Computer Standards & Interfaces 36 (2014) 889–898

Contents lists available at ScienceDirect

Computer Standards & Interfaces

journal homepage: www.elsevier.com/locate/csi

Software and hardware certification of safety-critical avionic systems: A comparison study

Wonkeun Youn *, Baeck-jun Yi [1]

Aerospace Safety & Certification Center, Korea Aerospace Research Institute, 169-84 Gwahangno Yuseong-gu, Deajeon 305-806, Republic of Korea

ARTICLE INFO

Article history:
Received 15 November 2012
Received in revised form 7 January 2014
Accepted 20 February 2014
Available online 13 April 2014

Keywords:
Software and hardware quality assurance
Airborne software and hardware
Safety certification
DO-178B
DO-254

ABSTRACT

To ensure the safety of avionic systems, civil avionic software and hardware regulated by certification authorities must be certified based on applicable standards (e.g., DO-178B and DO-254). The overall safety integrity of an avionic system, comprising software and hardware, should be considered at the system level. Thus, software and hardware components should be planned, developed and certified in a unified, harmonized manner to ensure the integral safety of the entire avionic system. One of the reasons for the high development costs of avionic systems complying with standards may be a lack of sufficient understanding of how to employ these standards efficiently. Therefore, it is important to understand the similarities and differences between DO-178B and DO-254 to effectively manage the processes required by these standards, to minimize cost, and to ultimately ensure the safety of the entire avionic system. Thus, the goal of this paper is to compare various aspects of DO-178B and DO-254 comprehensively. The paper may serve as a useful supplementary material for the practitioner to understand the rationales behind and the differences between two main standards used in avionic industries.

Contents

* Corresponding author. Tel.: +82 42 870 3542; fax: +82 42 870 2501.
E-mail addresses: wkyoun@kari.re.kr (W. Youn), ybj@kari.re.kr (B. Yi).
[1] Tel.: +82 42 860 2502; fax: +82 42 870 2501.

http://dx.doi.org/10.1016/j.csi.2014.02.005
0920-5489/

1. Introduction

Over the past several decades, safety has been a critical issue in many embedded applications in aerospace, aircraft, road vehicles, railways, nuclear systems, and implanted devices because the failure/malfunction of a safety-critical system may cause catastrophic damage or loss of life [1]. Society has the obligation to protect itself, and governments and industry organizations have established guidelines and standards for engineers to follow in developing systems in these areas. In particular, the introduction of such guidelines and standards started relatively early in the aviation industry due to the serious consequences of aircraft-related accidents.

To ensure that newly developed aircraft systems are designed and built to comply with applicable regulations and the highest levels of safety integrity, certification is mandatory in every country before a new aircraft system is put into operation. The definition of certification is the "procedure by which a third-party gives written assurance that a product, process, or service conforms to specified requirements [2]." For example, every country needs to certify new aircraft to assure compliance with applicable airworthiness requirements before it is cleared for flight. This procedure is known as type certification; a certification authority approves one sample of the developed system for flight usage [3].

Modern safety-critical systems for avionics utilize not only an increasing amount of sophisticated software but also a software-embedded hardware to process the large amount of data needed to control avionic systems and monitor their current status [4]. Avionics safety is considered at the system level and has no important implications when considered separately with regard to software and hardware [5]. The failure or malfunction of software can be due to interactions with hardware. Additionally, software and hardware domains have mutual influence on each other during aviation system development. Consequently, the software and hardware components of safety-critical systems must be developed and certified in a unified manner to ensure the integral safety of the entire avionic system [4].

To assure the reliability of the software/hardware and to ultimately ensure the safety of passengers, the U.S. Federal Aviation Administration (FAA) requires software/hardware certification suited to the development of safety-critical systems [6]. The FAA accepts standards developed by the Radio Technical Commission for Aeronautics (RTCA) for the reliability and safety that are vital in this field: Software Considerations in Airborne Systems and Equipment Certification (DO-178B) for software [7] and Design Assurance Guidance for Airborne Electronic Hardware (DO-254) for hardware [8]. DO-178B and DO-254 prescribe the design assurance guidance for airborne software and hardware, respectively. Although the implementation of DO-178B and DO-254 is not a mandatory regulation, DO-178B and DO-254 have been widely accepted in civil aviation [5].

However, it has been reported that DO-178B and DO-254 have several inadequate and ambiguous standards, such as ambiguity about the concept of low-level requirements and inconsistent terminologies [2]. According to applicants [4], the relative ambiguity and flexibility of these guidelines cause significantly different interpretations and implementations. Additionally, compliance with DO-178B and DO-254 guidelines in software development and hardware design is often considered as contributing to the high expense of commercial aviation systems because the guidelines require rigorously iterative processes and extensive documentation. However, the high cost may often be due to a lack of sufficient understanding of how to efficiently implement these standards [9].

Although there have been a number of studies exploring software and hardware guidelines [10–12], there have been few studies reviewing the similarities and differences between DO-178B and DO-254. It is important to understand DO-178B and DO-254 and how their processes differ to avoid potentially unnecessary work; this understanding will minimize the expense of development while ultimately ensuring the integral safety of the avionics at the system level. Therefore, the goal of this study is to present a comprehensive comparison study of various aspects of the software and hardware guidelines.

This paper is organized as follows. Section 2 outlines general software and hardware characteristics. Sections 3 and 4 present brief general overviews of DO-178B and DO-254, respectively. Section 5 discusses the specific similarities between DO-178B and DO-254, and Section 6 provides a detailed discussion of the differences. Finally, our conclusions are summarized in Section 7.

2. Software and hardware characteristics

Although there are some similarities between software development and hardware manufacture, they are basically different [13]. In essence, software is a logical system, whereas hardware is a physical system. Hardware and software are interconnected and require each other, and neither can be realistically used without the other. In particular, it is the norm in modern avionics that the desired end functionality is implemented using hardware (a microprocessor) running complex computer software. The reliability of software is a difficult issue, and there are even issues in defining software reliability (e.g., failure rate) quantitatively [3]. Moreover, it is impossible to test even the simplest software completely because the number of possible inputs, outputs, and paths through software is extremely large. In contrast, although there are several exceptions, in most cases, hardware reliability can be measured quantitatively by means of statistical and operational testing [14].

Fig. 1 shows failure rates as a function of time for software and hardware. Both software and hardware exhibit relatively high failure rates at the beginning of their service; defects are found and corrected, and the failure rate drops to a steady-state level for some period of time. Over time, however, the failure rate of hardware rises as hardware components are adversely affected by environmental phenomena, such as the cumulative effects of dust, vibration, abuse, temperature extremes or electromagnetic fields. In contrast, software is not susceptible to these problems because software does not physically fail as hardware does [15]. In practice, however, software will undergo changes

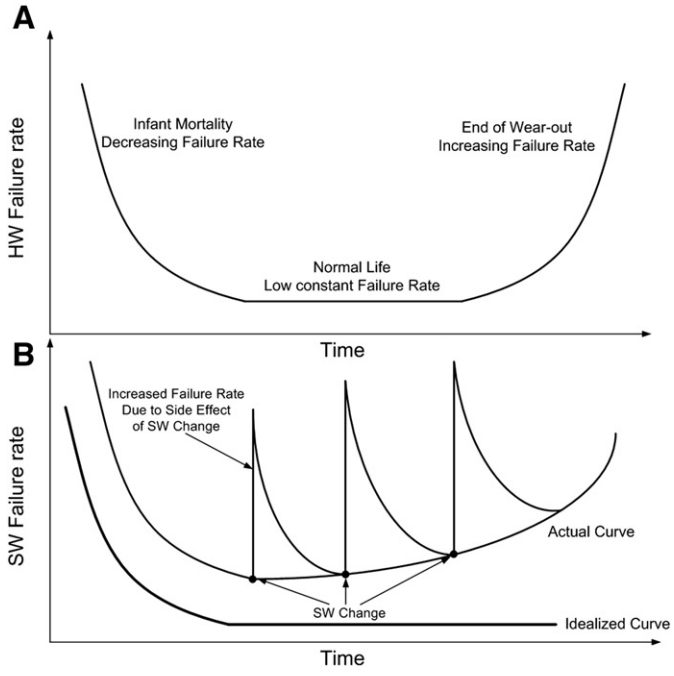

Fig. 1. Software and hardware failure rates.