Formal Certification for Embedded Aerospace Software

Rance Cleaveland
Professor of Computer Science and
Executive and Scientific Director, Fraunhofer USA Center for Exp. Software Engineering
University of Maryland
College Park, MD 20742
rance@cs.umd.edu

1. Introduction

Certification regimes, such as RTCA DO-178B, for airborne aerospace control software share an apparent motivation with formal-methods research in computer software: both are concerned with establishing that software systems are “fit for their purpose” and behave in accordance with their specifications. However, DO-178B and related certification guidelines contain virtually no mention of formalized modeling or proofs of correctness, even though certification can account for 30% or more of aerospace software-development costs and thus would appear to be a candidate for novel approaches to V&V.

The thesis of this white paper is that formal-methods research has paid too little attention to the production of *verification artifacts* that can serve as independent evidence of verification efforts, and that judicious research in this area would open up new vistas for improved, more cost-effective approaches to certification of aerospace software.

2. Current State of Certification

The base standard for aerospace software certification, DO-178B, categorizes airborne software according to five different “criticality levels”, and requires different testing based on the level. The most critical level, Level A, must be tested to 100% of Modified Condition / Decision coverage (a white-box coverage criterion for test data); the least critical, Level E, does not require any level of source code coverage in testing. The certification guidelines also stipulate that both high-level and low-level requirements be given for software, that these requirement sets be consistent, and that test data for source code both cover low-level requirements and be traceable to specific low-level requirements. Other activities for software V&V are also mandated, although these are not addressed here.

A key component of so-called “certification packages” for airborne software is the actual test data developed by aerospace firms during their certification testing. The test data serve two purposes.

1. They constitute replicable evidence that the software testing claimed by the company was indeed performed, and to the required level of coverage.

2. They may be used for forensic purposes, to determine in cases of system or software failure which scenarios may have been omitted during certification testing.

Although formal proofs of correctness are (rightly) touted as providing superior guarantees to those of testing, DO-178B makes no mention of formal methods, except in an Annex given over to techniques deemed not yet mature enough for certification purposes.

3. Research Agenda

The high cost of airborne software certification is due, on the one hand, to the careful scrutiny that such software should be subjected to, and on the other hand to the lack of automation that DO-178B-inspired guidelines enjoy. Formalized, mathematical approaches to specification and verification offer some promise for automation and thus cost reduction; the following research directions would help provide the groundwork for implementing this vision.

- **Proof artifacts.** As mentioned above, certification regimes require evidence of V&V. Investigation of automated techniques for constructing, storing and replaying proofs would enable formal methods to provide such archive-able evidence of verification.

- **Formalization of traceability.** For cases when formal proof cannot be conducted, testing will still be needed. Current certification guidelines refer to "traceability" of test cases to requirements, and yet this notion is not made precise and thus requires human adjudication. Formalizing this notion so that it can be machine-checked would benefit the certification process.

- **Formalization of requirements.** Another imprecision in the current guidelines concerns requirements: while they are required to exist, how they should be formulated is left open to interpretation. Defining appropriate formalisms for capturing requirements precisely, relating high-level requirement to low-level ones, and checking that requirements are provably satisfied by software, would provide a basis for automated support of these aspects of certification.

Research into formalizing other aspects of system development, including architectural descriptions and the relationship of coverage testing to correctness, could also yield substantial benefits to the certification process.