Reliability Engineering and System Safety 98 (2012) 7–23

Contents lists available at SciVerse ScienceDirect

Reliability Engineering and System Safety

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

Safety certification of airborne software: An empirical study

Ian Dodd [a,1], Ibrahim Habli [b,*]

[a] Airservices Australia, Building 101 Da Vinci Business Park, Locked Bag 747 Eagle Farm, QLD 4009, Australia
[b] Department of Computer Science, University of York, York YO10 5GH, United Kingdom

ARTICLE INFO

Article history:
Received 8 February 2011
Received in revised form
11 August 2011
Accepted 24 September 2011
Available online 1 October 2011

Keywords:
Software safety
Certification
Airborne software
DO178B
Safety standards
Safety requirements

ABSTRACT

Many safety-critical aircraft functions are software-enabled. Airborne software must be audited and approved by the aerospace certification authorities prior to deployment. The auditing process is time-consuming, and its outcome is unpredictable, due to the criticality and complex nature of airborne software. To ensure that the engineering of airborne software is systematically regulated and is auditable, certification authorities mandate compliance with safety standards that detail industrial best practice. This paper reviews existing practices in software safety certification. It also explores how software safety audits are performed in the civil aerospace domain. The paper then proposes a statistical method for supporting software safety audits by collecting and analysing data about the software throughout its lifecycle. This method is then empirically evaluated through an industrial case study based on data collected from 9 aerospace projects covering 58 software releases. The results of this case study show that our proposed method can help the certification authorities and the software and safety engineers to gain confidence in the certification readiness of airborne software and predict the likely outcome of the audits. The results also highlight some confidentiality issues concerning the management and retention of sensitive data generated from safety-critical projects.

1. Introduction

Commercial airlines provide one of the safest forms of public transportation [1]. This has partly been achieved by placing high safety-integrity targets on all aspects of the industry from aircraft design and maintenance to crew training and aircraft operation. To ensure that aircraft systems are designed and manufactured to the required targets, different countries have commissioned various organisations that are responsible for auditing these critical systems. Flight approval or certification authorities include the European Aviation Safety Agency (EASA) in Europe [2] and the Federal Aviation Administration (FAA) in the USA [3]. When aircraft systems are first developed or upgraded, it is the responsibility of the flight certification authorities to approve the system design before it is cleared for flight. This process is known as Type Certification, where the authorities approve one sample of the proposed system type for flight use. Any exact copy of that type is also approved for flight use as long as it meets predefined design and operational constraints.

In modern avionics, it is the norm that the functionality is implemented using a microprocessor running complex computer software. In many cases, the avionics is a safety-critical item and therefore must be designed and built to the highest levels of safety integrity. The overall safety integrity of the avionics, comprising both software and hardware, is typically specified quantitatively, e.g. in terms of failure rates. However, for software, it is widely accepted that there is a limit on what can be quantitatively demonstrated [4,5], e.g. by means of statistical testing and operational experience. To address this limitation, many aerospace software standards appeal instead to the quality of the development process to assure the dependability of the software. In the civil aerospace domain, DO178B (Software Considerations in Airborne Systems and Equipment Certification) is the primary guidance for the approval of airborne software [6].

Throughout the software process, the certification authorities are required to audit the development, verification and support activities. The audits are known as Stage of Involvement (SOI) audits. Each audit is positioned at strategic points in the lifecycle to reduce the risk of failing the final certification audit. An early indication of a potential certification failure is vital to ensure that the software process is not heading in the wrong direction. An audit failure will normally require that an artefact must be reworked before the audit can be repeated. The typical time

* Corresponding author.
E-mail addresses: Ian.Dodd@AirservicesAustralia.com (I. Dodd), Ibrahim.Habli@cs.york.ac.uk (I. Habli).

[1] Disclaimer: The research in this paper was completed before the author joined Airservices Australia and therefore does not necessarily represent the views of his current employer. None of the data used for illustration is related to any product or activity provided by Airservices Australia.

0951-8320/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.ress.2011.09.007

duration between audits is four to six months. It is therefore important for the software and safety engineers to have an indication of how well the software process is adhering to the certification requirements before a SOI audit is performed. In aerospace software projects, it is common practice to collect metrics about the defects found during the lifecycle of the project and relate these defects to a common denominator (e.g. the number of defects found per lines of code). By relating these defect metrics to the requirements of the certification authorities, indicators can be generated to determine the readiness of the software for its next SOI audit.

In this paper, we identify a set of issues concerning the auditing process for the approval and certification of airborne software. These issues were generated from interviews with experienced independent software auditors. We then propose, and empirically evaluate, a statistical method for supporting software certification audits based collecting and analysing data about the software throughout its lifecycle. This collected data is first normalised and then weighted against certification factors such as the number and types of defects, which relate to system safety. The evaluation of our proposed method is based on an industrial case study covering data collected from 9 aerospace projects and comprising 58 software releases. In this work, we focus on two groups of stakeholders, namely certification authority auditors and development teams. Auditors could use the trend of the data over the history of a project lifecycle to identify software problems and possibly misleading information. The data could also used by the development teams within aerospace companies to assess the readiness of a software project against the certification targets. As part of our evaluation, we present the advantages and limitations of our approach from the viewpoint of both the developers and auditors.

This paper is organised as follows. Section 2 reviews existing approaches to software certification and related work. Section 3 describes a set of auditing issues concerning the software certification process. Section 4 proposes a statistical method for addressing many of the auditing issues listed in Section 3 based on the concept of Statistical Process Monitoring (SPM). This method is empirically evaluated in Section 5 through an industrial case study based on a set of data collected from anonymous aerospace manufacturers responsible for the development of safety-critical airborne software. A detailed discussion of the case study is provided in Sections 6 and 7. The legal and ethical issues concerning our proposed method are discussed in Section 8 followed by conclusions in Section 9.

2. Background and related work

2.1. Software safety certification

Certification refers to the "*process of assuring that a product or process has certain stated properties, which are then recorded in a certificate*" [7]. Assurance can be defined as justified confidence in a property of interest [8]. Whereas the concept of safety and assurance cases [9–11] is heavily used in goal-based standards in critical domains such as defence [12,13], rail [14] and oil and gas [15], compliance with prescriptive standards tend to be the norm in the civil aerospace domain [16–18], particularly with regard to the approval and certification of airborne software [6,19]. In prescriptive certification, developers show that a software system is acceptably safe by appealing to the satisfaction of a set of process objectives that the safety standards require for compliance. The means for satisfying these objectives are often tightly defined within the prescriptive standards, leaving little room for developers to apply alternatives means for compliance, which might better suit their software products and processes. One fundamental limitation of prescriptive software standards lies in the observation that good tools, techniques and methods do not necessarily lead to the achievement of a specific level of integrity. The correlation between the prescribed techniques and the failure rate of the system is infeasible to justify [16,20]. In goal-based certification, on the other hand, standards require the submission of an argument, which communicates how evidence, generated from testing, analysis and review, satisfies claims concerning the safety of the software functions. Despite the advantages of explicit safety arguments and evidence, there are some concerns regarding the adequacy of the guidance available for the creation of assurance arguments, which comply with the goals set within these standards (i.e. lack of sufficient worked examples of arguments or sample means for generating evidence). Many studies have considered and compared these two approaches to software safety assurance [21–23], highlighting advantages and limitations of each and how they might complement each other [24].

2.2. DO178B

In the civil aerospace domain, DO178B is the primary guidance for the approval of airborne software. The purpose of the DO178B document is "*to provide guidelines for the production of software for airborne systems and equipment that performs its intended function with a level of confidence in safety that complies with airworthiness requirements*" [6]. DO178B defines a consensus of the aerospace community concerning the approval of airborne software. To obtain certification credit, developers submit lifecycle plans and data that show that the production of the software has been performed as specified by the DO178B guidance. The DO178B guidance distinguishes between different levels of assurance based on the safety criticality of the software, i.e. how software components may contribute to system hazards. The safety criticality of software is determined at the system level during the system safety assessment process based on the failure conditions associated with software components. These safety conditions are grouped into five categories: '*Catastrophic*', '*Hazardous/Severe-Major*', '*Major*', '*Minor*' and '*No Effect*' [25,26]. The DO178B guidance then defines five different assurance levels, which relate to the above categorisation of failure conditions (Levels A to E, where Level A is the highest and therefore requires the most rigorous processes). Each level of software assurance is associated with a set of objectives, mostly related to the underlying lifecycle process, e.g. planning, development and verification activities (Fig. 1). For example, to achieve software level 'C', where faulty software behaviour may contribute to a major failure condition, 57 objectives have to be satisfied. On the other hand, to achieve software level 'A', where faulty software behaviour may contribute to a catastrophic failure condition, nine additional objectives have to be satisfied—some objectives achieved with independence [27].

To demonstrate compliance with DO178B, applicants are required to submit the following lifecycle data to the certification authorities:

- Plan for Software Aspects of Certification (PSAC)
- Software Configuration Index
- Software Accomplishment Summary (SAS)

They should also make all software lifecycle data, e.g. related to development, verification and planning, available for review by the certification authorities. In particular, the SAS should provide