The growing level of aircraft systems complexity and software investigation

by **Paulo Soares Oliveira Filho**, Air Safety Investigations Manager, Embraer Air Safety Department.

1. Introduction

The aircraft design continues advancing on human-machine interface concepts in order to improve safety and facilitate the operation from the flight crew perspective. These advances also provide overall operational performance improvements for airlines. Most of these enhancements are obtained by using highly integrated onboard systems with intense usage of software which controls the majority functions including those considered as safety-critical.

However, improvements for pilots and airlines do not mean an easier life for aeronautical investigators when a deep examination of aircraft internal components and its interfaces is in order. Any factual evidences that could lead to scenarios involving possible system's flaws will request a great effort on understanding how the machine internally works.

At the same time, even for those cases when no clue of aircraft malfunctions are in view, with the objective to analyze aspects on human-machine interface, investigators possibly will be demanded to verify how information displayed to the pilots are generated and processed. This perspective can be associated also for general and executive aircraft designed or modified to receive glass cockpit.

Likewise, occurrences involving the automation aspect can demand a comparison between the pilot's mental model on how the aircraft functions work and how the machine works actually. Again, we have a situation to achieve an adequate level of comprehension regarding onboard systems.

The NASA Study on Flight Software Complexity [11] offers an interesting definition for complexity: *"(…) how hard something is to understand or verify. (...)"*

In this way, as the complexity of onboard systems grows, the challenge of investigators grows as well.

This paper aims to approach three major aspects:

- The reality of the growing aircraft systems complexity with intense usage of software.

- In the light of constant incoming technologies, the importance to revisit some aircraft system's concepts frequently adopted in the investigation process.

- An invitation for envisioning preparation measures to cope with complexity. In this way, a little contribution is offered on the topic "A practical approach for investigation on complex aircraft systems". This topic has the intention to be only an example of initiative in terms of guidance material and recommended practices that can be written in order to expand the set of references for investigators.

2. Uncovering hidden complexity

The aviation history has shown an increasing demand for improved onboard functionalities. Among the reasons for this demand, it is possible to list safety enhancements, performance improvements, and security issues. The below text is a good expression of this reality:

> "*Associated with the enhanced capability afforded by the technology, and as driven by the competitive pressures of the civil transport aircraft market, the functionality of avionics systems has continued to escalate."* Digital avionics handbook. SPITZER, Cary R (Editor). [8]

A direct observation of the cockpit panels on different aircraft generations shows indubitable growth of onboard resources to the pilots. However this assertion better applies to airplanes pre-glass-cockpit. Since the beginning of the glass cockpit fever, it is not so visually evident the amount of complexity behind the systems which is not directly associated with the panels. As an example, it is possible to mention the functionality called "Autobrake" which frequently corresponds to a single switch in the cockpit panels. Of course, the Autobrake switch could never give us an idea of how many lines of software code and how many interfaces with other systems were necessary in order to make this functionality come true.

In short, nowadays the real dimension of complexity growth on modern aircraft is quite hidden inside the onboard computers but pops out whenever its deep analysis is required.

Industry discussions about parameters and methods to measure software complexity are in place. For the purpose of this paper, the parameter of software size can help us.

As more onboard system functionalities are progressively being implemented, the size of software, measured in terms of source lines of code (SLOC), is believed to double on every four years. That trend has been observed for at least five decades as presented in the Figure 1.

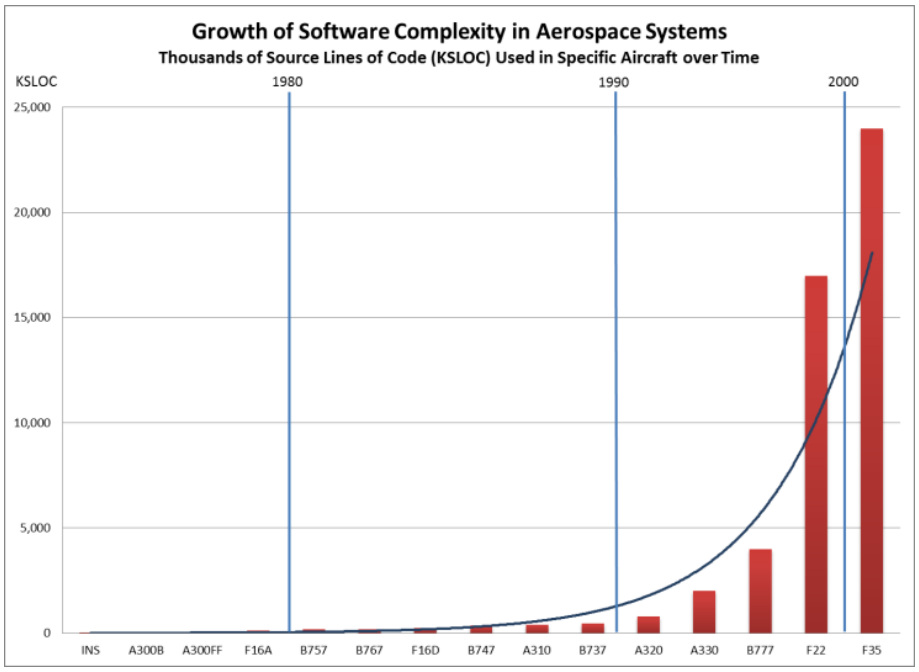

Figure 1: Growing of software complexity in aerospace systems [13]

Leveson [3] bring us a closer view of the different forms of complexity;

> *Complexity comes in many forms, most of which are increasing in the systems we are building. Examples include* ***interactive complexity*** *(related to interaction among system components),* ***dynamic complexity*** *(related to changes over time),* ***decompositional complexity*** *(where the structural decomposition is not consistent with the functional decomposition), and* ***nonlinear complexity*** *(where cause and effect are not related in a direct obvious way).* Engineering a safer world. Leveson, Nancy G.

3. Revisiting (or updating) some relevant concepts

The above mentioned "interactive complexity" can be understood as more intense interlacement among systems. In order to safely integrate such systems containing new technologies, the industry and authorities have developed new concepts and occasionally have revised some legacy ones.

As a consequence, investigators need to revisit some definitions with objective to establish a firm foundation to build analysis, conclusions and effective recommendations.

Updating, or even only revisiting, some legacy concepts and terminologies regarding aircraft systems will allow being better prepared for present and future complex investigations. Certainly, it is not the intention of this topic to try to identify all of these concepts. Instead, some examples have been selected to be explored here only for illustrative purposes.

3.1 Reviewing failure, fault and error concepts

The classic investigator's initial approach with respect to aircraft systems is to identify the existence of any failure which could be associated with the sequence of events that resulted in the mishap. There is nothing wrong with this mindset; however, a full understanding of the term "failure" (as used by the engineers that designed the aircraft) is in order. Additionally, the term "fault" can be possibly perceived as equivalent to "failure" by those people not long-term involved in aircraft design and maintenance. The term "failure" comes from pure mechanical systems era, when a function became inoperative or degraded frequently due to some jammed or broken part. While in software era, this term needs to be revisited.

It is worth mentioning that the classic "failure" concept is not applicable to software as it has no physical properties. Instead, certain software can be found in an undesired condition which occurs when the specific logic path that contains an error is executed. It means that an error may exist inside software but will never cause any consequence as long as it not executed by the processor. In this way, the error itself is not an event, but a state. However, it has a potential to ultimately cause the associated system to be inoperative, or no longer function as intended (according to the specifications).

Appendix 1 offers a compilation of failure, fault and error definitions. An attempt to organize it in a simple way can be found in Figure 2.

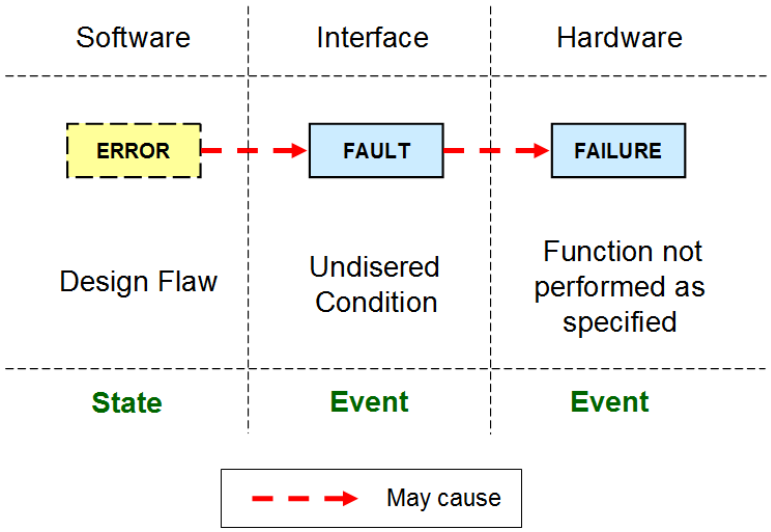

Figure 2 – Representation of error, fault and failure.

From the definitions, the term "fault" can be applied both to software and hardware. Aircraft systems design techniques can be used in order to detect faults and manage them in order to maintain the system fully functional or partially functional.

3.2 The role and challenge of integration

Investigations of highly-integrated systems require adequate tools, methods and the availability of a representative integration laboratory. Most of the times, it is not enough to perform tests/analysis of components separately. Figure 3 illustrates the evolution of avionics architectures and the increase of integration.

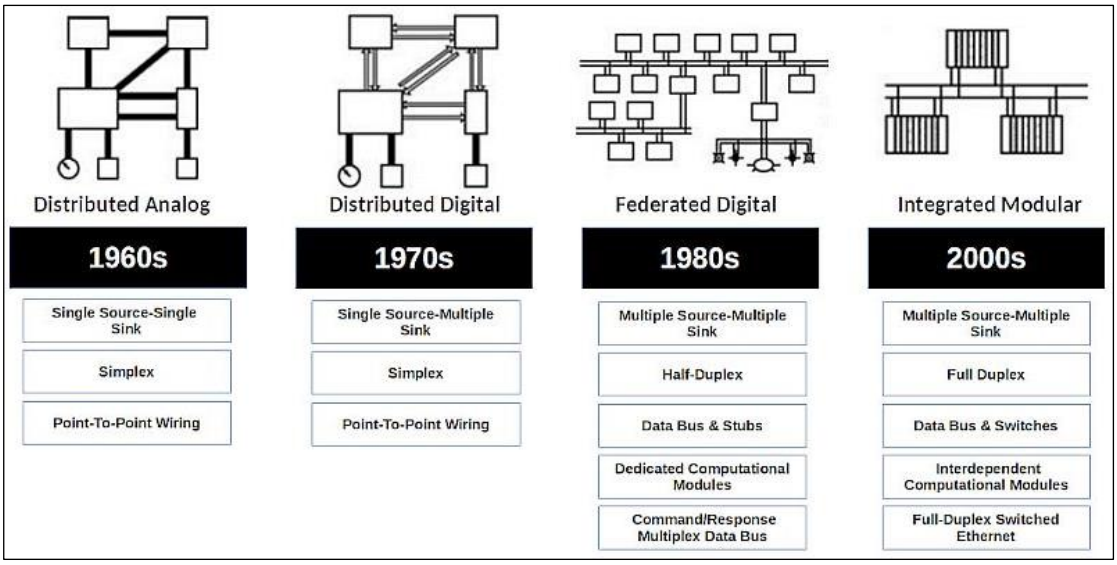

Figure 3: Growing level of integration in avionics architectures [8].

High integration means a greater number of interactions between parts, components and functionalities. During investigations, it is essential to distinguish system's misbehavior caused by single component anomaly and those that have root cause on the interactions between components tightly coupled.

Further reflection on these aspects leads us to consider that the investigation effort applied to understand and test the parts separately is required to be equivalent to the effort to understand, analyze and test the onboard systems as a whole, whenever possible. A further reflection could be to ask ourselves how to investigate interactions inside a software. Or even, how to investigate software at all.

3.3 Before jumping into the swamp of failures and faults

The strategy proposed is: first apply full effort to understand aircraft systems in normal conditions and, after that, start to approach occasional faults or failures.

The question here could be: what is the normal way to operate the system and what is the expected outcome. Normal way refers to the operation of the aircraft in accordance with the

approved procedures specified in the manuals. It includes respecting the approved operational aircraft limits/envelope. In this way, the aircraft manuals can be considered as an extension of the aircraft.

After all factual information has been collected from the accident site; it can be a big and natural temptation for the investigators to focus first on the eventual evidences of failures. However, understanding the system's normal operation first will avoid difficulties, and possible delays, on the effort to discover the failure mechanism.

You can invest some days to understand the normal operation and then some days on the failure, or, jump directly to the failure and eventually spend a month to fully understand how and why the failure occurred including its surrounding aspects. This is because complex systems include different modes of operations, protections, alerts, fault management strategies and application of fail-operational/fail-safe concept that only make sense if seen from the perspective of normal operation and human-machine-interface philosophy.

> *Fail-Operational: A characteristic design which permits continued operation in spite of the occurrence of a discrete malfunction.* FAA System Safety Handbook, Appendix A: Glossary

> *Fail-safe: A characteristic of a system whereby any malfunction affecting the system safety will cause the system to revert to a state that is known to be within acceptable risk parameters*. FAA System Safety Handbook, Appendix A: Glossary

3.4 Software versus Hardware

> *Software: Computer programs, procedures, rules, and associated documentation and data pertaining to the operation of a computer system.* FAA System Safety Handbook

Essentially software is an organized sequence of instructions to be executed by a processor. Instructions could be defined as ideas on "how to do". Therefore software is invisible, intangible and abstract.

As mentioned before, software has no physical properties and there are no physical laws underlying software behavior. Therefore there are no physical constraints on software complexity. You can write a software code as so complex as you wish.

The above aspects make evident that the differences between hardware/mechanical and software cannot be disregarded. The Table 1 is a proposal for mapping these main differences based on two books: Safeware – System Safety and Computers [2], and Software Safety and Reliability [4].

Table 1: Differences between Hardware/Mechanical and Software.

Hardware / Mechanical	Software
Subjected to wear-out.	There is no wear-out.
Some failures are due to wear, fatigue, overload or manufacturing issues.	The classic concept of failure is not applicable. A “fault” occurs when the logic path that contains an error is executed.
Reliability is time related and can be quantified.	Reliability is not time dependent and it is difficult to be quantified. Traditional reliability measures don’t apply.
Failure rates are somewhat predictable according to known patterns.	It is not consensus that failure rates can be directly associated to software.
Possible inspection or measurement.	It is not possible to perform direct visual inspection.
Preventive maintenance can be applied.	There is no equivalent to preventive maintenance for software.
Subjected to manufacturing variability.	Software can be replicated perfectly.
Hardware interfaces are tangible.	Software interfaces are conceptual.
A hardware or a mechanical device can exist without software.	There is no software without hardware. Pure software is useless; software exists only as part of a system. The software interface with aircraft happens only through the system’s hardware.

Hardware/mechanical parts may exhibit progressive malfunctions due to wear but without full interruption of operation. However, since there are no wear-out phenomena, software errors have a pattern to occur suddenly without previous clues or warnings.

According to Herrmann [4], unlike hardware, the cause of failures in firmware using embedded software is always systematic, not random.

3.5 The systems interaction with the environment

The last line of the Table 1 brings us an important aspect that needs to be explored. The hardware of an onboard computer is subjected to different external foreseen threats, as illustrated in Figure 4.

Any engineering measure to protect equipment against adverse external factors will always assume an envelope in terms of maximum and minimum values. Of course, it is impossible to design a component which resists, for example, an infinite high temperature. A certain value certainly needs to be defined by industry and authorities as adequate in terms of acceptable level of safety.

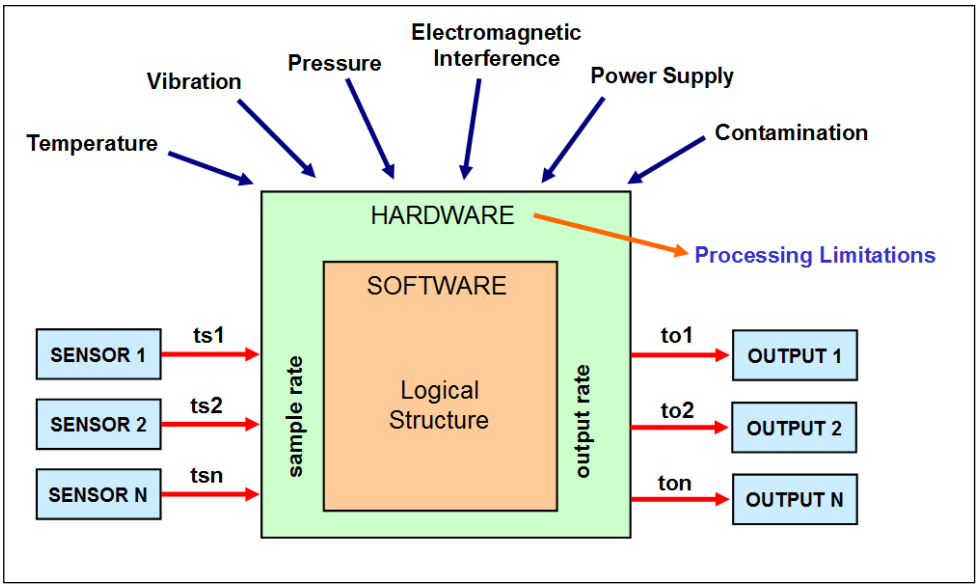

Figure 4 – Simplified schematics for hardware and software interface.

As the realities of equipment operation under actual environmental conditions are better understood, the engineering mitigation measures for these stresses are constantly being improved. A challenge for the investigator is that some of these threats are not measured, or even if it is measured, sometimes it is assumed that the value is not recorded. A good example is Electromagnetic Interference which can be produced by diverse sources, including lightning strikes.

However, as we live in the information society, every day, new sources of data become available to the public. In the recent years, meteorological science evolved in such a way that it is possible to obtain real-time lightning maps on Web (i.e. https://www.lightningmaps.org).

New reliable data sources are becoming available faster than a certain investigator can be aware. This aspect reinforces the value of sharing information, techniques, tips and tricks throughout the investigators' community. Additionally, it is essential to share also the assessment on how trusty a certain Web source is.

4. A practical approach for investigation of complex aircraft systems

This step-by-step must not be considered as definitive or seen as a formal manufacturer procedure, far from it. As mentioned before, it is only a little contribution to illustrate the embryo idea that some additional guidance material and recommended practices can be developed to be included on future investigator's toolbox.

The survival on a complex environment requires well established references and guidance, otherwise the investigator may find him or herself lost in a sea of information. The practices below can be useful for the investigators in the path to determinate failure mechanisms.

4.1 (Step #1) Obtaining all information regarding the aircraft configuration

Since the majority of system's behavior is determined by software it is essential to obtain the information about the software version of all involved components. For highly-integrated airborne systems, usually there is a so called "top level system part number", or software version (load) of the entire avionics suite which most of the times can be retrieved by accessing the central maintenance computers, or maintenance logs.

For distributed federated or avionics architectures, the only way is to verify the version of each component on ID plates. If the aircraft was totally destroyed, the maintenance records are the source to get this information.

Units that compose a distributed/federated avionics suite can be upgraded individually. However, there are not rare situations when the airframe manufacturer developed and certified a software upgrade of a certain component together with the upgrade of other unit(s). This can happen also for hardware upgrades. In this case, the upgrade is made involving a group of components, in accordance with the service bulletin issued by the manufacturer.

The operators need to pay special attention to what are the approved configurations in terms of component software/hardware versions. In other words, components that are individually airworthy (i.e. FAA form 8130), not necessarily will compose an airworthy configuration in the aircraft.

Intermixing components in a non-approved configuration may cause unpredicted consequences in terms of misbehavior, malfunctions and failures. The adverse effects can appear not necessarily during startup but maybe only during flight. If no proper hardware/software configuration analysis is performed, the investigation will probably become jammed or entangled in the net of conflicting information.

4.2 (Step #2) Collecting and analyzing onboard recorded data

It is well known that crash recorders are not the only possible source of onboard recorded data. From late 80's, the majority of onboard electronic units and modules have internal Non-Volatile Memories (NVMs) which may record fault/failure codes for maintenance purposes. The data retrieved from Central Maintenance Computer (CMC) is a top priority NVM as it usually concentrates status information from a set of components. In highly-integrated systems, the interpretation of the presence/absence of fault/failure codes in the NVMs is very dependent on the hardware/software versions. It is recommended that the interpretation of the logs of fault/failure codes is made meticulously, in teamwork, involving investigation authorities, component manufacturer, airframe manufacturer and the operator's representative.

Regarding the crash recorders, it is recommended that the analysis of the flight data recorder (FDR) has the involvement of the same team that worked on NVMs, in order to ensure the right correlation between the FDR data and NVM data.

4.3 (Step #3) Reproducing scenarios in a controlled environment

The third important step is to try to reproduce the aircraft fault/failure condition in a controlled environment. The concept of a controlled environment can be understood as an aircraft integration laboratory, "iron bird", or, even the aircraft under investigation eventually if it has not been significantly damaged in such way that the repair will not affect the representativeness of the sub-system(s) under investigation. Another aircraft tail number of the same model can be used as a controlled environment as long as it is free of malfunctions and using the same configurations of software/hardware. Variations can be accepted if an engineering analysis demonstrates that the differences will not affect its representativeness.

Before running any test or analysis, the setup of the controlled environment needs to represent, as close as possible, the systems conditions present at the moment of the occurrence.

The majority of in-flight conditions are very difficult to be reproduced on actual highly-integrated aircraft on the ground. Even an aircraft integration laboratory requires a good pre-test planning and hours of setup depending on the specific desired in-flight conditions

In terms of engineering, it is almost impossible to fully understand the failure mechanism without reproducing it in a controlled environment. No effective engineering effort to design a technical solution or correction is possible to be made without a full comprehension of the failure mechanism.

Note that Full Flight Simulators (FFS) used for pilot's training may not be adequate CEs for going deep into aircraft systems for failure investigation. A FFS aims to reproduce the predicted behavior of the onboard systems with the focus on cockpit effects and does not necessarily use the same hardware/software of an actual aircraft.

4.4 (Step #4) Software Investigations

For the purposes of this paper, “software investigation” refers to the activities performed on aircraft systems with the objective to understand their behavior as a result of software design and investigate possible flaws.

It is important to note that in a modern avionics suite, most interactions between parts, components and functionalities are virtually established at software level.

The identification of a flaw in the software can be achieved only if the error/malfunction/fault/failure is found or reproduced during the activities performed on the CE. Investigators need to be aware that, even applying the best effort on the CE, unfortunately the situation experienced on the occurrence under investigation may never be reproduced. This is because the software of a highly-integrated aircraft uses a lot of input variables and eventually an error became evident only in a very specific combination of input values and processing status. About this subject, Leveson [2] states that:

> *“Even if the possibility of software error is investigated, subtle errors that cause accidents in well-tested and sometimes long-used systems are not easy to find (or to prove that they may or not exist)”.*

Note: In the above excerpt, the author uses “accident” as a general term, not specifically in the context of aeronautical mishap investigation (ICAO Annex 13).

5. Conclusions

- The real dimension of complexity growth on modern aircraft is hidden inside the onboard computers.
- Software is invisible, intangible and there are no physical laws underlying software behavior. Traditional investigation concepts and techniques not necessarily apply.
- It is essential to retrieve any information regarding the aircraft configuration in terms of hardware/software versions.
- Components that are individually airworthy (FAA form 8130), not necessarily will compose an airworthy configuration in the aircraft. Non-approved configurations may cause unpredicted consequences in terms of misbehavior, malfunctions and failures which can be virtually impossible to be investigated in case of total loss.
- A planned usage of an adequate controlled Eevironment is a key factor for the success of the investigations involving highly-integrated onboard systems.
- Aircraft complexity and the unfolding challenges cannot be eliminated but can be managed through adequate training, specific guidance, clarification and harmonization of relevant concepts/terminology.
- It is worth to timely share indications about new trusty data sources (especially those Web on-line) which can be strategic for the investigators’ community.

6. References

1. GUNSTON, Bill. The cambridge aerospace dictionary. 2nd Ed., Cambridge University Press, YYYY.
2. LEVESON, Nancy G. Safeware – system safety and computers. University of Washington. YYYY.
3. LEVESON, Nancy G. Engineering a safer world : systems thinking applied to safety. MIT. 2011.
4. HERRMANN, Debra, S. Software safety and reliability. IEEE Computer Society. YYYY.
5. NICHOLS, William R.; SHEARD, Sarah. FAA Research Project on System Complexity Effects on Aircraft Safety: Candidate Complexity Metrics Software Engineering Institute, Carnegie Mellon University, May 2015.
6. AC 25.1309-1B (Arsenal). System design and analysis. FAA, 2002.
7. ARP 4754. Certification considerations for highly-integrated or complex aircraft systems. SAE International, 1996.
8. SPITZER, Cary R (Editor). Digital avionics handbook. 2nd Ed. CRC Press 2007.
9. OSTROUMOV, I. Avionics: digital data buses. Available: <https://de.slideshare.net/ostroumov/avionics-digital-data-buses>. Accessed: July 25th, 2018.
10. System Safety Handbook, Appendix A: Glossary, FAA, 2000.
11. NASA Study on Flight Software Complexity, Final Report, 2009.
12. TEK, Miles. MIL-STD-1553B in avionics: where data networking has been and where it's going. April, 2017. Available: <https://www.intelligent-aerospace.com/articles/2017/04/mil-std-1553b-in-avionics-where-data-networking-has-been-and-where-it-s-going.html> Accessed: July 25th, 2018.
13. Motivation for advancing the SAVI program. Available: <https://savi.avsi.aero/about-savi/savi-motivation/>, Accessed: July 25th, 2018
14. Sources: https://de.slideshare.net/ostroumov/avionics-digital-data-buses. https://www.intelligent-aerospace.com/articles/2017/04/mil-std-1553b-in-avionics-where-data-networking-has-been-and-where-it-s-going.html

7. Appendix 1

Failure (Cambridge Aerospace Dictionary): Separation of a part into two or more pieces that the part is no longer complete (FAA).

Failure (AC 25.1309–1B Arsenal Draft): An occurrence that affects the operation of a component, part, or element such that it can no longer function as intended.

Failure (SAE ARP 4754): The inability of an item to perform its intended functions.

Failure (IEEE Standard 610.12-1990) [DO178B]: The inability of a system or component to perform its required functions within specified performance requirements [limits].

Fault (NASA-STD-8719.13A): Any change in state of an item that is considered to be anomalous and may warrant some type of corrective action.

Fault (DO-254) - (1): A manifestation of a flaw in hardware due to an error or random event. A fault, if it occurs, may cause a failure. (2) An undesired anomaly in an item.

Fault (DO-178C): A manifestation of an error in software. A fault, if it occurs, may cause a failure.

Error (Cambridge Aerospace Dictionary): In electronic data-processor, or processing, incorrect step, process or result , whether due to machine malfunction or human intervention.

Error (DO-178C): Error – With respect to software, a mistake in requirements, design, or code.

Error: (N.G.Leveson): An error is a design flaw or deviation from a desired or intended state.

Error (FAA System Safety Handbook, Appendix A: Glossary): A mistake in engineering, requirement specification, or design, implementation, or operation which could result in a failure, and /or contributory hazard.

Error (FAA AC 25.1309–1B Arsenal Draft): An omission or incorrect action by a crewmember or maintenance personnel, or a mistake in requirements, design, or implementation. Note: Errors and events may cause failures or influence their effects, but are not considered to be failures.