DOT/FAA/AR-01/26

Office of Aviation Research
Washington, D.C. 20591

Commercial Off-The-Shelf (COTS) Avionics Software Study

May 2001

Final Report

U.S. Department of Transportation
Federal Aviation Administration

NOTICE

This report is available at the Federal Aviation Administration William J. Hughes Technical Center's Full-Text Technical Reports page: actlibrary.tc.faa.gov in Adobe Acrobat portable document format (PDF).

Technical Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
DOT/FAA/AR-01/26		

4. Title and Subtitle	5. Report Date
COMMERCIAL OFF-THE-SHELF (COTS) AVIONICS SOFTWARE STUDY	May 2001
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
Jim Krodel	

9. Performing Organization Name and Address	10. Work Unit No. (TRAIS)
United Technologies Research Center 411 Silver Lane East Hartford, CT 06108	11. Contract or Grant No. DTFA03-99-C-00030

12. Sponsoring Agency Name and Address	13. Type of Report and Period Covered
U.S. Department of Transportation Federal Aviation Administration Office of Aviation Research Washington, DC 20591	Final Report
	14. Sponsoring Agency Code AIR-130

15. Supplementary Notes

The overall anticipated results for this research are summarized in this report as well as in the hardware technical report titled "Review of Pending Guidance and Industry Findings on COTS Electronics in Airborne Systems." This summary is found in the Executive Summary. The FAA William J. Hughes COTR is Charles Kilgore.

16. Abstract

A trend has developed via the economics of software development to streamline process and products of organizations, increase their efficiency, and reduce overall software development costs. System development activities associated with Radio Technical Commission for Aeronautics, Inc. (RTCA) document "Software Considerations in Airborne Systems and Equipment Certification," DO-178B, can be expensive and, therefore, are the focus of efforts to increase efficiency and reduce costs. The motivation is even a bit beyond monetary resources as the scarcity of highly trained personnel that can develop such systems has also provided fuel to the attractiveness of considering reuse of commercial off-the-shelf (COTS) software.

Software professionals have long envied the reuse model that has been established in the hardware arena. Hardware designs are easily fabricated from subassemblies and other components, although the firmware is affecting this arena also. Software designers have not been as effective in establishing their own reusability model. Nevertheless, software component reuse is still sought as a means for increasing software development productivity. More recently, the advent of object-oriented techniques has facilitated the development of such reusable code components.

This report takes a snapshot of portions of industry domains related to safety. Avionics, nuclear, medical, space, and elevator domain information is surveyed. Key industry COTS components are identified and potential alternate methods for verifying a COTS component's applicability to the avionics domain application are studied.

17. Key Words	18. Distribution Statement
COTS, Software, Electronics, Commercial off-the-shelf, DO-178B	This document is available to the public through the National Technical Information Service (NTIS) Springfield, Virginia 22161.

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	40	

TABLE OF CONTENTS

EXECUTIVE SUMMARY

The anticipated results for this research are summarized below under the headings of Software and Hardware. There are two final deliverables, one technical report on software COTS components and one on hardware COTS components.

SOFTWARE.

This report is being submitted to support the Federal Aviation Administration (FAA) software portion of the contract to United Technologies Research Center on commercial off-the-shelf software and hardware research (COTS).

Typical COTS software components are used in low-risk applications and many have been developed without considering the safety aspects of the software. However, to utilize this software in airborne systems requires that the COTS product be scrutinized to determine its ability to meet the intent of Radio Technical Commission for Aeronautics, Inc. (RTCA) document "Software Considerations in Airborne Systems and Equipment Certification," DO-178B objectives. The ability to access a COTS product's development and product verification documentation, if any was produced, is usually limited. However, some products are being developed specifically for the airborne market, albeit the selection is limited, it is growing.

Issues concerning the usage of COTS are not completely addressed in DO-178B. Due to the current relationship with the vendor the avionics application represents efforts on behalf of both parties. Thus, problem reporting COTS prior operation environment, version control, and process and product examination requires special attention. Some domains using COTS have considered a separate COTS specific process plan to analyze the safety, integration and verification aspects of the COTS components, and the overall avionics application.

Operating systems are currently receiving the most attention, and several real-time operating system vendors have commenced efforts to provide DO-178B ready COTS operating systems. A variety of hardware and software architectural techniques exist to reduce the risk of using COTS components in avionics and other domains. Furthermore, verification techniques and other alternative methods are being applied in combination on the COTS component in efforts to meet the intent of DO-178B. The more popular techniques currently being used include reverse engineering of the COTS component, software protection wrappers, partitioning, and COTS component service history. Some COTS operating system vendors are paying particular attention to how their component is being integrated into the avionics product and provide a variety of data and services to support this integration.

A detailed review of DO-178B's objectives identifies which objectives are providing resistance to COTS technology insertion. This level dependent review shows that many typical COTS components are more applicable to the software levels C and D avionics applications and that the DO-178B levels A and B objectives relating to strict structural testing and independence make COTS utilization more difficult.

HARDWARE.

This hardware executive summary is presented here in this software report at the sponsoring agency's request to brief readers on the contract work relating to hardware. However the hardware report is contained in the technical report "Review of Pending Guidance and Industry Findings on COTS Electronics in Airborne Systems," DOT/FAA/AR-01/41 which will be available by the end of calendar year 2001.

The hardware report is being submitted to support the FAA hardware portion of the contract to United Technologies Research Center on commercial off-the-shelf software and hardware research (COTS).

The intent of the hardware report is to provide findings about the state of the industry relative to the design objectives identified in guidance document entitled "Design Assurance for Airborne Electronics Hardware," DO-254, with focus on the implications for the use of COTS electronic hardware components in safety critical airborne systems. Industry findings were gathered from conference proceedings, technical and trade journals, and a number of sources available through the World Wide Web.

The use of complex electronic hardware components in airborne systems poses a challenge to the meeting of safety requirements because, for complex components, complete verification is, at best, very difficult, and at worst not achievable. In order to address the potential lack of complete verification, it is recommended that the hardware design life cycle processes should include design assurances to mitigate the possibility that design errors may be introduced into the hardware component and cause anomalous behavior.

New technologies, being developed in the commercial sector, could provide enhanced safety in airborne systems if the technologies could be incorporated at an affordable cost. However, the use of COTS components in airborne systems raises a number of issues with respect to meeting airborne system safety requirements and DO-254 objectives.

Commercial market trends are rapidly diverging from the needs of safety critical airborne systems.

Key component attributes have been identified as desirable/necessary to meet design objectives. Key component attributes include design assurance, component reliability, operation beyond manufacturer specifications, service experience, configuration management, production practices, verification testing, and role of COTS tools in design and verification. COTS components are challenged to provide many of these attributes. Therefore, the ability of COTS components to meet the DO-254 design objectives may be limited by a number of issues, which are explored in the report.

Successful application of COTS components, in airborne systems, requires extensive original equipment manufacturer (OEM) effort to develop the required assurances for the COTS component. In addition, design assistance needs to be extended to the OEM, by the COTS

supplier, which is well beyond that which is normally provided in the commercial sector. If the COTS supplier is unable or unwilling to provide necessary design information, for a certain COTS component, then the OEM effort associated with developing design assurances will be significantly increased and may preclude the use of the COTS component.

Issues with respect to COTS usage may become barriers in certain cases, if necessary assurances cannot be achieved in a cost-effective manner. The assurances required for high criticality applications, such as levels A and B, will probably not be attainable for COTS components without additional assurances by other means.

1. SCOPE AND INTENT.

This report is being submitted to support the Federal Aviation Administration (FAA) contract to United Technologies Research Center on commercial off-the-shelf (COTS) software and hardware research. This report represents one of the final deliverables on the software portion of this contract.

DO-178B defines COTS as "Commercially available applications sold by vendors through public catalog listings. COTS software is not intended to be customized or enhanced. Contract-negotiated software developed for a specific application is not COTS software" [1]. Since this definition does not provide an unambiguous distinction of what COTS is, this report offers another definition.

COTS, for the scope of this study, is meant to be any software product that is not developed within a given company for a specific application for that company. In particular, information regarding the software product's development and fabrication is not known or not available to the user of the COTS product. For example, if a product was developed so that it could be used on other systems from a different company, it will be considered COTS for the purpose of this study. However, a product that would be used by the same company on a large number of its own internal projects would not be considered to be COTS.

This portion of the contract is reporting on the culmination of a year long study on the state of the COTS software industry and on possible alternate methods of verifying COTS products lacking a priori knowledge of the COTS product development.

There is an enormous wealth of information with regards to COTS software. The approach taken for this study was to obtain data from deployed or soon to be deployed safety critical COTS components. The focus included interviewing domain experts for this information. Some of the information obtained is proprietary and will remain so; however, trends and general observations have been made and are stated in the body of this report. Appendix A lists survey questions posed to domain experts for the purposes of this report.

The study conducted for this report collected a variety of sources such as periodicals, conference reports, web pages, and phone interviews. Library searches were conducted electronically and automatic topic notification was used for 18 separate journals. These journals are listed in appendix B.

This study provides a small snapshot of current activity in the COTS domain. A full in-depth assessment of the COTS domain is beyond the scope and resources of this contract. However, data collection and research for this contract was focused on obtaining data for use to assist in the development of regulatory guidance on COTS software in avionics systems. This report is not to be construed as guidance material for aviation or any other regulatory authorities.

2. INTRODUCTION.

In order to provide a systematic basis for evaluation of COTS, criteria for categorizing a COTS product should be established. COTS may contain many dimensions. Two important dimensions are listed below along with their value to this study:

- Wide Usage—This dimension measures the number of different application environments and users of the product. This provides the potential for comparison and possible approval on the basis of in-service operation.

- Level of assurance—This dimension establishes the source (governmental, industry, third party) of assurance and the standard for that assurance.

Using the above criteria, COTS that is to be included as part of this study can be identified, but not limited to, any product that has application to at least one or more nonaviation application environments.

3. BACKGROUND.

3.1 CURRENT STATE OF GUIDANCE FOR DEVELOPMENT OF AVIONICS CONTAINING SOFTWARE.

In 1992, Special Committee 167 under the auspices of Radio Technology Commission for Aeronautics, Inc. (RTCA) published a document identified as DO-178B entitled "Software Considerations in Airborne Systems and Equipment Certification." In January 1993, the FAA distributed Advisory Circular 20-115B recognizing DO-178B as the preferred guideline document to be used for software development activities for embedded software on avionics systems.

This report will not go into the details of DO-178B's guidelines. Suffice it to say, that DO-178B provides a set of processes and objectives that must be met when developing airborne safety critical software. The application of these processes and objective are necessary in order to apply for certification of avionics components containing software.

DO-178B contains little useful guidance with respect to the use of COTS software and, at this point in time, COTS software is treated much like originally developed software. However, any applicant (entity applying for certification of an embedded software avionics system) can submit to the FAA, via their plan for software aspects of certification, the basis for which they plan to implement and integrate COTS software.

The FAA issued a notice on March 26, 1999, identified as N8110.82, "Guidance for Applying the RTCA/DO-178B Level D Criteria to Previous Developed Software (PDS)." In brief, the notice states that for criticality level D software, previously developed software (PDS) does not need to meet DO-178B objectives A-2, 4, 5, and 6, as they are inherently met. These objectives mainly support the traceability guidance in DO-178B. The published notices can be viewed at the web site http://av-info.faa.gov/software/related.htm.

For applicants who seek certification of airborne systems with software from the FAA, the development process may include designs to avoid faults in the system, designing the system to tolerate faults without catastrophic consequences, or providing additional measures to mitigate the consequences of system failure. Developers of such systems focus on achieving safety goals, yet extensive resources could possibly be expended in achieving these goals.

3.2 SYSTEM LEVEL ASSESSMENT FOR COTS.

The initial task with any airborne system containing software is to conduct a system safety assessment that will identify potential critical failures. The identification and allocation of these safety critical requirements to software become the basis for perhaps not only the software architecture but also perhaps the hardware architecture. As such, any COTS products planned for the system should either play a role or be included in a design that would detect, prevent, or mitigate failures.

Once the environment for the COTS product has been established in terms of the safety functions to be performed, it is necessary to evaluate how it will interface with other system components in ways that will not lead to unsafe conditions. The attributes to consider in this evaluation are correctness, performance, reliability, and the product's erroneous interaction, with respect to the rest of the system. The interactions to consider include abnormal behavior under stress conditions and the immunization of legitimate features of the COTS product that are not needed for the safety-related application [2].

3.3 THE PREDICAMENT FACING THE FAA.

The pressures to employ COTS software in airborne systems have created a new set of difficult questions for regulators. Aside from technical questions related to the safe integration of a COTS software product into the system, frequent difficulties are also encountered due to various competitive or management concerns. These concerns tend to interfere with access to records and information needed for product assessment. If the desired records and information are unavailable, the regulator (and developer) is then faced with either rejecting the COTS product for use in the system or must make a determination about the acceptability of the COTS software product. The basis for this acceptance criterion for the COTS product is different than those used for new developments, because of the lack of evidence in compliance to DO-178B's objectives.

Alternative methods of compliance is not only difficult, but relies on the regulator's ability to understand the alternate method and apply some amount of subjective evaluation that the methods meet the intent of DO-178B objectives or provides equivalent levels of confidence. It should be shown convincingly that alternative acceptance criteria provide confidence equivalent to that obtained from the processes applied to new developments since, if it is possible to achieve this confidence with less costly new alternatives, the existing processes used in new developments will quickly be abandoned.

One technique suggested is to apply a gradient in evaluating COTS products [2]. The gradient would evaluate the COTS product's impact associated with potential failure, and with respect to the relative importance to safety of constituent components. The issue facing the FAA is that

even with a gradient evaluation, the task is nonetheless subjective and consistency in the assessments would be problematical.

4. ISSUES CONCERNING USAGE OF COTS WITH DO-178B.

4.1 THE BUSINESS RELATIONSHIP BETWEEN THE VENDOR AND APPLICANT.

Under consideration is the fact that once a software component has been integrated into a safety critical application, its responsibility for performance and reliability is assumed, at least in part, by the applicant. As such, one view that could be considered by the applicant would be to govern their relationship with the vendor, such as it would from a custom developed avionics application. With this view, all the processes in the custom developed avionics application need to be correspondingly evaluated for compliance to the intent of DO-178B. Should this correspondence not align, alternate methods of compliance should be considered.

4.2 PROBLEM REPORTS.

For in-house-developed systems, problem reports from the deployed system require analysis for their effect on the overall system safety. Systems incorporating COTS exposes the safety of the system further, and problem reports from the COTS vendor should be coordinated to allow visibility into problems that may have been previously masked. These reports should be studied to determine their safety impact to the system, if any.

An established problem reporting relationship should not only be established from the vendor to the applicant but also from the applicant to the vendor. If a COTS product has been evaluated sufficiently to allow it to be used on one safety-related application, then it has a high probability of being used on other safety-related systems. As such, COTS-related errors discovered in the avionics application should be reported to the vendor for the error's potential affect on other deployed systems.

Additionally, errors from other uses of the COTS component need to be monitored albeit from different domains. However, this may be difficult for many COTS products due to either a lack of problem report tracking rigor by the vendor or reluctance of users to adequately report problems found in the field.

4.3 UNUSED OR UNINTENDED FUNCTIONS.

Functions intended by the developer but not needed for the avionics application is one source of concern. Alternately, unplanned functions arising from the COTS design or implementation errors are another source of concern for COTS products. Neither of these is desirable for critical airborne systems. Unused or unintended functions might affect the performance of system safety functions if the functions are inadvertently activated. DO-178B provides guidance with regards to both dead code, that is code not reachable due to design error, and deactivated code, that is code which is part of the avionics application, but not enabled for operation.

4.4 COMMERCIAL OFF-THE-SHELF PREVIOUS ENVIRONMENT AND OPERATIONAL PROFILE.

Proper evaluation of the COTS software requires a thorough understanding of the operational profile of the software's role in the system. Inputs, outputs, performance requirements, and reliability requirements need to be understood. For COTS products, the operational profile assumed in its design might not be known precisely. Consequently, it is difficult to assess where discrepancies between the design and the specification for COTS usage in a safety critical system might appear.

4.5 VERSION CONTROL.

All of the information pertaining to the COTS item to be used should be clearly linked to the specific version to be incorporated into the system. Different versions require separate evaluations. It is also important to understand that a COTS product could be so highly configurable that version information may require version control of individual subcomponents of the COTS configured system.

4.6 NEW RELEASES.

New releases, as noted above, need to be evaluated on their own merits. However, if an applicant desires to upgrade a system with a new release of the COTS product, this may be an opportunity for the certification authority or applicant to obtain additional information on the product's development that would further substantiate the compliance to DO-178B's intended objectives.

4.7 PRODUCT EXAMINATION.

Product development records and operating experience data are areas that need scrutiny. Product development records include specifications, code, testing and verification records, software quality assurance records, etc. Operating experience data can provide information about reliability, correctness, and performance; however, care should be taken to ensure that the data is valid. Data based on other versions or different configurations may not be applicable.

4.8 PROCESS EXAMINATION.

Process examination helps to establish the validity and value of the product development records. Product information is not very useful without confidence that the processes that generated the information are comparable to commonly accepted practices for developing high-integrity software. Various methods have been developed to assess these processes, including International Organization for Standardization (ISO) and Software Engineering Institute Capability Maturity Model (SEI CMM) style assessments and other assessment processes that may have special focuses. It is understood that these processes do not meet the intent of DO-178B, yet portions of these assessments may be used to support some DO-178B objectives. As previously indicated however, either the data may not be available for an assessment or the COTS vendor may not cooperate with an assessment.

5. SURVEY INFORMATION.

The survey conducted was designed to reach both avionics and other industry domains that use embedded software in safety critical applications, e.g., avionics (levels A, C, D), nuclear, medical, space, and elevators. Much of the information was obtained from phone interviews from representatives of that industry.

Survey results showed that the underlying COTS components most often used in safety-critical applications appear to enable a system's capability; like operating systems and network software. Most of these components positioned the application to be open to additional capabilities for future growth.

5.1 AVIONICS.

5.1.1 Large Airframe Manufacturer.

A manufacturer of a large airframe reported that they decided not to use COTS software on their in-house-developed product. All airframe software developed in-house is fully under DO-178B and, as such, they have decided to custom-develop their software for the airframe. This is due in large part to the certification requirements for their level A aircraft and its related components. The manufacturer did report that there are some line replaceable units (LRU) of criticality levels C and D that are planned for their airframe that use COTS products. A vendor manufactures one LRU and the product uses a COTS operating system. Attempts to get information regarding this vendor's specific avionics application were not successful. The other LRU is described below.

5.1.2 Large Airframe LRU Manufacturer No. 1.

An avionics company has planned an LRU that uses the Enea's OSE real-time operating system (RTOS). More information regarding this and other operating systems is detailed below. This company selected OSE based upon claims that OSE has been used or plans to be used in a level B certified system. The avionics company further indicated that Enea is currently working with another avionics developer with intentions of upgrading OSE to permit real-time operating system partitioned space that would be DO-178B compliant. The intent is that custom avionics applications could be assigned to these protected partitions. Although, initially, the approach showed promise, the applicant has subsequently decided to abandon this vendor and has selected another RTOS to base their avionics application on.

5.1.3 Large Airframe LRU Manufacturer No. 2.

Another avionics company on a level D application is using an operating system called RTXC from Embedded Power Corporation. The applicant used only select functions of the operating system and conducted testing based on the operating system's user manual. The operating system was tested within the envelope of the operating system domain, but was not stress tested. No life cycle data was made available other than the user's manual and object code. The applicant plans on using this baseline version and is not considering any upgrades to the operating system. The product did have some stated service history whereby the COTS vendor

indicated it was used in the medical, nuclear, and one or two avionics applications. The vendor was in no way involved with the certification of the avionics application. The product is designed to permit user selection of functional, loadable object code, and the applicant took advantage of this feature. This COTS product did affect the software architecture in that the RTOS directed the assignment of task priorities and scheduling. Two tasks having the same priority could possibly run in the same time slice, when concerns of whether this was truly deterministic arose, the applicant decided to assign a different priority to every task to assure only one task would execute in a time slice.

5.1.4 Supplier to a Small Airframe Manufacturer.

A manufacturer of a product suited for the small aircraft domain has certified their system to level D and has been partitioned within a level C application. The level D product is based on Windows NT with Service Pack 4. The manufacturer has sold over 200 units to date and has had the system in operation for over 3 years. The manufacturer was able to use the Windows NT COTS software by implementing a data protection scheme, and they are continuing to seek this technique and further research other techniques to qualify Windows NT to higher criticality levels. The manufacturer did not use traditional approaches in complying with DO-178B. They did not seek requirements, design, code, or other data from Microsoft. In fact, Microsoft was not involved at all in the software aspects of certification for this product. However, to gain acceptance of Windows NT at the higher criticality level, they plan to seek development data from Microsoft.

The manufacturer did informally use nonavionics service history for this COTS product; however, the argument was not used for certification purposes. The argument in particular was that given the overwhelming number of installations of Windows NT, it has had tremendous exposure. This widespread exposure resulted in an understanding of the problem set associated with this software and resulted in providing supporting detail of the product's readiness for this avionics application. The manufacturer is currently collecting their own service history data on this current product. They did not suppress any Windows NT functions, but did opt to use only those functions they felt were conducive to safe operation of their avionics application. They did consider putting a wrapper layer around some parts of the operating system features, but argued their technique for protecting data covered these concerns. The technique used can be considered a wrapper, but it was not specifically a wrapper for protection against undesirable operating system features.

The manufacturer has maintained the baseline of Service Pack 4 for several years and when new upgrades become available from Microsoft they will carefully analyze the change summary to determine the overall benefit to their product. They also admitted that the product directed both their software and hardware architecture.

The manufacturer selected this operating system for a variety of reasons. Cost was one consideration, but there were other intangibles that were difficult to measure (e.g., a functionally rich target environment, good development and support tools, and access to a large engineering manpower pool very knowledgeable with the Windows NT product). Most surprising to them

was the false sense of available manpower. Many engineers are knowledgeable in NT and the idea was to tap this resource to develop their product, but the other skill-set needed was an avionics domain knowledge coupled with rigorous software development practices. The combination of skills was difficult to find and to compete competitively with other employment opportunities for NT-experienced programmers.

Perhaps the biggest value of selecting this operating system is yet to be borne by this company as they feel NT has a high-function benefit that will allow their product to grow rapidly and provide improved time-to-market. The manufacturer admitted that desktop products in general are not good candidates for avionics, and the COTS components that are most likely to be effective are industrial-grade components with a wide distribution base such as medical-based products.

5.2 NUCLEAR INDUSTRY.

Data on actual application examples of COTS technology in the nuclear industry was difficult to obtain. A Nuclear industry representative indicated that the Nuclear Regulatory Commission (NRC) has not formally addressed the COTS issue, but they do have several robust standards on software development in nuclear reactors. There is a topical report available specifically on the usage of COTS for nuclear reactors, and the NRC has concurred with its approach [3]. The report is a guideline on evaluating and accepting commercial-grade digital equipment for nuclear safety applications and addresses both hardware and software aspects of COTS. The guideline requires that a software quality assurance component be in place for the COTS product. It also suggests that service history data is a viable argument for the COTS product, but data with regards to number of units and years of service is key to the integrity of such data.

The NRC suggests that all COTS products have a “safety kernel” wrapped around them, not only to prevent inadvertent operation, but also to positively warn and allow the equipment operator to react. The NRC is very concerned about the fact that COTS products are not typically stress tested and that they usually provide inadequate fault tolerance.

5.3 MEDICAL.

This survey included discussions with a Federal Drug Administration (FDA) representative and a medical device software safety expert. The FDA has established three levels for software criticality.

Software failure leading to…

… death or serious injury	High
… minor injury	Moderate
… no possible injury	Low

As in the avionics industry, software criticality levels can be reduced with hardware design considerations. Two standards are used in particular, 60601-1-4: Risk Management for Medical Devices and EN 1441. It should be noted that 14971: Risk Management is a draft that is expected to replace EN 1441. The FDA’s position on COTS is similar to current DO-178B

avionics guidelines. The medical device is regulated, not the software. As such, COTS software is not prohibited. However, the medical device software safety expert has indicated that COTS usage in high criticality level devices has not been approved at this point. With third-party COTS, a safety firewall must be demonstrated. In particular, stack overflows and other dynamic allocations are areas of concern.

The medical industry has a variety of products utilizing COTS software, which includes operating systems such as Microsoft Windows 3.1 and Wind River's VxWorks. Some operating systems have a historic track record and are viewed in better light than newer operating systems. Databases, user interface libraries, and compiler-based math libraries (along with input/output (I/O) drivers for data acquisition and motor-driver hardware cards) are other types of COTS software used. Of concern to the medical safety expert was the usage of compiler libraries, as many developers do not typically consider them COTS. A viewpoint from the medical industry, as in the avionics industry referenced above, is that since some of these products are sold in millions of units to other domains, they have rapidly established a historic track record and, as such, the FDA considers this wide usage in their evaluations.

Three different applications of COTS products were discussed with the medical safety expert. The nation's blood bank system has over 5,000 systems in place. A defibrillator has tens of thousands of units. And the pacemaker has very large numbers of units in the field. In general, most COTS products for these applications have the source code available. In the medical industry since there is a high cost mark-up and since the number of units sold is quite large, COTS vendors are more than willing to provide validation data simply to improve marketing edge. This however does not appear to be the case for the avionics industry as commercial vendors are reluctant to cooperate with supportive data due to low return on investment potential.

A noteworthy aspect of the medical industry's use of COTS that could perhaps translate into the avionics arena is an infrequently used proprietary data bank available only to the FDA. It was founded from more physical-related proprietary relationships and has been extended and used for COTS products. Essentially, a master file has been established where vendors can supply information to the FDA regarding their COTS product that no other competitors can access. The FDA can then use this repository data for evaluating the COTS product in the medical device application.

5.4 SPACE.

Information regarding the use of COTS in space was obtained from a report, AIAA 94-4506, "A Distributed Architecture for On-Orbit Laboratory Automation and Robotics Using COTS Components." In this application, a lab experiment governed by software was put to use on a space shuttle. This was a one-time application, and because it was not considered a risk to any astronauts, the normal 99.9% reliability coverage was reduced to 97.0%. The notable point of this study showed that a decrease in reliability could be one approach to leverage the use of many COTS components on an application. By accepting this reliability reduction, the team projected that the COTS aspect of the program reduced their costs by approximately 50%.

5.5 ELEVATORS.

The elevator industry is certainly immersed in software safety issues, as the governing software development guidance document it uses is IEC 61508-1. The elevator system architecture for control and dispatch includes not only LRU controllers but also a networked system that permits dispatching and diagnostic information to be passed between system components. One elevator company has partitioned their safety-related components in such a way as to allow nonsafety related software modifications to be protected from the software approved for safety purposes. One means of accomplishing this is with a COTS network driver (IXXAT CAN), which is layered with an Application Programming Interface (API) for the user to gain access to network services. Because the safety portions have been effectively partitioned, there are no elevator code requirements upon this network product. Interestingly enough, this commercial product was used to position the design into an architecture that allows for safety critical software to be segregated.

The safety critical software side of this effort has no COTS products in it. And the elevator company is using a tool for which the approval body is allowing some level of verification credit.

6. CURRENT MAJOR COTS COMPONENTS IN USE IN SAFETY-CRITICAL EMBEDDED SYSTEMS.

From the data obtained in this study, it appears there is only a small set of COTS components that are seriously considered for COTS usage in safety-critical domains. Most of these are of the type that enables the system to meet standards and open architectures.

Most vendors of COTS components used in the avionics industry that were contacted are not fully cognizant in DO-178B. Vendors made many inaccurate statements like “our software is FAA certified” or “our software meets DO-178B.” Most of the vendors do not truly understand the software avionics domain guidelines in DO-178B. Indeed most do not even fully understand the different criticality levels of avionics software. Vendors will state that because their product was used in avionics equipment that it is fully DO-178B compliant even though their product has been used only to level D.

6.1 OPERATING SYSTEMS.

Note: The operating systems listed are only some examples, the FAA is not endorsing any particular vendor.

There are several operating systems that have been used to some extent on certified avionics products including VRTX, LynxOS, PSOS, VxWorks, and OSE to name a few. To better understand the issues, this report will explore in more detail Enea’s OSE and WindRiver’s VxWorks operating systems. Note that the other operating systems listed also provide some level of safety consideration to their embedded product. No evaluation of capability between these vendors was made during this study, and hence, no judgments on their specific product are to be construed from this report.

Enea's OSE operating system appears to be an operating system that is providing direct support with respect to the safety domain. The vendor reports that the OSE kernel is IEC 61508 certified and that they seek to have a single kernel that can comply with both this and DO-178B guidelines. Currently, they are working with several aerospace companies, one of which is seeking to obtain system level A certification. It was reported that a level B application is near certification.

Enea not only supports the certification process, but they also have established a standard "safety-related" practice manual that describes how their product should be used for safety-critical systems. They additionally supply related DO-178B document templates for their customers to assist in integrating their product into the avionics system. Safety documentation that is included is the Safety Plan, Safety Requirements Specification, Safety Test & Validation Plan, Test Specification, Impact Analysis, Code reviews, and the like, regarding the COTS product.

WindRiver's VxWorks is also paying attention to the avionics safety arena as they are planning to deliver an "avionics certification development package" by the end of 2000. A WindRiver technical safety expert states the package is comprised of a subset API to the operating system and a set of documents supporting level A system certification as identified in DO-178B.

The medical industry has "shaken out" several operating systems. That is to say various operating systems have been exposed to the scrutiny of the NRC and acceptance of these applications in this regard is smoother. While other operating systems have been specifically identified as needing careful inspection and have become indicators for the thoroughness of application scrutiny.

A very good source of operating system information is, "Selecting a Real-Time Operating System," by Greg Hawley, that was published in the March 1999 Embedded Systems Programming Journal. It contains over 55 real-time operating system products; however, many of them are for nonavionic applications. The report shows most of the compilers are C/C++ compatible with about 50 percent Java compatible. Many vendors include source code with the product or make it optionally available.

6.2 COMPILER LIBRARIES.

Compiler libraries were not studied in detail for this report. The medical industry is very careful about making sure compiler libraries are included in the evaluation of an applicant's COTS product, given that many companies do not view it as COTS.

7. GUIDANCE FOR COTS TECHNOLOGY INSERTION.

During this survey it became apparent that many domains have drafted or approved standards providing guidance on how to evaluate, design, integrate, maintain, and control COTS software. Even the approval authorities that decide to treat COTS no differently than in-house-developed software have made available guidance on recommended practices for COTS technology insertion. Below is a list of sources on information regarding the process and standards.

a. A three volume, “Guide to Software Engineering Standards and Specifications,” by Magee and Tripp should be noted as an excellent resource [4].

b. The FDA has developed guidance for their industry, and the FDA has specific reviews on compliance with off-the-shelf software use in medical devices.

c. The NRC currently points to the Federal Information Processing Standards (FIPS) Publication Guideline ANSI/IEEE 1012, which provides uniform and minimum requirements for the format and content of software verification and validation plans. They also use IEC60880 Software for Computers in the Safety Systems of Nuclear Power Stations. The COTS specific guidance provides direction on handling COTS at a system level/hazard analysis level and details the additional considerations needed for utilizing such products [3].

d. The Software Productivity Consortium (SPC) details how to take their Integrated Systems and Software Engineering Process (ISSEP) and tailor it for component-based development [5].

e. The SEI has developed a separate software acquisition capability maturity model. The document reads much like SEI’s CMM in that it discusses all the levels (1-5) that have to be tailored to the software acquisition process. The document is very much practice-oriented and lacks depth in the nuances of COTS utilization [6].

f. In a presentation by Mark Vigder of the National Research Council USA, an architectural approach to building systems from COTS software components is presented. The premise is that requirements development and COTS selection are concurrent activities because selection impacts the architecture and adds development activities to the life cycle such as a defined COTS selection process, COTS specific testing and modeling, and an impact analysis on how the COTS product affects the design. Vigder suggests that all COTS products have wrappers designed around them [7].

g. In the June 1998 issue of *Computer*, Jeffery Voas details the challenges of using COTS software in component-based development. In this article he discusses the advantages of COTS and leveraging the technology, but cautions that a selection and integration process is needed. He warns of COTS restrictions and inflexibility and discusses the difficulties of finding the source of system problems with COTS products since they typically cannot be instrumented for failure analysis.

8. COMMERCIAL OFF-THE-SHELF INTEGRATION PROCESS.

A large number of both academic- and aerospace-based institutions have developed approaches to consider how to assess, acquire, integrate, test, and maintain COTS products, such as the Software Engineering Institute and Software Productivity Consortium. These approaches are different in nature than in-house development practices. Understandably the COTS vendor and applicant need to consider business relationships, configuration control of the product, and error-

reporting mechanisms and upgrades that are now out of the domain of the applicant. A pointed focus is needed on the verification aspects of the integrated COTS product.

NATO COTS Software Acquisition Guidelines defines their approach to evaluating the risks of integrating COTS software into a system. Four primary risk areas have been identified.

- Complex integration problems, which may lead to a failure to meet requirements.
- Cost and resource escalation due to this integration.
- Lack of product control: license agreements, product discontinues or not supported, and release management.
- Increase configuration management due to mixture of custom and COTS software [8].

Specific attributes to consider for COTS products integration are accuracy, availability, clarity, completeness, correctness, previous assurance methods, interface consistency, interoperability, performance (efficiency and timing), security, testability, understandability, user friendliness, etc.

The affects of COTS products on a system are such that a separate development process could be justified. Some credence to this process and some of the components of a good COTS integration process are listed below.

- Determine risks and hazards of the system and the relationship of the COTS component to those risks and hazards.
- Evaluate the COTS product for its fit into the system.
- Evaluate the COTS vendor's development process for the product.
- Develop an acquisition plan, license, lease, maintenance agreements, etc.
- Develop a Configuration Management (CM) and Software Quality Assurance (SQA) plan for handling the product.
- Develop a COTS test plan including both static if possible and dynamic testing.
- Develop an integration plan for the component. (User selectable options may require special integration procedures be developed)
- Acquire and submit problem reports with the vendor.
- Analyze problem and bug reports and design your use of the COTS product to accommodate any known anomalous behavior.

The list above is certainly not exhaustive and will require tailoring to each avionics applicant's development environment.

9. TECHNIQUES USED IN DETERMINING COTS INTEGRITY.

Techniques for mitigating the risks of using COTS components can vary widely and include COTS development process recognition, prior product certification, restriction of COTS functionality, hardware architecture, software architecture, and a wide variety of verification techniques. A brief (but certainly not all-inclusive) summary is provided below in subparagraphs. Little, if any, of these techniques can stand by themselves to support the intent of DO-178B. The avionics application developer should carefully evaluate the COTS product's functionality and pedigree and apply a number of techniques to assist in supporting the intent of DO-178B.

Special Committee 190 under the auspices of RTCA, Inc. has offered process recognition, prior product certification, restriction of COTS functionality, and other approaches that seek to demonstrate compliance with the intent of DO-178B.

9.1 COMMERCIAL OFF-THE-SHELF PROCESS RECOGNITION.

Process recognition is the acceptance of the evidence that a recognized development process was applied to a COTS product. In essence, if the process that developed the component possessed an independent accreditation, then credit may be taken for satisfying objectives achieved in the process. What needs to be assessed in detail in this case is the method of assurance used by the process and that it was conducted for the full development of the component. This may pose problems for the FAA since the process for delegation is clearly specified in the regulations and associated guidance material. Acceptance of independent accreditation is in effect a form of delegation [9].

9.2 PRIOR PRODUCT CERTIFICATION.

Prior product certification is a situation where the COTS component was approved as a part of a previously certified or qualified system avionics application. Examples provide a range of prior product certification domains including aircraft, medical, nuclear, and military. Again, an assessment is needed of the method of assurance used by the process [9].

9.3 RESTRICTION OF COTS FUNCTIONALITY.

The concept "restriction of functionality" involves restricting the use of a component to a subset of its functionality, by methods such as, but not limited to, run-time checks, design practices (e.g., company design and code standards), or build-time restrictions. The restricted set of functional requirements may then become the basis for use of the COTS product. Additional verification will need to determine that the restrictions are enforced [9].

9.4 HARDWARE ARCHITECTURE.

Today, the complexity of building our software and hardware intensive systems has certainly grown. Indeed, neither can be developed independently and, as such, to build a system with high integrity requires both to exist in a symbiotic relationship. This fact should be kept in mind when

considering either hardware- or software-based architecture schemes, because, in many instances, a combination of hardware and software efforts will be needed to complete the accommodation of the safety requirement. Some hardware architectural techniques in use today are listed below. The applicability of these techniques to COTS components will necessarily be established depending on their usage.

9.4.1 System Partitioning.

A technique where the safety-related portions of the system are built with high fidelity and separated from the rest of the system.

9.4.2 Memory Partitioning.

A technique to separate data and/or code such that one group of address spaces do not have an adverse affect on another group of address spaces.

9.4.3 Time Partitioning.

This is a technique used to separate the effects of the operation of one portion of a program on the timing behavior of another portion of the program.

9.4.4 Watch-Dog Timers.

Watch-dog timers are components that require regular attention from the operational software to prevent a predefined recovery or shutdown action. Watch-dog timers may be dedicated hardware components or a separate computer and associated software designed for this task.

9.4.5 N-Redundant Components.

N-redundant components are a way to accommodate faults in a system whereby the fault in one component is accommodated by another component redundant in functionality. Dissimilar, independent designs utilize this technique. This class of architecture could possibly discover random physical failures and design errors. The design should assure the independence of requirements, algorithm, data, and other potential sources of design error [10].

9.4.6 Network Separation.

A scheme to separate safety-critical portions of a system by limiting all communication between components with the use of packetization of messages versus the use of communication via shared memory space.

9.4.7 Safety Bag.

In this technique, an external monitor, called a safety bag, is implemented on an independent computer using a different specification. The primary function of the safety bag is to ensure that the main system performs safe operations. The safety bag continually monitors the main

system to prevent it from entering an unsafe state. If a hazardous state does occur, the system is brought back to a safe state by either the safety bag or the main system [11]. The safety bag is similar in concept to the active monitor parallel design technique described in reference 10 and in Section 9.4.5, N-Redundant Components.

9.5 SOFTWARE ARCHITECTURE.

9.5.1 Fault Detection and Accommodation Including Software Redundancy.

Fault detection is the process of checking a system for erroneous states. Fault detection checks values and timing for faults that may be physical (e.g., temperature and voltage instruments), logical (e.g., error-detecting codes), functional (e.g., assertions), or external (e.g., feasibility checks). Fault accommodation techniques can identify safe states where the system is operating properly. Through the use of diagnostic programs, the software checks itself and the hardware for incorrect results. The diagnostic programs can be run periodically or continuously as background processes. Diagnostic programs may include duplicating a calculation two or more times, parity checks, and cyclic redundancy checks. For critical functions designed with redundancy, voting between the redundant components is used to decide the correctness of those components [11].

9.5.2 Retry Fault Recovery.

Fault recovery via retry is often used by communication-related systems and is not a common technique for rapid real-time systems. The system monitors itself for a fault and will reset itself to a previous safe state and continue forward. If used in a real-time-related system, assurances need to be made that the recovery will be able to be completed before the fault can externally manifest itself at the system level [11].

9.5.3 *n*-Version Programming.

In *n*-version programming, independent teams produce a specified number *n* of software products called versions. Three versions are typical, however, for systems that have a safe state, two versions can be used with a bias toward the safe state. All *n*-versions of the software product are part of the software system.

Different programming languages and algorithms are often used to reduce the exposure to common-mode failures. However, common-mode errors could possibly still occur due to inadequate top-level specification. Various voting strategies can be used between the versions to select the output with the highest pedigree. There has been considerable debate as to realizing the full potential from *n*-version programming as it makes the assumption that the independence will lead to statistically independent mistakes. Evidence has shown that this premise may be faulty [12].

9.5.4 Recovery Block Programming.

Recovery block programming is a technique where independently written modules check themselves for correctness. The technique applied to COTS would be to isolate the COTS component in a module and, prior to exit, assess the results for any error. If the module detects an error, another module is instantiated, which cleans any side affects from the COTS encapsulated module and proceeds to operate error free. A concern of course is the ability of the module to properly assess its health and recover in a safe manner. The technique can be extended from modules to programs to subsystems if the system requires those levels of redundancy.

9.5.5 Model Following.

Model following is a technique where a rudimentary model of the COTS component is present in the system and used to verify correct operation of the COTS component itself. The model can be represented by any number of techniques, from a simple table look-up to a full-up model representation, depending upon the complexity and requirements of the COTS function to be modeled. Any number of strategies can then be taken to alleviate the COTS incorrect output from using the model results temporarily to accomplishing a total system reset. The action taken will depend upon the application's safety needs.

9.5.6 Wrappers.

Wrappers is a rather convenient word and, during this investigation, is used frequently as a suggested solution to protect the system from the COTS component or vice versa. The use of wrappers can be rather complex and could be a study unto itself. However, three fundamental types of wrappers were studied here.

A porthole wrapper is designed to allow access to the COTS functions via a small set of application-developed interfaces. These interfaces can be simple, doing little but validating and passing parameters, or they could be robust and determine user-privileges to access the COTS software. This protects the system from incorrectly using the COTS component, but does little in the way of protecting the system from the COTS component.

The strategy of a shell wrapper is to provide immunization to the system by encapsulating the COTS component. This particular technique requires a detailed knowledge of how the COTS products interfaces to all parts of the system and full immunization without detailed knowledge is very difficult.

A worm wrapper typically is used to encapsulate and protect data as it passes through a COTS component. An example would be utilizing a communications package whose robustness is unknown. The data to be transferred can be encrypted prior to the communication and decrypted after the data has wormed its way through the COTS component. This encryption could possibly include error detection and correction schemes, if time and data integrity so warranted.

9.5.7 Object-Oriented Architectures.

Object-oriented solutions have historically been shunned by real-time safety-critical applications due to their dynamic instantiation of data and functions as well as the lack of traceability when polymorphism is used. However, deterministic object-oriented approaches are evolving and are used in lower criticality software at this time. Some architectures or patterns are being offered as solutions to accommodate safety-critical issues in object-oriented base systems [13].

- Homogeneous Redundancy Pattern
- Diverse Redundancy Pattern
- Monitor-Actuator Pattern
- Safety Executive Pattern

9.6 VERIFICATION TECHNIQUES.

It is clear that verification techniques are prolific and have many attributes. Some are temporal in that they can be applied to different stages of the software development life cycle.

- For instance, formal methods are typically applied to the development stage of a component. Or if a component was developed with a specification that has a mathematical basis, it could possibly be verified after-the-fact using formal methods.

- Exhaustive input testing could only be conducted after the component has been generated.

- And indeed, some techniques even step back in time by reverse engineering the product and make some determination about the product's original fabrication.

However, the technique applied is not as important as to how the technique can meet the intent of DO-178B. How it meets the intent of DO-178B is an analysis that should be done on a case-by-case basis and is a function of the type of COTS being used. For instance, a control system component would use very different verification techniques than a component that had a database engine as its foundation. The verification techniques detailed below in subparagraphs are provided as candidates for the verification of COTS components. Why they are selected, how they are used, and where they are applied is left to the system developer. Each technique has its own attributes, and some can be modified to best fit the avionics application. Each technique has also its benefits and deficiencies and these will vary for different COTS components.

In section 12 of DO-178B, the guidance tries to give special attention to alternative methods for complying with the intent of DO-178B. Five alternate methods are discussed and the techniques vary greatly. These alternate techniques were emerging into maturity at the time of publication of DO-178B and, as such, provided some guidance in usage of the technique as it was known at that time.

A combination of these techniques might be used in demonstrating equivalence to one or more of the objectives of DO-178B. It is up to the developer to analyze the system's safety needs and determine the best alternate methods available to assist in meeting the intent of DO-178B.

Attention should be paid to the techniques and benefits that can be reaped by applying those techniques to a COTS component. Indeed, one can rationalize the type of component testing to be applied to a COTS component is a function of software level. A level C component would not need to be exercised as rigorously as a level A component. Each type of component-level test that is conducted on the software could possibly be measured and analyzed as to its effectiveness to meeting the intent of a DO-178B objective.

Note: The information below should not be construed as an exhaustive treatise on COTS verification techniques.

9.6.1 Functional Component Testing (Black Box).

Black Box means the type of testing where the tester does not use any knowledge of the internal structure of the component. In these cases, the testing is performed by exercising the inputs to the components while observing the correctness of the outputs. Although rudimentary in nature, functional component testing is a natural consideration for COTS components since linkable object code units may be available for testing in a stand-alone black-box manner. Compiler libraries are also COTS candidates for this type of testing. An example is a sine or cosine function from a math library. Individual units or modules can be tested for conformance to their specific functional task. This testing attempts to uncover several types of software faults such as incorrect operation, performance errors, and interface errors. These tests are generally developed from the component specification and the specific techniques and can be applied in various manners.

Additionally, since the function to be performed is known by the tester, tests can be developed to intentionally disrupt the unit's operation and its side affects can be observed and any anomalies discovered can be potentially mitigated by redesign.

Some examples of functional component testing are interface testing, boundary value testing, equivalence class testing, random input testing, environmental testing, stress testing, initialization testing, temporal testing, database testing, etc. This report discusses a few of these types of testing approaches immediately below.

9.6.2 Stress Testing (Black Box).

Stress testing is a functional technique that uses the black box method. It is particularly good practice to use this technique on the higher-level software components. The basic idea is to determine component degradation and failure modes. Stress testing places an exceptionally high workload on the component to verify that it will function easily under the expected workload and to determine any component failure modes. COTS components tested in this manner will yield a better understanding of the components' true operation and failure modes if discovered. This allows for the COTS integrator to consider component integration risk mitigation strategies if necessary. However, the tester may not be able to establish that the component has been stressed without an understanding of the internal structure of the component.

9.6.3 Process Scenario Testing (Black Box).

The goal of process scenario testing is to test software functions in their expected environment and usage. Flight profiles can be provided as the basis to exercise the component. This technique generates a wide range of operational scenarios for which the system is evaluated, including scenarios that are too dangerous or impossible to reproduce in the actual operating environment.

9.6.4 Equivalence-Class Testing (Black Box).

Large safety-related systems have an enormous number of inputs. Testing each of these inputs is impractical. It can therefore be very beneficial to divide the input domain into classes of data and then to use a sampling of inputs based on these classes of data to create test cases. Guidelines are used to establish the equivalence classes. As equivalence-class testing only uses a sampling of input values, not all input values are tested. This procedure can leave a software fault undetected [1].

9.6.5 Boundary-Value Testing (Black Box).

This is related to equivalence-class testing. Boundary value tests are those tests selected from equivalence classes that are directly on, above, and beneath the edges of input and output equivalence classes. Some definitions of boundary-value testing consider the inclusion of output equivalence classes and attempts to hit the edges of the output spaces as well [14].

9.6.6 Random-Input Testing (Black Box).

Random-input testing is also referred to as statistical testing. To quantify software reliability, the software product or its components are tested with both systematic and random approaches. The systematic perspective takes an overall approach to test control, whereas the random perspective is applied to a specific test case. For the random approach, a predicted distribution of system behavior determines the tests to be executed. Unusual states or rarely used portions of the system may not be included in the predicted distribution. For that reason, especially in safety-related systems, careful attention should be given to including test cases that require the system to take action only in rare but significant circumstances. Another approach is to apply random test cases to the input domain.

Since the number of test cases can be large, automatic testing tools are sometimes used to provide test-input data. Since a specification is not required, this method can be used on poorly documented systems, a characteristic not foreign to some COTS products.

Note that there is considerable controversy on the appropriateness of random testing to quantify software reliability. Software reliability is not easily quantifiable, hence, the controversy rages.

9.6.7 Performance Testing (Black Box).

Performance tests verify response times under varying loads, throughput, memory utilization, and many other real-time properties. Performance is a critical requirement in real-time-embedded systems. Improper timing for a component can affect the system's overall ability to perform and can fail the system. Results of the performance tests can be used to optimize the system for the best throughput. Caution needs to be applied in that performance testing requirements for COTS components could be difficult to generate, demonstrate repeatability, and difficult to prove for worst-case values.

9.6.8 Formal Methods.

DO-178B/ED-12B defines formal methods as "descriptive notations and analytical methods used to construct, develop, and reason about mathematical models of system behavior."

Formal methods provide a means of developing a description of a system at some stage in its specification, design, or implementation. The resulting description is in a strict notation that can be subjected to mathematical analysis to detect various classes of inconsistency or incorrectness [11]. By the use of the mathematical proofs available from formal methods, the verification data needed to demonstrate compliance to higher-level requirements might be replaced under certain cases. In practice, the application of formal methods may be limited by the complexity of the system or subsystem [9]. For the most critical systems, a formal mathematical approach will give added confidence.

Formal models generally describe only some aspects of a system's behavior, and behavioral characteristics of sequential programs are straightforward to model. At the verification side of the development, it is difficult to ensure that the actual behavior of the running system will match that described in a formal model. The correspondence is limited by factors such as the programming language, compiler, operating system, and hardware. It is not possible to assume that a program will function correctly simply because correspondence between the high-level source code and the specification has been proven.

10. A DEEPER LOOK AT SOME ALTERNATE METHODS.

10.1 REPLACING MISSING DOCUMENTATION.

Considerable controversy rages about the acceptability of various approaches to replacing, reconstructing, or substituting for missing COTS product documentation. Some viewpoints focus on the question of applying engineering judgment to these and other assessment questions. In that light, engineering judgment should be applied carefully to specific, narrowly defined questions, and that the rationale for the judgment should be documented and able to withstand critical, external scrutiny. Engineering judgment cannot be used to justify generic arguments about COTS product acceptability.

10.2 REVERSE ENGINEERING.

An approach to replacing missing data is to perform data reconstruction with methods such as reverse engineering. This can be a difficult task requiring as much effort as doing a new development and, even if accomplished, it is not clear that the reconstruction process was error free. However, it may produce software life cycle data that can be reviewed or analyzed to satisfy the intent of the objectives of DO-178B/ED-12B, such as design structure, source code, or calling trees. Indeed, some have argued that reverse engineering can elicit structural information that could provide additional evidence that the component was developed with the use of a standard.

10.3 SERVICE HISTORY.

Product service history is the utilization of previous in-service experience of the component. Previous use of a software product that is relevant to its intended application may constitute evidence of product integrity. The data integrity of the service history records needs to be validated. Details of the service history are needed, including information about the problem tracking process and software configuration management process. Care should be taken that the environment that the service history records were tracked and is applicable to the new environment.

It seems intuitive that operating experience data derived from extensive usage of the same version of a product in similar applications would indicate that the product is acceptable for the intended application. Several issues with this assumption should be considered. First, configuration data of the actual version used in the problem report supplied can be difficult to obtain. As such, the statistical validity of the data is unknown. Error-reporting databases commonly span multiple releases and configurations of a given product. Another issue is that circumstances surrounding the occurrence, monitoring, and recording of failures are often vaguely reported. Even the highest avionics software level system have difficulty in reproducing and diagnosing problems due to the inability to recreate the problem from the information provided. Indeed, some of these activities are not only improperly controlled, but in fact, there might be some motivation for COTS vendors to limit publication of negative experiences, particularly if the COTS product was not originally intended for safety-critical systems.

Another key issue regarding operating experience is that extensively used products can still have crucial faults that could cause problems in safety systems. The tendency is to consider operational experience to be like extensive random testing. In this regard, operational experience suffers from the same shortcoming as testing: testing cannot prove the absence of faults.

In contrast, the goal of assurance mechanisms such as DO-178B is to provide confidence that the in-service experience will be acceptable for safety-rated systems prior to being put into service. If there were some way to peer into the future and determine the in-service reliability at the end of a program, then one would not need the a priori assurance methods such as DO-178B. The goal then is to approximate this situation. One method is to examine service experience is in allied fields where the possibility of taking advantage of extremely large numbers exists. If some of the difficulties of applicability of service experience can be overcome, then there is a possible

approach for acceptance of COTS. This is an area where more research is needed both from theoretical (i.e., statistically valid approaches) and a practical (i.e., what information is realistically available and the degree of confidence in that information). From a theoretical viewpoint, it is possible to account for some degree of uncertainty in the data, although this would probably only be applicable for software whose failure conditions are classified as major or less severe.

11. COMMERCIAL OFF-THE-SHELF OPERATING SYSTEMS.

At the time of this report's research effort, numerous operating system vendors were seeking to provide their commercial operating system (O/S) so that it could satisfy the intent of DO-178B, level A capability on a true off-the-shelf basis. Several of these vendors shared their approaches and expected deliverables with the principal investigator. In particular, one vendor shared, in detail, their approach to developing a "DO-178B-ready" operating system. The following observations were made in these approaches and many of these observations may extend to other COTS products anticipating DO-178B readiness.

Some vendors were working directly with applicants on a particular aircraft system, while others hired experts to assist in developing a DO-178B-ready package. All of these efforts to make the COTS operating system DO-178B-ready were done with the COTS vendor and some form of the COTS developmental data was available. Because of this availability of data, the task of meeting the intent of DO-178B was somewhat simpler than trying to do the same without vendor cooperation.

11.1 AN O/S-BASED SOFTWARE HAZARD ANALYSIS (SHA).

One vendor theorized that if they view their product as an airborne system by itself then they would be able to improve their understanding of the integration of their COTS product into the real avionics application. The approach taken was to first conduct a software hazard analysis, and from that derive requirements necessary to mitigate the risks of identified potential hazards. This approach not only provided risk mitigation, but it also identified those items the applicant should consider when developing their final product. In essence, the vendor provides an indication of what the applicant has to resolve from an O/S hazard point of view. This technique however does not alleviate any obligation by the applicant to perform his or her own system safety assessment on the system containing the COTS O/S.

11.2 AN O/S PLAN FOR SOFTWARE ASPECTS OF CERTIFICATION (PSAC).

The need for a PSAC for an avionics application is clear, as it provides the FAA with a fundamental plan of what, who, when, and how the applicant will seek certification. Several vendors planned on providing a separate PSAC for their O/S that was to be incorporated into the applicant's overall system PSAC. Other vendors were providing a PSAC template for the O/S considerations. One vendor was further offering professional services for assisting in the planned development if the applicant deviated from the board specific options of the delivered DO-178B-ready COTS package.

11.3 ALTERNATE METHODS CONSIDERED FOR A COTS O/S.

11.3.1 Reverse Engineering.

Even with available data from the COTS development, many COTS vendors relied heavily on reverse engineering. This was done with a combination of reverse engineering the assembly code, reconstructing design information and other missing data. Perhaps one of the most driving reasons for doing reverse engineering from the code to requirements was for traceability purposes. A traceability matrix was generated by one vendor between the object code, the generated source code, and the original header information.

Other information could be derived from this data. For instance, the reverse engineering of code provides a map to the code's structure. The structure maps could perhaps now provide insight into whether the code was built to a standard. If so, then this information along with perhaps a CMM level 3 assessment might provide an argument that the vendor has complied with the intent of the design and code standards objectives of DO-178B.

11.3.2 Partitioning.

Vendors had different views as to how to handle partitioning. At least two vendors planned on providing multiple criticalities-partitioning capability in their O/S, and at the time of this writing, were still attempting to do so. Another decided to have no partitioning integrity checks, as this O/S vendor is assuming the system safety assessment will require all tasking be done to the same level.

11.3.3 Restriction of Functionality.

Some safety attributes of real-time systems, has resulted in one vendor prohibiting the applicant from using O/S capabilities that could foster a hazardous condition. For instance, the vendor only allows memory allocations on initializations and does not allow memory de-allocations. Much of this restriction of functionality comes via selectable features within the selected compilers. One O/S vendor has an environment that assists in building the O/S and avionics application into the target computer. This environment also has some amount of restriction of functionality to disallow potential hazard-generating situations.

11.4 COMPLIANCE TO STANDARDS.

For standards compliance, one vendor considered only the safety-related aspects of the standards. Essential safety sets of standards were developed from reverse engineered components, and those standards were then used in the review of other reverse-engineered components.

11.5 REVIEWS—REQUIREMENTS, DESIGN, AND CODE.

Reviews are the one data item that, from a COTS point of view, is rather difficult to produce and reproduce, because in most instances, the original reviewed item is not available. Many of the

vendors sought to comply with the intent of DO-178B in several ways and some approaches are listed below.

Requirements Review: One vendor's approach was to carry out a safety analysis into the structure of the software, which they refer to as "software hazard analysis." There is no industry standard for this approach.

Design Review: One vendor's design review was again scoped to safety-specific aspects. This included looking for recursion type functions or other in-determinant code.

Code Review: A focus for this review for one vendor was to determine if there were any areas that could not be tested.

11.6 COMMERCIAL OFF-THE-SHELF INTEGRATION.

Integrating the COTS component into the avionics application is a major concern to the vendors, and indeed, to be DO-178B ready, they need to have an applicant willing to use their product in a certification. Indisputably, all vendors expect the applicant to verify the interfaces to the O/S and verify the timing aspects of their avionics application.

11.7 SOFTWARE QUALITY ASSURANCE CONSIDERATIONS.

Software quality assurance should consider not only obtaining historic SQA data from the vendor, but also have oversight in the integration and test of the component. In addition, if an outside party such as an independent consultant was used, then they too need to address the SQA aspects of DO-178B to their work.

11.8 THE GOING FORWARD POSITION FOR A VENDOR-SUPPORTED UPDATE.

The question of updates to the system creates several issues that need to be addressed.

- Configuration control and acceptance of the COTS product is essential.
- With a new version delivery, the vendor needs to consider what their plan is for supporting the DO-178B objectives.
- The applicant needs to assess and perform regression testing on the component in such a way that no new undesirable side affects are evident.

These and other items should be addressed in the applicant's PSAC.

12. DO-178B'S OBJECTIVES AND ITS AFFECT ON COTS.

The intent of this section is to identify the DO-178B objectives that hinder the use of typical COTS components. These DO-178B objective hindrances may be overcome with the use of

alternate methods discussed elsewhere in this report. Typical COTS products mentioned below are those that were not originally developed to DO-178B.

12.1 LEVEL A.

Level A software has 66 objectives for compliance. With the evidence available today, in general, it can be stated that COTS products typically do not meet the requirements for level A criticality software. In particular COTS products do not typically meet the Modified Condition Decision Coverage (MCDC) objective. Indeed, many of the COTS vendors contacted were not familiar with the term MCDC. Additionally, the requirement of independence on 25 of the 66 objectives further hinders some COTS products from use in level A systems.

12.2 LEVEL B.

Level B software has one less objective to meet than level A, and that is the MCDC objective. This does little to open up the usage of COTS for level B systems. There is no preponderance of evidence available today that demonstrates compliance of typical COTS software to level B. In general it can be stated that, COTS products do not typically meet the requirements for level B software. DO-178B's level B's structural coverage objectives and the requirement of independence on 15 of the 65 objectives again hinder some COTS products from use in systems with software level B.

12.3 LEVEL C.

Level C opens the typical COTS market for avionics systems. MCDC, decision coverage, and other objectives have been eliminated. In addition, the target compatibility with requirements objective and the source code and architecture requirements have been reduced. COTS products will still need to substantiate even simple statement coverage, and as such, alternate methods may need to be used to show the intent of DO-178B compliance.

There are yet several objectives that could make compliance from a COTS product point of view thorny. Statement and data/control coupling coverage objectives are still in force, and one popular technique is to have a good traceability capability to accomplish meeting these objectives. Reverse engineering has been one alternative method used to gather trace information to enable accomplishment of the "statement and data/control coupling" structural coverage.

Software development processes, standards, and their inter-relationships are to be defined for level C. The best evidence of this would come from the COTS vendor. However, data replicated or reverse-engineered could possibly demonstrate that the intent of DO-178B is being met. When an applicant is demonstrating they have met the intent of any objective, they need to offer alternative evidence to support their claim. This can come in several forms or combinations of forms as discussed elsewhere in this report such as reverse engineering, service history, process recognition.

12.4 LEVEL D.

For level D, the major objectives to meet include a definition of high level requirements (HLR), low level requirements, and source code developed and operational on the target computer. Verification of HLRs and that they trace to system requirements are necessary. Most importantly, perhaps, is the partitioning integrity. Configuration management objectives and independent SQA activities also need to be met.

Many of the typical COTS packages offered can comply with these objectives. The objective that perhaps needs the most attention is the partitioning integrity as it is critical to assure the level D software will not affect other higher-level software in the system.

13. TOOLS.

The research in the area of tools for this contract was limited to an "as discovered" basis. That is to say, if in this study of COTS, as pertinent tool information became available, it would be documented.

13.1 AUTO-CODE GENERATOR.

A commercial auto-code generator Safety Critical Application Development Environment (SCADE) has appeared to be given credence, as one nonairborne certification authority has recognized the tool to allow for a reduction development effort in the module test area.

13.2 TARGET ENVIRONMENT WIZARD.

The technology for tools to assist in application/system development is growing. In one instance, one interviewee reported that their design environment allows them to create shell code for their complex system by using a wizard. The wizard allows for the selection and integration of components based on processor and hardware configured in the system, thus bridging the gap between the hardware and software design definition process.

13.3 REQUIREMENT-BASED TOOLS.

Several tools exist for tracking requirements and other system (design/code) entities. Yet the technology is now growing to effectively analyze the system requirements and other components to the point where regulatory agencies need to pay attention as to the robustness of the analysis tool and whether it should be used in lieu of other requirement analyses processes.

13.4 TOOL GUIDANCE IN SAFETY-RELATED EQUIPMENT.

Professor Brian Wichmann of the British Computer Society has drafted "Guidance for the adoption of tools for use in safety-related software development." The paper goes into detailed issues in regards to tools in safety-related systems. He has some verbiage on guidance for regulatory authorities use that is comes in the form of questions posed to tool suppliers. (http://www.iee.org.uk/PAB/SCS/tools.html)

14. REFERENCES.

1. Requirements and Technical Concepts for Aviation, SC-167, "Software Considerations in Airborne Systems and Equipment Certifications," DO-178B, December 1992.

2. J. A. Scott, G. G. Preckshot, and J. M. Gallagher, "Using Commercial Off-the-Shelf (COTS) Software in High-Consequence Safety Systems," UCRL-JC-122246, LLNL.

3. Electric Power Research Institute, Working Group, "Guideline on Evaluation and Acceptance of Commercial Grade Digital Equipment for Nuclear Safety Applications," EPRI TR-106430 4488-01 Final Report, October 1996.

4. Magee, S. and Tripp, L., *Guide to Software Engineering Standards and Specifications*, three volume set, Artech House Publishers, February 1997, pp 330.

5. Software Productivity Consortium, "Integrated Systems and Software Engineering Process" Appendum C: Component Tailoring, SPC-98116-MC, Ver 1.0.0, February 1999 (note: available to member companies only).

6. Software Engineering Institute, "Software Acquisition Capability Maturity Model," CMU/SEI-99-TR-002, April 1999.

7. Vigder, Mark, "An Architectural Approach to Building Systems From COTS Software Components," 2nd Annual Software Engineering Workshop, NASA/Goddard Space Flight Center, Greenbelt, MD, December 3-4, 1997.

8. Debra S. Herrmann, *Software Safety and Reliability*, IEEE Computer Society Press, 1999, p. 218.

9. 2nd Annual Report, Special Committee 190, 2000.

10. Aerospace Recommended Practice 4754 Certification Considerations for Highly Integrated or Complex Aircraft Systems, 1996.

11. Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems, International Electro-Technical Commission, 1998.

12. Leveson, Nancy, *Safeware: System Safety and Computers*, Addison-Wesley, 1995.

13. Douglas, Bruce Powel, "Reliable and Safe: Patterns and Practices for Designing Mission and Safety-Critical Systems," Software Development Conference Proceedings, Spring 1998.

14. Myers, G., *The Art of Software Testing*, John Wiley & Sons, 1979.

15. ADDITIONAL INFORMATION.

United Technologies Research Center, Thornton, R., “Review of Pending Guidance and Industry Findings on COTS Electronics in Airborne Systems,” October 2000.

Friedman, Voas, “Software Assessment: Reliability, Safety, Testability,” Wiley & Sons, 1995. An overview which integrates a wide range of software testing, safety, and reliability topics.

Parnas, van Schouwen, Po, “Evaluation of Safety-Critical Software,” Communications of the ACM, 33(6), pp. 636-648, June 1990. An alternate view to Leveson’s work that feels safety engineering should not be treated separately from reliability. Topics include code reviews, software documentation, and software reliability estimates.

APPENDIX A—SURVEY QUESTIONS

Below is the set of survey questions used when interviewing industry on their use of COTS.

1. COTS product name/version info
2. COTS company name, phone, contact info
3. Domain (avionics, medical, nuclear, railroad, other)?
4. Type of software (I/O, math, O/S, etc.)?
5. Number of units sold
6. Number of units used in domain application
7. Assurance/certification method (FAA certification, internal QA only, external, TUV)?
8. Software level (criticality level, safety integrity level)?
9. Life cycle data availability (requirements, design, code, test, PR)
10. COTS versioning (baseline, upgrade)
11. What CM does the buyer use?
12. What CM does the seller use?
13. Service history?
14. Seller active in certification for application?
15. Modifiability (at integration/load time?)
16. Suppression of unwanted functionality?
17. Cost-benefit information if available (acquisition, maintain, verification, licenses, source code buyout)
18. Did COTS affect the architecture of the application?

APPENDIX B—JOURNALS USED WITH AUTONOTIFICATION

ACM Transactions on Programming Languages and Systems
Advances in Engineering Software
Aerospace Engineering and Technology
Aerospace Science and Technology
AIAA Journal
Aircraft Engineering and Aerospace Technology
Aviation Week & Space Technology
Computer
Computer Languages
Computer Science, Interdisciplinary Applications
Computer Science, Software, Graphics, Programming
ESA Bulletin-European Space Agency
IEEE Aerospace and Electronic Systems Magazine
IEEE Transactions on Aerospace and Electronic Systems
IEEE Transactions on Software Engineering
Journal of Object-Oriented Programming
Journal of Systems and Software
Object-Oriented Systems