The Challenges of Software Certification

George Romanski
Verocel, Inc.

The safety critical community – those involved in developing and verifying safety critical systems – is very conservative and adverse to change. Meanwhile, technology is changing rapidly, and there is pressure to adapt systems to improve their efficiency and safety. This presents a number of challenges. The community has already addressed some; others are in process. While the guidance on airborne software certification is mature, the issues with software reuse, military avionics certification, ground-based software, and object oriented technology are still evolving.

Computer-controlled systems pervade our lives. We take many of them for granted, although many are critical to our safety. Dire consequences may result if the software in an automobile's computer-controlled braking system failed during a highway maneuver. Software faults in the automobile's computer may cause the brakes to malfunction. What can be done to make sure that this software does not contribute to the cause of an accident?

An extreme measure would be to limit the speed of automobiles to a walking pace. When automobiles were first licensed to travel on public highways, a person carrying a red flag had to walk in front of them. Clearly this restriction is impractical, and was abandoned many decades ago. An alternative is not setting up the computer system as the sole means of control. A computer-controlled braking system could include hydraulic/mechanical controls that provide backup operations when the computer system fails. However, this approach may be impractical for some complex systems.

A bicycle can turn corners much faster than a tricycle, but it requires active control by the cyclist to maintain balance. Similarly a high performance airplane may be built with an unstable flight profile. For example, a paper plane with wings that are level may swerve and dive during flight, but a plane whose wings tilt up slightly glides smoothly. Fighter planes are often built unstable to make them more agile but they include movable surfaces. A computer system controls the movable surfaces and induces stability through control algorithms.

A modern transport airplane may be built to have a stable flight profile, but computers are used to control and optimize aircraft flight. These types of computer systems and their software have a direct impact on the safety of the aircraft and its occupants. A level of assurance is required to provide confidence in this software. In commercial avionics systems a document called "Software Considerations in Aircraft Systems and Equipment Certification" is used to provide assurance. In the United States, this document is called DO-178B and is published by Requirements and Technical Concepts for Aviation, Inc. (RTCA) [1].

> "Although DO-178B is referred to as a guidance document, it is treated as a standard that imposes requirements on the development and verification of airborne avionics systems."

DO–178B

Although DO-178B is referred to as a guidance document, it is treated as a standard that imposes requirements on the development and verification of airborne avionics systems. A Federal Aviation Regulation lists DO-178B as a means of compliance that is acceptable to the software regulators in the avionics community. In the United States, the regulatory body is the Federal Aviation Authority (FAA). While DO-178B is not the only means of compliance, compliance with the objectives of DO-178B must be shown if a different approach is used.

In Europe, a similar statutory regulation called ED-12B is published by EUROCAE. This is the same document as the DO-178B and was produced by an international consensus-based committee representing practitioners as well as regulators.

DO-178B is intended to describe the objectives of the software life-cycle processes, the process activities, and the evidence of compliance required at different software levels. The software levels are chosen by determining the severity of the failure conditions, which may affect the aircraft and its occupants [2]. The failure conditions are named and have corresponding levels identified by letters. For each level there is a set of process objectives that must be satisfied. An example of a process objective is A-7.3 "Test Coverage of low-level requirements is achieved." The number of process objectives by software level is shown in Table 1 (see page 16).

The certification agencies have received many requests to clarify the intent of DO-178B since its publication in December 1992. A consensus-based committee called SC-190 (and WG-52 in Europe) was formed and tasked to propose clarifying text. After four years of work by 150 registered members, a document called The Annual Report for Clarification of DO-178B (DO-248A) was published by RTCA, Inc. [3]. This document provides corrections (typographical and editorial), answers to frequently asked questions (FAQs), and discussion papers.

A subset of the SC-190 committee is continuing with a new document for use in Communications, Navigation and Surveillance/Air Traffic Management systems (CNS/ATM). This will result in a new software assurance document intended for ground-based (and space-borne) systems. There are subtle differences between the way the two application domains are treated. Airborne systems undergo a certification process while ground-based systems go through an approval process. In the following text certification materials describe materials that support certification or approval. Here are some typical FAQs published in DO-248A (shortened for brevity):

Q: Is recursion permitted in airborne applications?

Failure Condition	DO-178B Software Level	Process Objectives
Catastrophic	A	66
Hazardous	B	65
Major	C	57
Minor	D	28
No Effect	E	0

Table 1: *Relationship Between Failure Conditions, Levels, and Objectives*

A: Yes! But it must be bounded.
Q: Is source-code to object-code traceability required?
A: Yes, if the applicant is providing coverage analysis at the source-code level and the assurance is at level A. No, if coverage analysis is provided at the machine-code level.
Q: If some run-time functions are inlined, is coverage still required of the run-time functions?
A: Yes! Coverage analysis is required of all of the code that may be reached within the address space.
Q: Can compiler features be used to simplify coverage analysis at object code?
A: Yes! For example, short circuit conditions may be used. However, as the compiler feature is being used as a verification tool, this feature of the compiler must be qualified as a verification tool. (Qualification is the process of assuring that a tool can be used in place of a verification activity performed manually.)

One of the most discussed topics is the use of previously developed software (PDS). Commercial off-the-shelf (COTS) software is considered PDS as it may be developed independently of any specific airborne application. Operating systems (OS) may be considered to be COTS software and may have been developed using in-house development processes that are not necessarily compliant with DO-178B requirements. These pose a burden on the user of the OS to reengineer the verification evidence required unless it is made available by the OS developer.

DO-178B makes provisions for reengineering requirements, design information, tests, and review of all artifacts in accordance with the DO-178B objectives. The processes must be documented in a set of plans and evidence must be available to show that this was performed in a controlled way. DO-178B provides an alternative means mechanism allowing developers to present evidence that is not typical. The developer has the burden of proof that the materials presented are acceptable alternatives to a risk adverse audience.

Operating System Issues

An OS takes control of the target machine on which it runs. It shares resources between processing threads. The threads may be represented as processes, tasks, or co-resident applications, depending on the nature of the OS. The shared resources include processor time, interrupts, memory, and input/output transactions to name a few. The OS has visibility and control over application programs and would have a direct impact on system behavior if the OS were to malfunction. Because of this close connection between the OS and the applications, the OS must have certification evidence at least to the same software level as the systems that it supports. This means that all certification criteria that apply to the applications also apply to the OS. In particular, there must be a level of confidence that the OS itself behaves deterministically, and that the underlying applications will be controlled in a deterministic way.

The level of determinism may be based on functionality, resources, and time. Functional determinism can be demonstrated through testing only if the results of a function are the inevitable consequence of its inputs. The function inputs are the parameters, but may also include global variables, and possibly external states based on interrupts. Clearly the more variables there are, the more difficult it is to demonstrate functional determinism through testing alone.

Use of resources must be bounded, otherwise their consumption will grow unchecked. This includes the use of dynamically allocated memory, like the heap, but also includes a bound on stack space. The OS used in safety critical systems may prevent use of dynamic memory allocation or may restrict this after the system completes its initialization. If the OS does not offer such precautions, then the application developers must be very cautious to ensure that memory creep does not cause the system to malfunction at some point in the future. Application code as well as the OS affect stack growth. An operating system will allocate space for the applications and may allot some of this space for control data structures such as task control blocks. After this allotment, the operating system functions typically consume little of the stack, and release it as soon as the call is finished. It is up to the application programmer to estimate the worst-case stack usage space. Users typically allow large margins to the space allocated to ensure that the resource is not entirely consumed.

Time is a difficult resource to measure and allocate. In an effort to improve processor throughput, hardware engineers have added many improvements to modern computing devices. Cache memory, pipeline processing, and co-processors may improve performance tremendously, but they also make it very difficult to put an absolute bound on the worst-case execution time for a particular function. Nevertheless, the performance improvement through the use of cache memory can be so dramatic that in many situations it becomes the overriding concern. The performance of an OS function may be dependent on the contents of instruction and data cache memory.

This varies depending on the execution paths taken through the applications as they run. In practice, timing measures of OS functions add little to the question of deterministic behavior. Typically, the application execution time is measured under estimated worst-case conditions to determine if the time bounds can be met. Measures of tasking performance under these worst-case conditions can be used to calculate the total throughput, and then determine whether deadlines of the tasks cooperating in the application can be met. This leaves the burden of timing measurement and verification to the application developer.

The COTS Reuse Argument

Software may have been developed and in service for a long time with no problems. If we have a record of the behavior, then intuitively it seems that we should be able to trust this software. Use-of-service history as a means of obtaining approval credit is an attractive option to help reduce certification costs. The notion is that if an application is used and its service history is recorded (frequency, severity, and distribution of faults found) then by extrapolation, similar behavior would be expected in applications running the same software in equivalent environments.

Low fault rates in the past may mean expected low fault rates in the future. This reasoning is particularly attractive for com-

panies that developed systems for military airplanes who want to reuse the same technology for commercial aircraft. However, this approach should be used with caution. At the higher software assurance levels, requirements for coverage analysis are intended to provide a measure of the absence of unintended functionality. By showing what code is executed in the application and what is not executed, the shortcomings of the requirements-based testing process can be estimated. If any code is discovered that cannot be traced to a requirement, then this dead code must be removed. Use-of-service history will not show presence of unintended functionality, so it does not satisfy the coverage analysis objective.

Coverage analysis is also used to show the effectiveness of testing. The type of coverage analysis required by DO-178B depends on the software level. At level C, only statement coverage is required. At level B, all entry points and exit points as well as all decisions and their outcomes must be covered in addition to all statements. At level A (the highest), coverage requirements of level B in addition to Modified Condition/Decision Coverage (MC/DC) is required.

MC/DC analysis is a unique requirement to DO-178B. Its goal is to show that each condition has its intended effect on the outcome of a decision. Applicants trying to reduce the scope and cost of the test and analysis effort have applied a number of different interpretations. To ensure that the interpretations are common, the SC-190 committee produced clarification through a discussion paper. This paper is included in the DO-248B document. Here are some valid approaches: perform coverage analysis at the machine-code level, show source-to-object code traceability together with coverage analysis at the source-code level, or develop multiple voting systems and use different languages where each compiler used is created by a different developer.

A draft policy for reusable software components (RSC) has been developed to allow certification evidence and its approval to be reused when it exists. The intent of this policy is not aimed at reusing components in different variants of a particular system being deployed. If a certified system exists, certification credit can be taken for components when moving the system to a different aircraft. This is already covered by other regulations.

A developer producing a software component and developing certification material in the absence of a specific avionic system may be creating a reusable software component. The FAA does not charge for its approval services, so it does not deal with such developers directly. (Otherwise the burden of dealing with software suppliers directly would be overwhelming.)

Airframe or subsystem manufacturers (such as the developer of an aircraft or a flight management system) may establish a certification liaison with the FAA as an applicant. As part of the delivery of materials for review, the applicant may submit DO-178B compliant materials for the RSC. This submission will also include information such as proposed software level, identification of the processor, and identification of the compiler used. Once approval of the airframe or subsystem is obtained, the FAA may provide a letter to the RSC developer and to the applicant documenting certification credit. This letter either reduces or eliminates certification effort required on a new project.

Military Avionics

To reap the benefits of a wider audience and participation by practitioners outside the Department of Defense (DoD) domain, the agency gradually moved towards standards that were co-developed with industry members. DO-178B is a consensus-based guidance document that has been adopted by the DoD for certain safety critical systems. Development of certification evidence in accordance with DO-178B is not undertaken retroactively; new projects and updates to projects do adopt this guidance document in place of a military standard. The initial draft policy focuses on transport aircraft. Fighters, bombers, and unmanned vehicles are excluded. The policy is directed at Communications Navigation and Surveillance/Air Traffic Management (CNS/ATM) systems, both airborne and ground based. The FAA owns the U.S. airspace. The U.S. Air Force is required to show that its transport planes do not degrade airspace safety during peacetime.

It could be argued (in jest) that military pilots have parachutes and cannot sue the government if an airplane malfunctions, so the software levels for the systems can be lowered. In practice, the software quality is taken seriously, but not all of the objectives of DO-178B are applicable in the DoD setting. The regulations, policies, and procedures within the FAA have evolved to encompass the DO-178B document. Certification liaison procedures are part of the approval process documented in DO-178B. An airborne system development and certification project is encouraged to form a relationship with an Aircraft Certification Office of the FAA very early in the life cycle of the project. Throughout the project lifetime, FAA personnel and/or designated engineering representatives (DERs) oversee all steps through the project phases. DERs are engineers who have been accepted through an approval process to act on behalf of the FAA. These engineers may provide guidance to the developers and have the authority to approve the materials developed for certification.

This certification liaison process is still to be developed within the DoD. On military projects, the contractual and approval processes and adherence to military standards have been used to measure a project. The DoD approach has been to contract with a supplier to develop a system subject to the provisions of an agreed contract. The FAA aircraft certification approach is much more open-ended. It allows applicants to spend their money seeking certification approval from the FAA.

The DO-178B guidance document lists objectives that must be satisfied, but it does not prescribe how. Through the certification liaison and DER review/approval process, the process plans should be developed and agreed upon. As long as certification materials are produced according to the documented processes, they should be acceptable during the final audits before approval.

Ground-Based Systems

The ground-based community (CNS/ATM) faces a similar challenge to the DoD as both funding and approval are bestowed from the same organization.

The Air Traffic Management systems have growth challenges. Many of the control centers use systems that are becoming obsolete, while at the same time air traffic continues to increase. The projected growth is 6 percent annually in Europe and 4 percent in the United States. This comes at a time when the capacity loading is already very high.

The promise for the future is to improve capacity and safety through the introduction of free flight. Current technology allows commercial airplanes to fly from one airport to another inside prescribed corridors at prescribed heights. This reduces the workload of air traffic controllers and allows them to focus on maintaining separation between aircraft. The free-flight system will allow an aircraft to choose its preferred climb from the departure airport, its preferred path, and its preferred descent to the arrival airport. Clearly if each aircraft were to take this

approach independently, the result would be chaotic and dangerous.

By disclosing its intended behavior, an airplane may join the set of aircraft managed by a ground-based system. There is much data to be accumulated, shared, and tracked to avoid possible conflicts. Static information must be uploaded to the plane describing the local terrain, airways, and other airport information. Dynamic information is uploaded as required throughout the flight, including weather, possible warnings, capacity constraints, and special use airspace schedules (e.g., military requirements). Given this information the pilot can produce a flight plan that results in a filed flight trajectory. This can be treated as an object, which will then be used by ground-based systems.

During flight, the pilot may wish to change the flight plans, but can only propose a change that must be approved by the ground-based system before it can be adopted. Furthermore, the actual trajectory is recorded and transmitted by the aircraft, so that the ground-based systems can track it as an object. The accumulation of this airspace data allows traffic density predictions to be calculated, and dynamic route structure objects to be produced [4].

These objects – produced, consumed and manipulated by computers – may be modeled and even implemented through some Object Oriented Technology. There are languages that support these concepts and provide a direct way of manipulating them. The implementation of the free-flight initiative has still not addressed such issues. The FAA is evaluating the problems of Object Oriented Technology.

Object Oriented Technology

There is pressure from industry to use object oriented paradigms in the development of safety critical software. The expectation is that, as in other industry sectors, such programming will lower the development costs. There is some reluctance by regulators to approve this type of programming as it introduces concepts of information hiding, polymorphism, and inheritance. This makes the coupling between code and data less obvious to an auditor. It may invoke run-time support code that creates and destroys these objects dynamically, depending on the scope of the objects during execution. The timing and resource usage of such run-time programs make the application less deterministic, complicating the analysis and approval of such systems. It is expected that ultimately some compromise will be reached and a subset of the object oriented programming paradigm will be adopted, thereby satisfying the concerns of determinism and providing the benefits of this new technology.

Conclusion

Although a number of challenges remain, the industry is very focused on safe air transportation. It is through tremendous vigilance and determinism that the industry has a good safety record. It can be improved, and these on-going initiatives will contribute to safer flight.◆

References

1. DO-178B. Software Considerations in Airborne Systems and Equipment Certification. RTCA, Dec. 1, 1992.
2. AC 25-1309-1A, Advisory Circular, Federal Aviation Administration.
3. DO-248A. Annual Report for Clarification of DO-178B. RTCA, Oct. 6, 1999. (DO-248B to be published in 2001.)
4. National Airspace System Concepts of Operations. RTCA, Dec. 13, 2000.

Resources

- For a complete listing of RTCA documents please see <www.rtca.org>.
- The FAA Flight Standards Service provides links to the regulatory Web sites at the following Web site <www.faa.gov/avr/afs/fars/far_idx.htm>.

About the Author

George Romanski has specialized in the production of software development environments for the past 30 years. Romanski was vice president of Technology at EDS/Scicon, vice president of Engineering at Alsys and director of Safety Critical Software at Aonix. Romanski also serves the safety-critical industry as a member of the HRG (Annex H Rapporteur Group) for the Ada95 ISO standard addressing safety and security issues as well as the Requirements and Technical Concepts for Aviation (RTCA)/SC-190 committee working to provide clarification of DO-178B for avionics and ground-based systems. Romanski is president of Verocel, a company specializing in the verification of software, and in the development of tools that help in this process.

Verocel, Inc.
234 Littleton Road, Suite 2A
Chelmsford, MA 01886
Phone: (978) 392-8860
E-mail: romanski@verocel.com