for the implementation of interfaces inside of the test benches. ASAM XIL [15] addresses test bench

interfaces directly and is, at present, the only known HAL specification that is open. The above-mentioned problems remain.

1.4. Objectives Development—Industry 4.0 as a Problem Solver in Obsolescence

As described in the previous section, the major mitigation strategies for functional software obsolescence are open, standardized, configurable and modular systems. Additional requirements (described in the following Section 2.1) concern especially the real-time capability of the avionic system and thus, the test bench and its I/O.

The Industry 4.0 paradigm addresses these features. The term was initially developed as part of the public private partnership of the German government. It describes the fusion of enabling information and communication technologies described, such as the Internet of Things (IoT) or cyber-physical systems (CPSs). They form a smart factory [16] (pp. 225–226) through digitalization, automation and interconnection of the relevant stakeholders around producing companies [17] (p. 7).

In the field of the functional testing in aviation, several concepts around the Industry 4.0 paradigm were implemented. An interconnection of the test benches with different purposes were shown by Martinen [18], as a configurable machine-to-machine communication. The authors see the usage of the open group standard data distribution service (DDS) [19] as a solution for the obsolescence. A different suitable standard for this purpose is OPC-UA. Franke applied the SCXML standard for the creation of consistency, in exchange and execution of real time test cases, throughout the supply chain [20]. Both approaches were already applied and evaluated using the HighLift test benches.

For the connection of sensors, actuators and control units at the field level configurable communication technologies, such as Profinet, Sercos III and EtherCAT, were standardized. In the classical Industry 4.0 hierarchy, those standards were developed for typical automation tasks. Today, only these standards are suitable for the control of electric motors [16] (pp. 147–154). The modular concept of these technologies, along with its open standards, satisfies the requirements to minimize the risk of obsolescence. Additionally, the currently available industrial assets in the smart factory environment at the field level are considered to be mature. The standardized I/O and programmable logic controller have an expected lifetime of 20 years [21].

In summary, the objective of the research described in this article is to show a proof of concept for the applicability of industrial standards from the aforementioned domain, at the field level, as an interface technology for test benches in avionics.

2. Materials and Methods

Within this section, the important features of the HiL test benches in avionics, as well as avionics itself, are explained. These features define the requirements for the I/O. Using these features, the concept of the obsolescence safe I/O will be explained in detail.

2.1. Relevant Featurs of Avionics and Test Benches

The avionics system represents the system under test in the HiL approach, also called the SUT -system under test. Avionics is a word made up from aviatics and electronics, i.e., flying electrics and electronics. This term was established when electronics were first integrated into flight systems in the 1960s. From this perspective, control devices responsible for control tasks and are to be certified, are avionics [22] (p. 8).

For the present work, knowledge of properties and underlying requirements of avionics systems are relevant because they determine the properties and requirements of test benches. For example, the degree of complexity of the avionics system is thus transferred to the complexity of the test bench. Furthermore, for example, requirements on the real-time capability of the avionics system also sets these requirements on the test bench. The aspects of modern avionics relevant to testing are described below.

2.1.1. Reactive Real-Time Systems

Reactive real-time systems record aspects of the real world, through sensors, using a measurement system, and process them within a computing system [23]. Subsequently, they trigger actions by an actuator, which influences the real world. Depending on the scenario, a time limit for the calculation per cycle is defined, which must be adhered to the system. In a hard real-time system, the defined cycle time is not exceeded in any case. The explained model of a real-time system is shown in the Figure 3 [24].

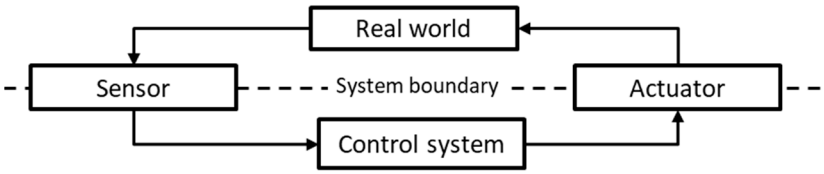


Figure 3. Model of a real-time system.

Hard real-time requirements demand a deterministic behavior of the system. This means that the computing system always delivers the same output signals for the same input signals.

The test benches must comply with the specified timing conditions and must not influence the timing behavior. Test benches with a real-time behavior are more complex to develop and maintain. The I/O components of the test benches must therefore meet the same requirements. The real-time behavior cannot be implemented on any kind of hardware. The software, the operating system and protocols were specially developed for this purpose. Depending on the degree of sharpness of the real-time behavior, only certain programming languages can be used for the system implementation. Real-time systems thus impose on severe limitations in the implementation of the solution to the obsolescence problems.

2.1.2. Components of an Avionics System

All avionics systems have one thing in common. They all implement a power supply, a control unit, input and output interfaces, and access to the central on-board data network [1] (p. 260). These components are relevant for functional testing.

Figure 4 shows the generic architecture of an avionics computer as an example. The components shown are available in the SFCC. The central processor unit and the bus interface to the ARINC 429 network are relevant for the present work. The other components are shown for the sake of completeness.

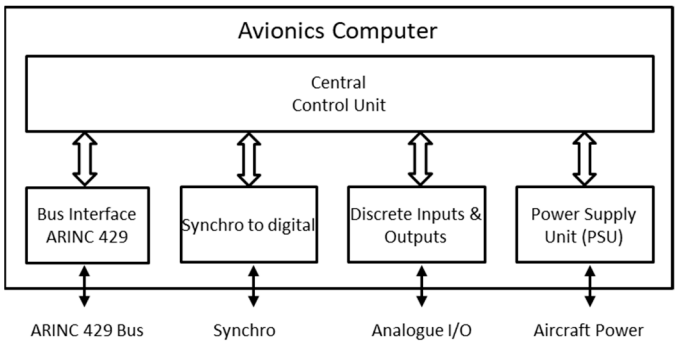


Figure 4. Rough architecture of a control computer.

Control unit: The control unit executes the programmed instructions in the airplane. In its additional memory, the signal values are stored for the subsequent processing and generation of output values for the actuators. For this, the programmed instructions are implemented in software. Thus, the software is the basis for the (flight) control system. It implements the mathematical functions, algorithms, tasks for data transformation or

calculation. The software represents the core of the device under test (DuT) for functional testing and for the certification. The goal of the verification is to prove the correctness of the software running here.

Connection to the on-board data network: The aircraft's internal on-board data network transmits data for the exchange between the individual systems. This includes data from velocity sensors, attitude sensors and others. This data is generated by the air data inertial reference system (ADIRS) and exchanged via the ARINC 429 or ARINC 664 bus.

During the execution of a test, the components outside the boundary of the system under test, such as the ADIRS, are simulated. The simulation generates the data and puts it on the bus system via the I/O interfaces to supply the DuT with the data. The data sent by the DuT to the on-board network is recorded and continuously checked for correctness. Thus, the on-board network represents one access point for the ARINC 429 I/O of the test bench.

2.1.3. Product-Side I/O Example for the ARINC 429

With the use of digital systems in the 1960s and 1970s, it became necessary to exchange data between them. From the beginning, the numerical data in the network protocols—analogous to the processing systems—were binary coded, in order to transport them from system to system. An example of this is the ARINC 429, which is an inexpensive communication protocol to implement. It has resulted in being, by far the most widely used communication protocol for airborne equipment in civilian aviation. It was developed in the late 1970s and was first used in the early 1980s [25] (pp. 88–89).

With the ARINC 429, an 8-bit label number is assigned per message. This label number is the identifier for a 32-bit message. The label number is used to determine how the message is to be decoded. Each message has a parity bit at the end, which is used to verify the correctness of the message. The bits in between—9 to 31—can be used for the user data [6] (p. 34-4) (see Figure 5), even if bits 9 and 10 are reserved for source/destination identifiers (SDE) and 30 and 31 are reserved for state identification (signed/status matrix—SSM). The ARINC 429 operates in 12,5 or 100 Kb/S.

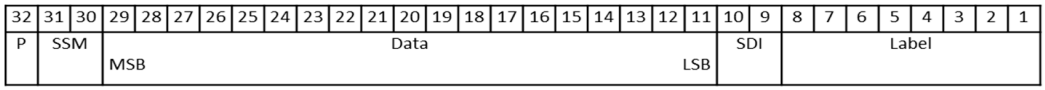


Figure 5. Format of an ARINC 429 message–derived from [6] (p. 34-4).

Due to the high prevalence of the ARINC 429 in avionics, the interface is supported by effectively every test bench in aviation. However, this protocol is of particular importance for the present work, due to its wide distribution within only one domain, as it is susceptible to obsolescence.

2.1.4. The Test Bench

The test bench for the system and/or integration testing is a complex system that is coupled to the avionics system under test via, the I/O components. It contains various components that are used for the test automation, test execution, simulation and operation. The HighLift test bench, as depicted in Figure 2 is a cluster consisting of many computers installed in several rigs.

2.1.5. Description of the Functional Test at Airbus in the HighLift Use Case

The SFCC is embedded in the test bench as the device-under-test (DuT). Its interfaces are directly connected to the test bench. With the automated test execution, the DuT is initialized and brought into the operational state for the test execution. Depending on the test type, the signals within the SuT are either monitored or monitored and manipulated. These types will be illustrated with an example. In this simplified setup, a sensor transmits the temperature of the hydraulic oil to the control computer via the ARINC 429. The purpose is to test if the control computer performs correctly during the operation.

Additionally, it is tested to see how the control computer handles the sensor values if they are not plausible. The temperature values are additionally manipulated. An abstracted structure of the real-time system is shown in Figure 6.

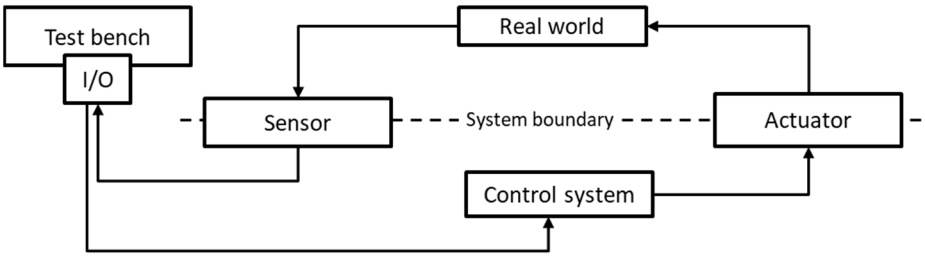


Figure 6. Real-time circuit extended by the test bench: schematic representation.

The example outlined above describes the receiving of an ARINC 429 signal, verification, manipulation and forwarding. The purely physical signals must be transformed into interpretable information for this purpose. To illustrate the transformation, Figure 7 shows a schematic layer representation of a test bench. It includes an indication of the transformation steps of the physical signals into interpretable information and vice versa. The signal processing is described in the following:

- Signal acquisition begins with the reception of a complete message. The message is interpreted in the I/O card and passed through the PCIe bus to the main processor and the operating system.
- The driver for the PCIe I/O card is integrated within the operating system. The driver provides an interface that is addressed by the application software.
- The application layer transforms the data into a format that the applications can interpret.
- This means that a 32-bit ARINC 429 message becomes information after decoding, according to the configuration (example: temperature of hydraulic oil: 50°).

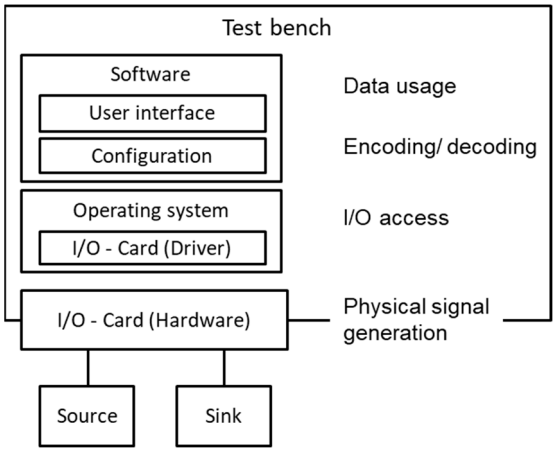


Figure 7. Test bench in the layer representation with data/signal transformation, according to VDI 2882.

In addition, a set of meta information is transmitted to each value from the I/O card. This includes: the message number, the repetition frequency of the message, the correctness of the message (parity bit check).

When the subsystem has achieved the necessary certification at the system level, it leaves the responsible department and is integrated with all the subsystems to a complete system—the iron-bird. This is now tested until the certification bodies approve the aircraft for flight testing. The process described is applied in exactly the same way to the design changes [8].

2.2. The Concept of Open, Syntactically Describable Interfaces for the HiL Test Benches

The concept of open, syntactically describable interfaces for the HiL test benches addresses the problem of functional software obsolescence of the I/O, in the described three levels (see Section 3.2). These levels describe the occurring obsolescence from software to software, from software to hardware and from hardware to hardware. The concept is first presented for the respective levels and then combined into an overall concept.

Limitations:

1. in the I/O or standards, the software part (driver) is related to the hardware part (I/O card). This means that the I/O standards that contribute to solving the hardware-to-hardware obsolescence must also contribute to solving the software-to-hardware obsolescence and vice versa. For this reason, the hardware-to-hardware obsolescence is only implicitly considered.
2. Care is taken to ensure that the hardware part conforms to a standard that is no less common than PCIe. Technologies, based on fieldbuses are more prone to obsolescence than PCIe and Ethernet. For this reason, Ethernet can be used as a standard.
3. The avionics system operates under hard real-time conditions. This means that the test bench and its I/O must also operate under those conditions. This point limits the choice of I/O technologies to only a few candidates.

2.3. Interoperability at the System Level

The concept of creating interoperability at the system level consists in replacing the proprietary parts of the I/O with open standards and their open-source implementations. Despite the application of different strategies for the new network standards 5G and WiFi 7, such as the non contiguous channel bonding, multi-link operation or the cooperative AP, radio based networks cannot guarantee a low latency or deterministic timing behavior [26].

For wired networks, Ethernet is by far the most widespread and therefore obsolescence-proof standard. In this respect, Ethernet has no competitors [27] (p. 308).

Suitable wired standards come from the IoT and automation sectors. Authors in [28,29] provide methods for the technology selection for the two sectors. Real-time capability is not mandatory for use in most cases in the HiL test benches in avionics. Examples are the domains of the cabin or cockpit. The situation is different for flight controls. Standards for communication under real-time conditions, according to (category 3, according to IEC 61784-2), using Ethernet can only be found in automation technology.

These requirements are met by the PROFINET IRT, SERCOS III and EtherCAT standards. [16] (p. 145) EtherCAT is the only standard that achieves hard real-time conditions in any topology. In addition, SERCOS III and PROFINET IRT can only operate with additional proprietary hardware, which brings additional obsolescence problems. EtherCAT is a proposed standard for the HighLift scenario. Different systems under test may allow different standards with no timing restrictions. Here, common standards, such as MQTT, are sufficient. In this article EtherCAT is an example of suitable candidates.

2.4. Interoperability at the Syntax Level

Interoperability at the syntax level describes the encoding, decoding and representation of the data. The encoding of the data in the HiL test benches is determined by the SuT. In the avionics integration test, it is the interfaces of the control computers. The syntax of data exchange in avionics is stored in Interface Control Documents (ICDs). The ICD format for the development of the A340 and A350 aircraft variants is table-based. The syntax for the interface description is organized by the avionics equipment. The individual data are recorded in comma-separated format (CSV) [30].

Standards for the ICD in aviation do not exist. The ARINC 653 [31] contains a proposal for the standardization of configuration. However, it does not include the configuration of the interfaces. ASAM XIL API describes the interfaces of the control computers. Its syntax is not applicable to avionics hardware [15]. Two authors published aviation ICD standardization efforts [32,33]. Both proposals are XML-based and call their result Meta-

ICD. They allow the encoding of signals for all common communication protocols in avionics. Both publications do not mention the progress of the standardization efforts.

3. Results

3.1. Implementation of the Open and Standardized I/O

The use of EtherCAT prescribes the change of the architecture of the I/O to a master-slave architecture. This changes the architecture of the test bench from Section 2.1 to the one shown in Figure 8. The individual components are described below.

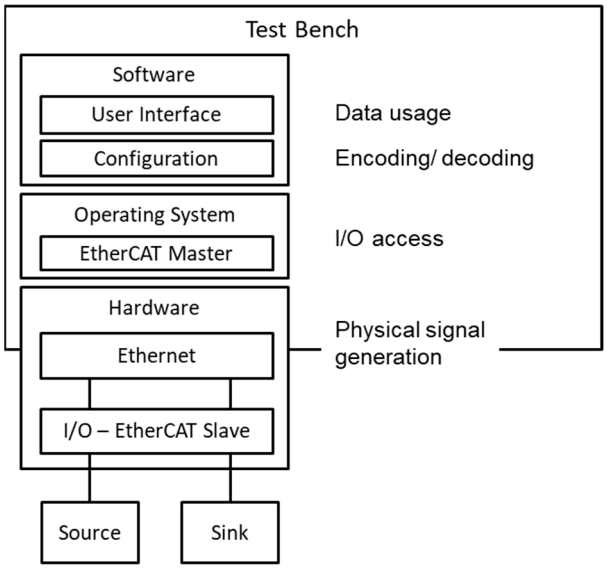


Figure 8. Test bench architecture after the implementation of EtherCAT for the I/O.

EtherCAT Master—replaces the driver in its function as the open-source driver. The master configures the network and sends the messages to the slave. It receives the messages from the slave again and communicates them back to the upper layers of the software [34]. The EtherCAT Master is the exposed software interface to the test bench on the system level.

EtherCAT Slave—Receives the messages of the master, codes them in the ARINC 429 and sends them via the designated channel [34]. The EtherCAT Slave is the exposed hardware interface to the test bench on the system level.

Ethernet—is used as a medium for packing the EtherCAT packets into the Ethernet packets. The packets then pass through all nodes in the ring.

There are no components available on the market that combine EtherCAT with the ARINC 429. In the presented work, an EtherCAT master was configured, according to the EtherCAT standard and integrated into the Airbus A350-1000 HighLift test bench. In addition, a slave for the ARINC 429 was implemented in hardware and software. The implementation is described in the following.

3.1.1. Configuration and Integration of the EtherCAT Master

The EtherCAT Master was implemented, according to the specification [35]. For this purpose, an EtherCAT network information (ENI) file is created. It contains information about the EtherCAT network structure, the data structure of the network and the data structure of the individual slaves. From the listing of the tested masters, SOEM was not able to perform the configuration by the ENI files at the time of execution. In this case the EtherCAT network was realized by reading the configuration of the slaves. The given network configuration from Figure 8 contains a slave. The datagram from the master to the slave contains four 32-bit placeholders each for the receive and the transmit directions. This structure was derived from the ARINC 429 structure of the test bench, which uses 4-channel communication.

To demonstrate the portability, three master implementations were ported, integrated and tested. These implementations are:

TwinCAT—[36] the master implementation of the Beckhoff standard master.

SOEM—Simple Open EtherCAT Master [37]. An open-source implementation by RT-Labs.

IgH Master—[38]. An open-source implementation of IgH.

The integration of the software interface of the EtherCAT masters into the test bench was carried out by adapting the applications to the software interface of the EtherCAT masters.

3.1.2. Implementation of the ARINC 429 EtherCAT Slave

The implementation of the EtherCAT slave for the ARINC 429 is the main focus of this work. Firstly, the market was investigated for available integrated circuits for the implementation of EtherCAT and for the implementation of the ARINC 429. The architecture of the slave is shown in the Figure 9. The upper part shows the connection to the test bench through EtherCAT. On the lower side, the ARINC 429 connection to the SuT through four transmit and four receive channels is shown. The EtherCAT and the ARINC 429 ICs are managed by an application on a microcontroller in the middle. Both directions are connected to the microcontroller by SPI. In the following, the implementation is described in detail.

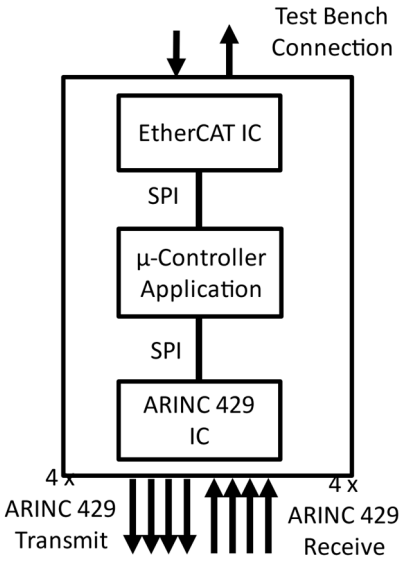


Figure 9. ARINC 429 EtherCAT Slave Components.

Implementation of the ARINC 429 connection: For the ARINC 429, only HOLT IC manufactures the integrated circuits with multiple channels and a standardized connection. Only the standard serial peripheral interface (SPI) is provided as the interface for the access of the control components. For the 4-channel implementation, the IC HI-3210 [39] was chosen. The HI-3210 acts as the SPI slave.

Implementation of the EtherCAT connection: The connection to EtherCAT was implemented in different ways. In order to demonstrate the transferability of the standardized solution, two different EtherCAT slave stacks and two different EtherCAT ICs were implemented and verified.

SOES—Simple open EtherCAT Slave [40] is an open source implementation of the EtherCAT standard. This implementation was transfered and tested on STM32 microcontrollers. The used EtherCAT IC are LAN9252 from Microchip [41] and AM3359 from Texas Instruments [42]. Both approaches are usable for the implementation.

SSC—EtherCAT Slave Stack Code ET9300 [34] is the open source slave implementation of the standard EtherCAT controller Beckhoff. It was ported to TI AM3359 and tested.

The same slave configuration was used for all three implementations. The process data object structure was adapted to that of the master.

3.2. Syntactic Description of the I/O

The configuration of the I/O interface, via the ICD, requires implementation efforts within the configuration component of the test bench and in the I/O, Figure 10. Within the configuration component, the section of the ICD is extracted, that concerns the I/O. This section is transferred to the EtherCAT Master. The EtherCAT Master forwards this section to the Slave part of the I/O, during the initial phase of the test execution. In the I/O, the ICD section is parsed. The extracted data is used to configure a look-up table, in order to keep the configuration data ready for the real-time access.

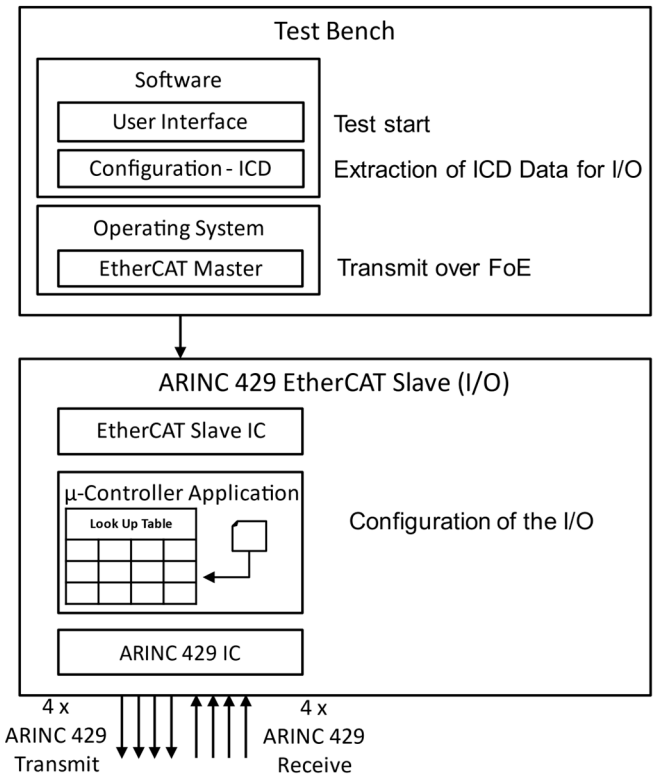


Figure 10. Configuration of the I/O with the ICD data.

3.3. Validation

This section presents the validation and the results of the implemented I/O interface. Within the project, the I/O was validated using several scenarios. This article will present three scenarios, two of which will be described briefly and one in detail.

3.3.1. Application Scenario–Test Bench Interconnection

Objective: this scenario aims at showing the applicability of the EtherCAT based I/O within other Industry 4.0 adaptations in the test benches, namely the connection of different test benches across enterprise boundaries. The setup was operated, using interconnection with different benches, to show the application of interconnection, configuration and automation standards. The network comprised five test benches of different test bench suppliers. A dedicated test bench was used inside the network for the test script execution. The EtherCAT I/O interface was used only to collect the data and to loop it back to the network (see Figure 11).

Boundary conditions: the focus of this article is not the interconnection. It is only on the behavior of the I/O.

System under test description: no actual system under test was used. A dummy system was putting the data back in the loop.

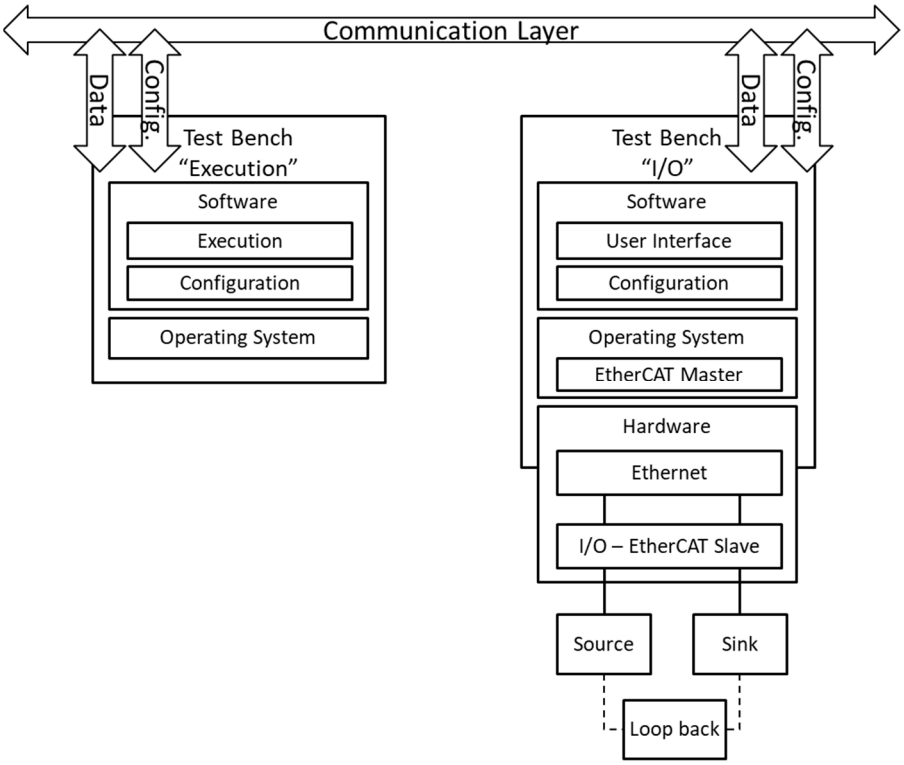


Figure 11. Test bench interconnection.

Test scenario: Figure 11 shows the rough architecture of the test setup. Within the connection of several test benches, only two were communicating. Test bench *Execution* was leading the test. That means test bench *Execution* starts the test, addresses the relevant test bench (I/O), and puts it in operation mode. The transition to the operation mode includes the receiving of the configuration for the syntax of the I/O.

Result: the test included a permanent transmitting and receiving of variables until the user stops. No variables were lost, no variables were received after the loop back with a wrong syntax.

3.3.2. Application Scenario–Test Slat Electric Motor Failure Injection

Objective: the objective of this scenario was to test the application of the EtherCAT I/O using the faster cycle times of the fast ARINC 429 at 100 kb/s.

System under test description: the system under test is an electric motor driving the leading edge (slat) of the wings.

Test scenario: for the test, the A350-1000 slat was extended and retracted. Initially, the data was only monitored by the test bench. During the second run, two types of failures were injected. First, the motor was set to overheat. Second, the rotation speed of the motor was set to 0. The SFCC was expected to behave accordingly to requirements.

Result: the SFCC accepted the EtherCAT I/O during all operations.

3.3.3. Application Scenario–Functional Test of the Airbus A350-1000 HighLift System

Objective: the goal of this application scenario is to demonstrate the proof of concept of the replacement of the classical I/O cards with standardized components, as well as their syntactic configuration. Specifically, the case aims to show that the I/O module:

- routes the ARINC 429 bus over the standardized I/O and does not compromise the real-time capability of the SuT and the test bench,
- can be configured via the test bench and the standardized medium,
- can translate the test bench variables into ARINC 429 messages,
- can translate the ARINC 429 messages into test bench variables.

Based on these criteria, the case must provide all coding procedures for evaluating the test bed variables.

Boundary conditions: since the HighLift test bench operates at cycle rates of 500 µs, the real-time conditions must not exceed this time frame. SFCC operates at slower rates. The ARINC 429 minimum interval is at 10 ms, in the selected scenario at a minimum 60 ms.

System under test description: The HighLift system controls the leading and trailing edges of the wing, according to (i) pilot control (via the flap lever) and (ii) automatic control functions (initiated by other aircraft systems via the ADIRU bus).

Test scenario: For the evaluation of the implementation of this work, the flap load relief function of the Airbus A350-1000 SFCC is used. It controls the flaps, according to the CAS values received from ADIRS via the ARINC 429 media. The objective of the function is to protect the landing flaps from damage or total loss. The damage would occur when the aircraft is operating at high speeds with the flaps extended. The air load on the flaps may become too high that the flap shaft breaks.

Figure 12 shows the semi-integrated SuT. The DuT are two redundant SFCCs. They are connected to an electric motor and to the mechanical HighLift system. The input signals, such as the aircraft speed over ground (Calibrated Air Speed—CAS), are provided by ADIRS via two independent ARINC429 buses. The redundant control computers perform a calculation, based on the three values. The resulting CAS selection is fed into the flap load relief function. Depending on the position of the system and the speed of the aircraft, the flaps are automatically retracted or expanded. In addition, other sensor values are evaluated by the SFCC and used to calculate the flap load relief function. These values are additionally provided to other aircraft systems via the ARINC429. It is expected that the landing flaps will remain extended during take-off and will be retracted with the increasing speed delivered via ADIRU.

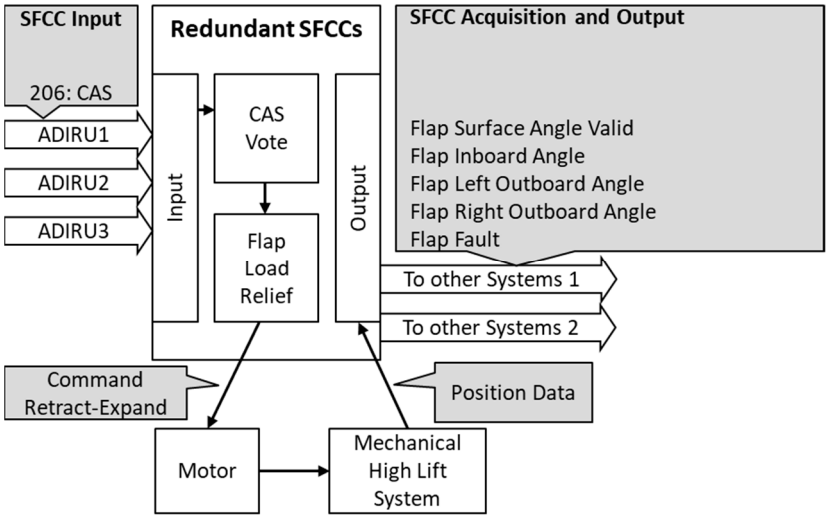


Figure 12. Architecture of the flap load relief test scenario.

3.3.4. Measurement Setup for Measuring the Signal Propagation Delay

To ensure that the real-time capability of the SuT is guaranteed and not compromised by the standardized I/O, the signal propagation time of each message is measured. The rate at which the ARINC 429 operates is 12,500 bits per second. With a message length of 32 bits, this results in a minimum signal propagation time of 32 (bits)/12,500 (bits/sec) = 2.56 ms. This is the minimum time required by an ARINC 429 message for transmission in the medium, according to the standard [43] (p. 45-14). The test bench, including the SFCC as the SuT, can handle delay times of 3.5 ms.

Figure 13 shows the setup for the time measurement before and after the messages have passed the standardized I/O module. An ARINC 429 signal analyzer was used for the time measurement. The ARINC 429 messages were respectively recorded before and after

passing the ARINC 429 I/O (e.g., $t_{ADIRU1} - t_{ADIRU1'} = t_1$). The ARINC 429 analyzer records the messages and marks them with a timestamp. The time difference of a message from ingestion over, say, ADIRU1 and ADIRU1′, determines the dwelling time of the message in the test bench. The delays of the proposed standardized I/O can be shown this way.

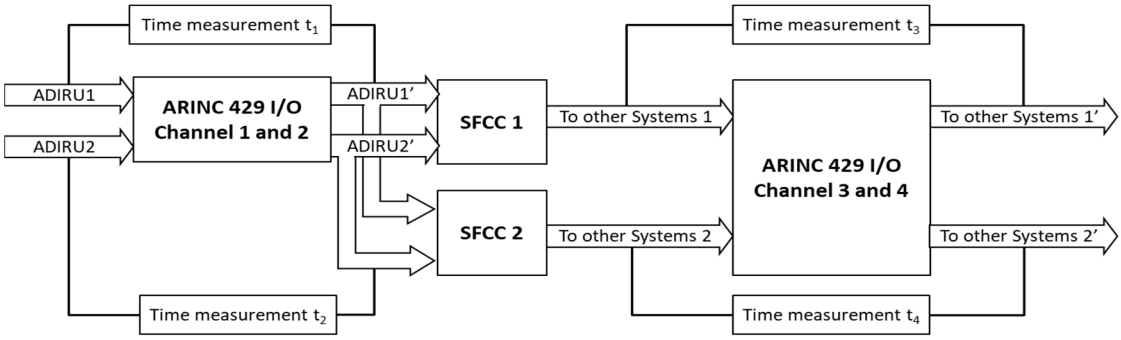


Figure 13. Setup for the time measurement.

3.3.5. Replaceability of the I/O Cards Used

The measurements were first performed by running the flap load relief function for a single channel (1xreceive, 1xtransmit), later including all four channels (4xreceive, 4xtransmit). This procedure provides information about the possible effects of the performance load on the delay times.

Figure 14 shows the cumulative results of the 1-channel measurement. The number of transmitted messages per label number results from different repetition rates. For the transmitted messages, the dwelling time exceeds 3.2 ms. The SFCCs operates properly during the test. None of the transmitted messages are lost. The time difference is determined between the points ADIRU1 and ADIRU1′.

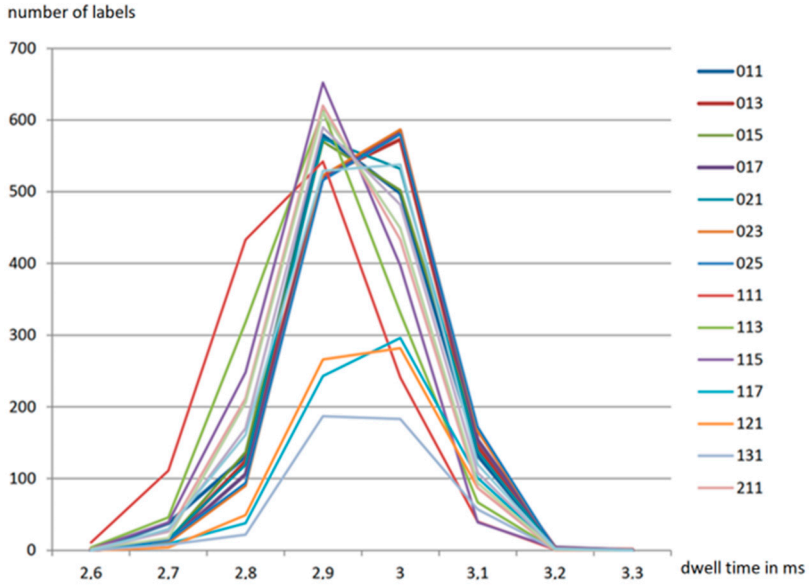


Figure 14. 1-Channel Measurement for the flap load relief function.

Figure 15 shows the cumulative results of the 4-channel measurement. The number of transmitted messages per label number results from different repetition rates. For the transmitted messages, the dwelling time never exceeds 3.4 ms. The SFCCs operate properly during the test. None of the transmitted messages are lost. Messages with dwelling times greater than 3.2 ms are delayed, due to execution start and stop. The time difference is measured between the points ADIRU1 and ADIRU1′.

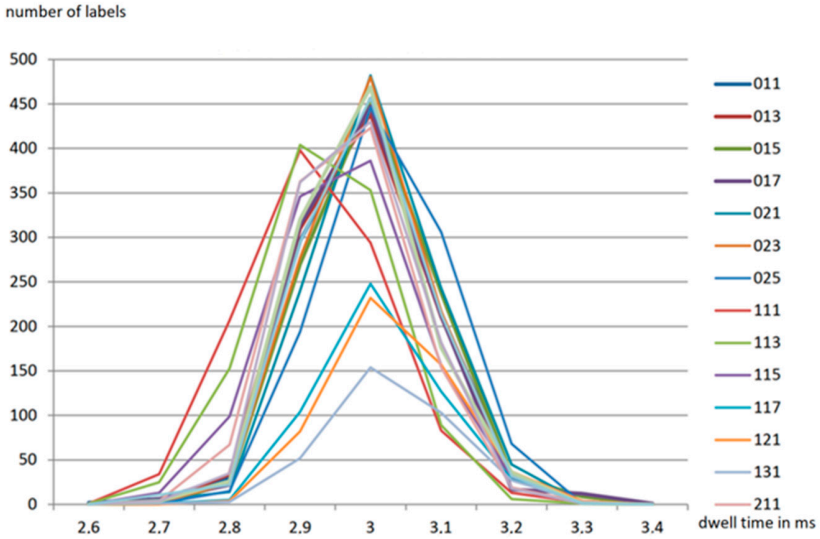


Figure 15. 4-Channel Measurement for the flap load relief function.

4. Discussion and Outlook

4.1. Challenges and Findings

The objective of the implementation of the standardized industrial Ethernet-based I/O have been achieved. The standardized I/O can operate in the testing environment of flight controls. However, the objectives of this research were purely technical. The impact of obsolescence is economically studied in the literature and norms/standards. The economic efficiency of the presented approach remains to be quantified. The technical modifications to the existing test benches were complex and far-reaching. Especially the integration of the EtherCAT master was complex.

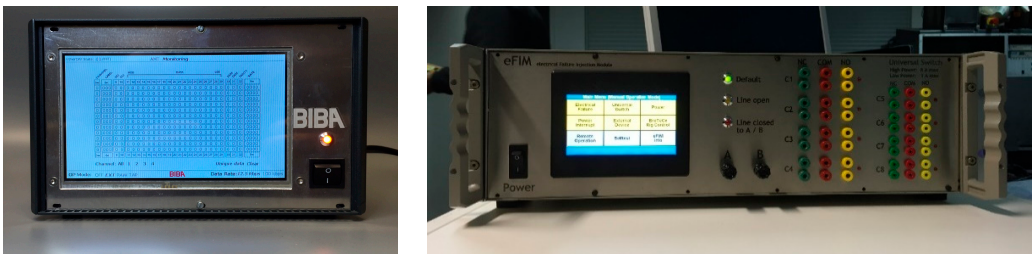
The use of the industrial Ethernet/EtherCAT in test benches only makes sense if a sufficient number of suppliers support these standards. If they do not, a failure of even the standardized I/O will mean the obsolescence of the test bench.

4.2. Implication for Practice

As in this article, obsolescence was the main objective in the research and development project as well. However, the proof of concept brought additional advantages. These advantages are:

Costs—in the implementation, COTS components were used. This brought a significant cost reduction, compared to the current available field programmable gate array (FPGA) based implementations.

Application to different avionic standards—the same EtherCAT infrastructure, including the slave logic, were used to implement (linear variable differential transformer) the LVDT measuring and the electric failure injection, such as the open circuits and short circuits (see Figure 16). Their integration in the test bench does not require any additional implementation as the EtherCAT infrastructure is implemented.



(**a**) (**b**)

Figure 16. EtherCAT based- (**a**) ARINC 429 (**b**) electrical failure injection I/O.

Additional functions—the implemented I/O modules are capable to apply failure injections directly in the I/O, without the transmitting of the messages to the test bench. This allows to relieve the load of the test bench and to decrease the latency of the messages on the ARINC 429 bus.

Scalability—most test benches come with four PCI or PCIe interfaces. One of the interfaces are consumed, an additional server needs to be acquired and integrated. Industrial Ethernet standards allow to combine 65,535 units.

4.3. Brief Discussion on Standards

The work is based on the widely accepted assumption in standards, as well as in the scientific and technical literature, that standards prevent obsolescence and create interoperability. By no means, everything that has become a standard actually contributes to interoperability and helps against obsolescence. One example is the IEEE's 802 committee. It has produced standards, such as 802.3 (Ethernet), 802.11 (WiFi) and 802.15 (Bluetooth and Zigbee). Yet 11 of the 21 working groups are no longer in existence [27] (p. 73). They have not contributed to the fight against obsolescence.

There are standards that fail to gain acceptance because they are inadequate, inconsistent, underspecified, or overspecified [44]. Other standards disappear even though they had already become leading standards, because a simpler or better applicable standard has emerged. An example of this is IEEE-1394 FireWire. It has become the standard for camcorders, yet has been superseded by USB 3.0, as the standard in the domain [45]. Even though FireWire is no longer the leading standard for camcorders, plug-in cards are available to keep camcorders with FireWire ports from becoming obsolete [46].

Ref. [46] showed five features of standards which describe their advantage. They address: (i) diffusion, (ii) extensions for purpose, (iii) supplier support, (iv) stability of standardization authority, and the (v) support of key industry players.

Author Contributions: Idea development: K.K. and K.-D.T.; conceptualization: K.K. and K.-D.T.; software, K.K.; validation, K.K.; writing—original draft preparation, K.K.; writing—review and editing, K.K. and K.-D.T.; visualization, K.K.; supervision, K.K. and K.-D.T.; project administration, K.K.; funding acquisition, K.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research has been funded by the German Federal Ministry for Economic Affairs and Climate Action (BMWK) in the project STEVE (project number 20Y1301G) and AGILE-VT (project number 20X1730D).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: Many thanks to Nicolaus Bamstedt, Wolfram Hilbrich and Volker Meyer for the support.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Seabridge, A.G.; Moir, I. *Design and Development of Aircraft Systems*, 3rd ed.; John Wiley & Sons: Hoboken, NJ, USA, 2020; ISBN 1119611512.
2. Rojo, F.J.R.; Roy, R.; Shehab, E. Obsolescence management for long-life contracts: State of the art and future trends. *Int. J. Adv. Manuf. Technol.* **2010**, *49*, 1235–1250. [CrossRef]
3. De Florio, F. *Airworthiness: An Introduction to Aircraft Certification and Operations*, 3rd ed.; Elsevier Science: Saint Louis, MO, USA, 2016; ISBN 9780081008881.
4. Robinson, B.; Hughes, B.; Bauer, R.; Harnack, J. Proactively managing obsolescence with test system architecture. In Proceedings of the 2015 IEEE Autotestcon, National Harbor, MD, USA, 2–5 November 2015; pp. 87–92, ISBN 978-1-4799-8190-8.
5. Lampl, T.; Wolf, T.; Hornung, M. Preliminary design of advanced flight control system architectures for commercial transport aircraft. *CEAS Aeronaut. J.* **2019**, *10*, 613–622. [CrossRef]

6. Spitzer, C.R. *Digital Avionics Handbook*, 3rd ed.; CRC Taylor & Francis Group: Bosa Roca, FL, USA, 2015; ISBN 9781439868614.
7. Hans, C.; Hribernik, K. NFF Special Session—Potentials of Applying Methods, Tools, Processes and Knowledge from Testing in Product Development to the NFF Problem. *Procedia CIRP* **2014**, *22*, 53–58. [CrossRef]
8. Jandaurek, K.; Johst, M. Development Trends and Innovations in Aerospace System Testing Using the Example of High-Lift. In Proceedings of the 55th AIAA Aerospace Sciences Meeting, Grapevine, TX, USA, 9–13 January 2017; American Institute of Aeronautics and Astronautics: Reston, VA, USA, 2017. ISBN 978-1-62410-447-3.
9. Bartels, B.; Ermel, U.; Sandborn, P. *Strategies to the Prediction, Mitigation, and Management of Product Obsolence*; Wiley: Hoboken, NJ, USA, 2012; ISBN 9781280588426.
10. *DIN EN 62402:2017-09*; Obsoleszenzmanagement (IEC 56/1716/CD:2016). Deutsche Institut für Normung; Beuth Verlag GmbH: Berlin, Germany, 2017.
11. Sandborn, P. Design for Obsolescence Risk Management. *Procedia CIRP* **2013**, *11*, 15–22. [CrossRef]
12. Zeng, M.L. Interoperability. *KO* **2019**, *46*, 122–146. [CrossRef]
13. Deputy Director for Engineering. *Modular Open Systems Approach (MOSA) Reference Frameworks in Defense Acquisition Programs*; Deputy Director for Engineering: Washington, DC, USA, 2020.
14. AUTOSAR GbR. Specification of I/O Hardware Abstraction, 2006. Available online: http://www.autosar.org/download/AUTOSAR_SWS_IO_HWAbstraction.pdf (accessed on 15 August 2022).
15. Association for Standardization of Automation and Measuring Systems. *ASAM XIL: Generic Simulator Interface*, 2.2.0th ed.; ASAM: Höhenkirchen, Germany, 2020.
16. Langmann, R. *Vernetzte Systeme für die Automatisierung 4.0: Bussysteme—Industrial Ethernet—Mobile Kommunikation—Cyber-Physical Systems*; Hanser: München, Germany, 2021; ISBN 9783446469846.
17. Obermaier, R. *Handbuch Industrie 4.0 und Digitale Transformation: Betriebswirtschaftliche, Technische und Rechtliche Herausforderungen*; Springer Fachmedien Wiesbaden GmbH: Wiesbaden, Germany, 2019; ISBN 9783658245764.
18. Martinen, D.H.; Lagalaye, M.; Pfefferkorn, J.; Casteres, J. Modular and Open Test Bench Architecture for Distributed Testing. In SAE Technical Paper Series, Proceedings of the AeroTech Congress & Exhibition, Fort Worth, TX, USA, 26 September 2017; SAE International400 Commonwealth Drive: Warrendale, PA, USA, 2017.
19. Object management Group. *OMG Data Distribution Service: DDS*, 1.4th ed.; Object Management Group: Boston, MA, USA, 2015.
20. Franke, M.; Thoben, K.-D. Interoperable Test Cases to Mediate between Supply Chain's Test Processes. *Information* **2022**, *13*, 498. [CrossRef]
21. West, S.; Ebel, M.; Anderson, M.; Stoll, O.; Poeppelbuss, J.; Khan, M. Nested Lifecycles-Improving the Visibility of Product Lifespans in Smart Factories. *Front. Manuf. Technol.* **2022**, *2*, 837478. [CrossRef]
22. Moir, I.; Seabridge, A.G.; Jukes, M. *Civil Avionic Systems*, 2nd ed.; Wiley: Chichester, UK, 2013; ISBN 9781118536728.
23. Aceto, L. *Reactive systems: Modelling, specification and verification*; Cambridge University Press: Cambridge, UK, 2007; ISBN 9780521875462.
24. Zöbel, D. *Echtzeitsysteme: Grundlagen der Planung*, 2nd ed.; Springer Vieweg: Koblenz, Germany, 2020; ISBN 9783662604212.
25. Moir, I.; Seabridge, A. *Design and Development of Aircraft Systems*, 2nd ed.; Wiley: Reston, VA, USA, 2013; ISBN 9781118469149.
26. Adame, T.; Carrascosa-Zamacois, M.; Bellalta, B. Time-Sensitive Networking in IEEE 802.11 be: On the Way to Low-Latency WiFi 7. *Sensors* **2021**, *21*, 4954. [CrossRef] [PubMed]
27. Tanenbaum, A.S.; Feamster, N.; Wetherall, D. *Computer Networks*, Sixth ed.; Global Edition; Pearson: Harlow, UK, 2021; ISBN 9781292374017.
28. Wang, X.; Chen, Y.; Fang, D.; Zhong, J.; Wang, Y. Design and Implementation of EtherCAT Master of Gyrowheel under RTX System. In Proceedings of the 39th Chinese Control Conference, Shenyang, China, 27–29 July 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 4515–4520, ISBN 978-9-8815-6390-3.
29. Salman, T.; Jain, R. *Networking Protocols and Standards for Internet of Things. Internet of Things and Data Analytics Handbook*; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 2016; pp. 215–238.
30. Annighoefer, B.; Brunner, M.; Schoepf, J.; Luettig, B.; Merckling, M.; Mueller, P. Holistic IMA Platform Configuration using Web-technologies and a Domain-specific Model Query Language. In Proceedings of the 39th DASC—Digital Avionics Systems Conference, San Antonio, TX, USA, 11–15 October 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–10, ISBN 978-1-7281-9825-5.
31. Aeronautical Radio Incorporated. *ARINC 653: Avionics Application Software Standard Interface*; ARINC: Singapore, 2006.
32. Bauer, O.; Mattis, S. Entwicklung von Standards zur Etablierung von Methoden der Modellgetriebenen Entwicklung in der Luftfahrtindustrie im Verbund: STEVE—System-Technik und virtuelle: STEVE—System-Technik und virtuelle Erprobung—Schlussbericht zum Verbundvorhaben LuFo V-1, Poing, 2018. Available online: https://www.tib.eu/en/suchen/id/TIBKAT:1023331144/ (accessed on 15 August 2022). [CrossRef]
33. Halle, M.; Thielecke, F. Next generation IMA configuration engineering—From architecture to application. In Proceedings of the 2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC), Prague, Czech Republic, 13–17 September 2015.
34. *Application Note ET9300: EtherCAT Slave Stack Code*; No. 1.8; Beckhoff Automation GmbH & Co., KG: Verl, Germany, 2014.
35. *EtherCAT Specification*; No. 1.0.4; EtherCAT Technology Group: Nürnberg, Germany, 2007.
36. *TwinCAT 3: Product Overview*; No. 1.11; Beckhoff Automation GmbH & Co., KG: Verl, Germany, 2022.
37. RT-Labs. Simple Open EtherCAT Master: SOEM Documentation. Available online: www.rt-labs.com/docs/soem/start.html (accessed on 1 August 2022).

38. Pose, F. *IgH EtherCAT Master 1.5.2: Documentation*; Essen, Germany, 2017. Available online: https://www.iram.fr/~{}blanchet/ethercat/ethercat_doc_fr.pdf (accessed on 15 August 2022).
39. *HI-3200, HI-3201: Avionics Data Management Engine/ARINC 429—CAN Bus Bridge*; Holt Integrated Circuits Inc.: Aliso Viejo, CA, USA, 2020.
40. RT-Labs. Simple Open EtherCAT Slave: SOEM Documentation. Available online: www.rt-labs.com/docs/soes/start.html (accessed on 20 August 2022).
41. *LAN9552: 2/3-Port EtherCAT Slave Controller with Integrated Ethernet PHYs*; Microchip Technology Inc.: Singapore, 2015.
42. *AM335x Sitara Processors*; Texas Instruments Incorporated: Dallas, TX, USA, 2020.
43. *Industrial Communication Technology Handbook*, 2nd ed.; Zurawski, R. (Ed.) CRC Press: Boca Raton, FL, USA, 2015; ISBN 9781315215488.
44. Lewis, G.A.; Morris, E.; Simanta, S.; Wrage, L. Why Standards Are Not Enough to Guarantee End-to-End Interoperability. In Proceedings of the Seventh International Conference on Composition-Based Software Systems, Madrid, Spain, 25–29 February 2008; IEEE: Piscataway, NJ, USA, 2008; pp. 164–173, ISBN 978-0-7695-3091-8.
45. Gibbons, M. The Future of FireWire: Is FireWire Dying? 2012. Available online: https://link.gale.com/apps/doc/A307920719/AONE?u=anon~{}a9ba933f&sid=googleScholar&xid=4bb04488 (accessed on 10 August 2022).
46. *Item No. 89153: Delock PCI Express Card > 2 x External FireWire B + 1 x External FireWire A*; Delock: Dyer, IN, USA, 2017.