DOT/FAA/AR-06/2

Office of Aviation Research
and Development
Washington, DC 20591

Flight-Critical Data Integrity Assurance for Ground-Based COTS Components

March 2006

Final Report

U.S. Department of Transportation
Federal Aviation Administration

Technical Report Documentation Page

1. Report No. DOT/FAA/AR 06/2	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle FLIGHT-CRITICAL DATA INTEGRITY ASSURANCE FOR GROUND-BASED COTS COMPONENTS		5. Report Date March 2006
		6. Performing Organization Code
7. Author(s) Yann Hang Lee[1] and Jim Krodel[2]		8. Performing Organization Report No.
9. Performing Organization Name and Address [1]Arizona State University Office of Research and Sponsored Projects Box 873503 Tempe, AZ 85287	[2]United Technologies Pratt & Whitney Aircraft 400 Main Street East Hartford, CT 06108	10. Work Unit No. (TRAIS)
		11. Contract or Grant No.
12. Sponsoring Agency Name and Address U.S. Department of Transportation Federal Aviation Administration Office of Aviation Research and Development Washington, DC 20591		13. Type of Report and Period Covered
		14. Sponsoring Agency Code AIR-120

15. Supplementary Notes

The Federal Aviation Administration Airport and Aircraft Safety R&D Division Technical Monitor was Charles Kilgore.

16. Abstract

Ground processing systems are likely to use commercial-off-the-shelf (COTS) software and hardware for maintaining flight critical data. Hence, the COTS ground processing systems must be trustworthy and secure in order to maintain the integrity of the data. This requires various approaches, including information protection, data integrity and access security.

This report describes the results of our studies on the existing guidance governing the use of COTS components for safety-critical applications of ground-based systems, and the objectives of current guidance from the point of view of applicability and shortcomings. The use of hazard analysis and vulnerability analysis as a means for developing an effective risk mitigation strategy is provided and the relevance of the Rotorcraft Advisory Circular to a COTS components scenario is summarized. In order to address security and vulnerability concerns, several technologies such as encryption/decryption, authentication, access control, intrusion detection, etc., particularly in relation to application within a health and usage monitoring system (HUMS) context, are discussed. The report also includes two case studies involving COTS products to determine if they can be qualified by following existing software guidance, such as DO-178B and DO-278.

17. Key Words COTS, Ground-based, HUMS, Security, Safety, Access, Process, Software	18. Distribution Statement This document is available to the public through the National Technical Information Service (NTIS) Springfield, Virginia 22161.

19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 86	22. Price

ACKNOWLEDGEMENTS

The authors wish to express their appreciation for the assistance of the Federal Aviation Administration who provided valuable comments and suggestions. The authors would also like to acknowledge their deep appreciation for the support of William Weiss of United Technologies Research Center who provided significant contributions to this research.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

AC	Advisory Circular
ACARS	Aircraft Communication Addressing and Reporting System
ACL	Access control list
AIS	Automated Information System
CAST	Certification Authorities Software Team
CC	Common criteria
CFR	Code of Federal Regulations
CNS/ATM	Communication, Navigation, Surveillance/Air Traffic Management
CORBA	Common Object Request Broker Architecture
COTS	Commercial off-the-shelf
CRC	Cyclic redundancy check
DBMS	Database management system
DoS	Denial of service
EAL	Evaluation assurance level
FAA	Federal Aviation Administration
FHA	Functional hazard assessment
GUI	Graphical user interface
HF	High frequency
HUMS AC	Health and Usage Monitoring System Advisory Circular
HUMS	Health and usage monitoring system
ICA	Instructions for Continued Airworthiness
ICBM	Inter Continental Ballistic Missile
IDS	Intrusion Detection System
IT	Information technology
IVM	Independent verification means
MAC	Message Authentication Code
MD5	Message Digest version 5
MDC	Modification Detection Code
N/A	Not applicable
NAS	National Airspace System
OS	Operating system
PC	Personal computer
POC	Proof-of-concept
PP	Protection profile
SHA	Secure hash algorithm
ST	Security target
TOE	Target of evaluation
VB	Visual Basic®
VHF	Very high frequency

EXECUTIVE SUMMARY

Ground-based processing systems for aviation applications may process and store flight-critical data. Hence, they must be trustworthy so that the integrity of the data can be assured. To be trustworthy, a safety-critical, ground-based avionics system should provide not only security, but also a number of other system properties, such as reliability, safety, and quality of service. The increasing reliance on data networks and commercial off-the-shelf (COTS) components has continued to expand the focus of safety and security from individual subsystems to the connections between them.

Issues in making a COTS component-based, ground-based processing system for aircraft maintenance trustworthy and secure have been investigated. Three guidance documents constitute the basis of the investigation: (1) RTCA DO-178B, Software Considerations in Airborne Systems and Equipment Certification, which provides guidance for airborne systems; (2) DO-278, Guidelines For Communication, Navigation, Surveillance, and Air Traffic Management (CNS/ATM) Systems Software Integrity Assurance, which gives guidance for ground-based systems; and (3) the Health and Usage Monitoring Systems (HUMS) Advisory Circulars (AC) (AC 27-1B Change 1 and AC 29-2C Change 1), which provides guidance to achieve airworthiness approval for rotorcraft HUMS installation, credit validation, and continued airworthiness.

This report includes the investigation results on a variety of issues that arise from the use of COTS components in safety-critical systems and the approaches needed to ensure the integrity of the data. With a focus on HUMS, the report first gives a big picture view at the use of COTS in ground systems, the advantages and challenges posed by the use of COTS components. The issues in applying the current available guidance relating to COTS and ground-based systems are also discussed. This discussion includes the guidelines available in DO-278 pertaining to planning, acquiring, verifying, and managing the use of COTS software.

Research was conducted into possible safety hazards and security threats leading to a loss of data integrity by using vulnerabilities and safety hazard analysis techniques. The first step was to identify the threats and hazardous events in the system. Following this identification, the possible security and safety impacts and the criticality to the system operation were determined. In addition, possible remedial techniques were devised to address the hazards and threats. A risk mitigation strategy was adopted for each hazard and vulnerability corresponding to its risk level. This strategy may, in turn, affect some of the original requirement definitions at the design phase, resulting in a more robust design.

A significant amount of research effort was devoted to the evaluation of the applicable DO-178B and DO-278 objectives to the COTS components of HUMS. A preferred method of determining the applicability of the DO-178B and DO-278 objectives would be to perform a detailed analysis of each system component for all the objectives of the applicable guidance document and provide a rationale of compliance. A sample of the research investigation into this technique is presented in the report, specific for the HUMS system under study. It provides insight into possible ways in which such components may be analyzed, even if it is not always possible to

meet the required objectives due to the nature of COTS components. Of key importance in this technique is the component's assigned assurance level from the system safety assessment.

To evaluate the feasibility of risk mitigation techniques to address various hazards and vulnerabilities in the HUMS system, a scaled down demonstration prototype of a HUMS system was developed. In this prototype, risk management mechanisms are implemented in a scaled-down, proof-of-concept version to determine the feasibility of such mechanisms. The application of lessons learned from the case study and demonstration prototype were considered for the HUMS case, however, they may be applied to other ground-based systems (such as communication, navigation, surveillance, and air traffic management systems).

As a part of this investigative report, a study on the use of COTS in a different domain area for the purpose of safety and security was performed. Towards this end, a case study was performed on COTS component use in the development of a nuclear certifiable tester for the Minuteman III Inter Continental Ballistic Missile. The study found that, similar to COTS components in HUMS, the primary advantage gained in adopting a COTS approach is the reduction in design cycle time and costs. In addition, there is an absolute necessity for a planning, requirement, design approval, and certification process to incorporate COTS components in safety-critical applications.

1. INTRODUCTION.

The primary purpose of this research is to investigate the challenges involved in making the commercial off-the-shelf (COTS), ground-based processing system for aircraft maintenance trustworthy and secure. With the advent of powerful information management and communication tools, flight data and maintenance records can be easily collected, analyzed, and disseminated for use by ground-based systems with COTS components. This presents an opportunity for cost optimization and an increase in efficiency.

For instance, in the helicopter domain, a health and usage monitoring system (HUMS) may provide several benefits including improved aircraft safety, increased availability and reliability, rapid determination of aircraft status, reduced overhaul and repair costs, and reduced scheduled component removal. Many helicopter parts are dynamically loaded and are subject to develop fatigue cracks. If a flight-critical part fails, it could cause a catastrophic accident. As a result, many of these parts are life-limited and are assigned a safe-life limit in flight hours based on cumulative fatigue damage under variable amplitude loading. For example, according to Miner's theory of linear damage rule, the total fatigue is the sum of the cumulative damage proportions at different stress levels. To calculate the safe-life limit, the aircraft's usage or flight maneuvers (flight regimes), the frequency of those maneuvers, or the percent usage of that regime in a particular time frame is determined. This information is combined with material strength characteristics and part test results, and then used to establish an acceptable calculated retirement time for these parts. This calculated retirement time is conservative. The parts are typically removed from service once the predefined retirement time is reached. However, structural usage monitoring approaches dynamically adjust service life for individual components based on measured actual usage spectrums while the helicopter is being flown. This results in cost optimization and efficient use.

COTS components are used in many applications in ground-based aviation systems. For these systems, the hardware is likely to be a set of personal computers (PC). The operational software is also COTS, such as Microsoft® Windows®. The flight data may be saved in COTS storage devices. Also, it is very likely that the equipment is connected to the Internet or wireless networks, and installed in a nonsecure environment, and may involve human intervention in the decision-making process. A different approach to system design, certification, or a mitigating action may be required to make the system fail-safe.

As previously stated, the primary purpose of this research is to investigate the challenges involved in making ground-processing aircraft maintenance systems composed of COTS components trustworthy and secure. The investigation primarily focuses on three areas:

- Information and data protection. Data can be corrupted at any point during its lifetime. Corruption can occur during computation, transmission, and storage. This area deals with the protection and verification of information. Traditional techniques to handle these problems are investigated and new techniques or combinations of existing approaches are suggested.

- Access security. Flight data is very vulnerable to malicious attacks as it can be intercepted during data transfers and tampered with. In order to prevent this, various forms of access security such as authentication, digital signature, and public key infrastructure are investigated. The challenges involved in setting up such an infrastructure, as well as the issues related to accountability of flight data accessed are investigated.

- Process and objectives for data integrity in ground-based COTS systems. Data integrity must be maintained. To ensure this, processes followed by the ground-based COTS systems are analyzed to identify possible fault conditions and their potential impact on safety requirements. Fault mitigation techniques are proposed to ensure that all objectives for maintaining data integrity in ground-based COTS systems are satisfied.

This report closely considers the existing guidance: (1) RTCA DO-178B, Software Considerations in Airborne Systems and Equipment Certification, which addresses the software components in airborne systems; (2) DO-278, Guidelines For Communication, Navigation, Surveillance, and Air Traffic Management (CNS/ATM) Systems Software Integrity Assurance, for the assurance of flight-related software contained in nonairborne CNS/ATM systems; and (3) the HUMS Advisory Circulars (AC) 27-1B Change 1 and AC 29-2C Change 1) for airworthiness approval for rotorcraft HUMS installation, credit validation, and continued airworthiness. The applicability of this guidance material to COTS components and any remedial techniques that help meet the objectives outlined in the guidance were investigated.

To investigate the above stated areas of focus, the following steps were taken:

- Studied current and emerging industry approaches and guidelines for ground-based data protection schemes with a focus on HUMS and condition-based maintenance systems.

- Studied the application of security and authentication approaches in other industries and identified their limitations and strengths.

- Evaluated the objectives of DO-278 to the COTS components in HUMS application and the approaches listed in the Rotorcraft HUMS Advisory Circular for certifying HUMS ground-based systems that contain COTS components.

- Developed a functional hazard and vulnerability analysis as a means for developing an effective risk mitigation strategy.

- Performed two case studies involving COTS products to determine if they can be qualified by following the existing software aspects of certification guidance such as DO-178B and DO-278. Case study 1 investigated both Microsoft Windows and Microsoft® Access® as COTS components. Case study 2 considered a customer-designed HUMS system, which used a COTS database management system (DBMS).

- Conducted a representative demonstration project to evaluate the flexibility of the proposed protection process and objectives in ensuring proper handling of flight-critical information.

The report is structured as follows:

- Section 2 defines the term COTS and describes the motivation to use COTS components in building ground-based, safety-critical systems. A detailed discussion on the advantages (such as faster and cheaper development cycles) and challenges (such as ensuring safety and security) when using COTS components in safety-critical systems is provided.

- Section 3 provides an introduction to the HUMS, which is chosen as a representative ground-based system for the study. A functional description and a top-level architecture of a HUMS system are provided. The safety and security concerns of a HUMS system are highlighted.

- Section 4 provides a description of the current available guidance relevant to use of COTS components in safety-critical systems. DO-278, DO-178B, and the HUMS Advisory Circular (HUMS AC) are evaluated to provide an introduction to the existing approaches of integrating COTS components in safety-critical systems. This section also discusses the guidelines pertaining to planning, acquiring, verifying, and managing the use of COTS software available in the DO-278, DO-178B, and the HUMS ACs. Alternate approaches for acceptance of COTS software in safety-critical systems are also discussed.

- Section 5 presents the current and emerging industry approaches regarding safety and security. An introduction to cryptography, access security, intrusion detection, and information and data protection technologies is presented. The use of these technologies is also discussed in context of a HUMS. Toward this end, vulnerabilities to which airborne data is prone and the ways by which it can be protected are highlighted. The notions of safety and risk mitigation strategies are introduced. This section also presents a hazard and vulnerability analysis of HUMS and its various subsystems, such as ACARS, HUMS database, COTS software, HUMS gateway, and HUMS internal network. Appropriate safety and risk mitigation approaches are presented. The section concludes with a discussion on integration of safety and security approaches for HUMS.

- Section 6 uses the HUMS as a representative ground-based system to see how the COTS-specific objectives of DO-278 help in the planning, acquisition, verification, and configuration management process. A preferred method of determining the applicability of the DO-278 objectives would be to perform a detailed analysis of each system component for all the objectives of this guidance document and provide a rationale of compliance. To illustrate this technique, two case studies involving use of COTS software in a sample HUMS system are presented. Case study 1 investigated both Microsoft Windows and Microsoft Access as COTS components. Case study 2 considered a customer-designed HUMS system, which used a COTS DBMS. Both case studies determine the applicability of the DO-278 objectives to the COTS component by performing a detailed analysis of that component for all the objectives of DO-278.

- Section 7 provides a conclusion and identifies promising areas of further research.

- Sections 8 and 9 document the references and the glossary of terms used in this document, respectively.

- Appendix A describes the outline of the demonstration project implemented as a proof-of-concept.

- Appendix B describes the current standard used in evaluating the security of a system (i.e., the common criteria).

- Appendix C presents a case study (within a different domain) in which COTS components are used to meet safety and security requirements.

2. OVERVIEW.

DO-178B [1] provides guidance for software used in airborne systems, while DO-278 [2] provides guidance for flight-related software used in ground-based systems. This report focuses on ground-based COTS systems and, as such, uses DO-278 for software assurance guidance. This report also considers ways and means of effectively using ground-based data and considers the HUMS advisory material (AC-27-1B Change 1 and AC-29-2C Change 1).

Since this report revolves around the use of COTS components, a definition of the term COTS is imperative. As defined in DO-278, COTS software is a term used to encompass a wide range of software that includes [2]:

- Purchased software,
- Nondevelopmental items, and
- Software previously developed without consideration of DO-278 (or DO-178B).

The typical characteristics of COTS software are [2]:

- May or may not be approved through other approval processes

- Partial or no data may be available as evidence of compliance with objectives of the DO-278 (or DO-178B)

The term off-the-shelf in COTS means that it was not developed by the user but was already existing. Rather than write volumes of new code, COTS software components are used as building blocks to be fit together to satisfy the requirements of new systems. A COTS-based system is one that uses one or more COTS components. COTS-based systems have been cited to have multiple advantages including [3]:

- Development cycle time is reduced. The complex, sophisticated software systems typical today would not be practical to build if every function had to be written from scratch for each new system.

- Total development cost, which includes both development and maintenance cost, is reduced.

- The component is often already in extensive use with a large community of satisfied users.

Consequently, software developers build many parts of new systems by adding functionality provided by COTS software components. However, many challenges need to be addressed when using COTS in safety-critical systems including [3]:

- It is often difficult to know what someone else did to ensure that their software component is reliable and will execute as expected when integrated into a new and complete system.

- The user has limited or no access to source code or other software life cycle data.

- The user usually has no control over the requirements and updates of the COTS.

- There is a learning curve associated with the use of COTS.

- COTS component developer may not have given any attention to safety-critical aspects.

A paper developed by the international Certification Authorities Software Team (CAST) also documents several concerns when using a COTS operating system (OS) in safety-related systems, when the COTS OS does not have supporting DO-178B software life cycle data. These concerns include (not an exhaustive list, since each project will have its own specific issues) [4]:

- Integrity of COTS OS design and implementation is unknown to the user or integrator.

- Unknown functionality and side effects in the COTS OS may exist.

- The COTS OS can negatively affect the operation of other software applications that are executing using the functions of the OS and COTS resource management hardware and software.

- Mitigation approaches to address the COTS OS's potential effects are sometimes themselves implemented in the COTS OS's environment.

- It may not be technically feasible to establish completion criteria for identifying and addressing all the failure condition classifications of functions that could be affected by the COTS OS.

- Verification of correctness and completeness of functionality and mitigation schemes external to the COTS OS is difficult.

- Errors may exist in the COTS OS that are unknown to the applicant or developer.

- Patches to the COTS OS could have effects on the safe implementation.

- The configuration control, problem report resolution, and continued operational safety of any application containing COTS software requires special attention.

- It is difficult to satisfy the objectives of DO-178B/ED-12B (or other certification authority approved guidance) for all software in the airborne application when COTS software is used.

- COTS products might be susceptible to viruses.

- The support provided by COTS software suppliers may not be compatible with the long-life needs of safety-critical systems (e.g., aircraft).

As can be seen from the above discussion, the same characteristics that make COTS an attractive alternative to in-house development may cause serious safety and security concerns. Nevertheless, the potential gains of using COTS should not be ignored. By following guidelines governing the use of COTS, systems can be built that have a high degree of reliability.

3. DESCRIPTION OF THE SYSTEM UNDER STUDY.

For the purpose of this report, a generic HUMS has been selected as a representative ground-based system. A HUMS attempts to predict appropriate times to change parts based on actual use of the parts, as opposed to the more traditional technique of changing parts after a specific number of operating hours. A HUMS can be defined in several ways. One way is based on what it does as identified by the following characteristics [5]:

- Health. A measure of the overall flightworthiness of the aircraft. Health is assessed by examining instantaneous indicators of the well-being of vital components on the aircraft, as well as by trend analysis of these indicators.

- Usage. A measure of how the life of components is being expended on the life-limited parts of the aircraft. Usage determines the time to overhaul the major components.

- Monitoring. A means by which information can be gathered on the vital systems of the aircraft.

The HUMS AC definition of HUMS separates the health and monitoring systems as follows [6 and 7]:

- Health Monitoring System. Equipment, techniques, and/or procedures by which selected incipient failure or degradation can be determined.

- Usage Monitoring System. Equipment, techniques, and/or procedures by which selected aspects of service history can be determined.

Although the general role of HUMS is widely accepted, little consensus exists on what exactly constitutes a HUMS. References 6 and 7 require that a HUMS consist of a variety of sensors and data acquisition systems onboard the rotorcraft or on a ground station (or a combination of both). Other texts only require the HUMS to contain the ground-based processing system. References 5 and 7 provide detailed descriptions of the HUMS concept and specific implementations of the concept. Because of this lack of consensus, this report's view of a HUMS needs to be stated. For the purpose of this report, a HUMS is a system which includes the following:

- Avionics equipment and data.
- Associated ground equipment and data.

A typical high-level architecture diagram of a HUMS is shown in figure 1.

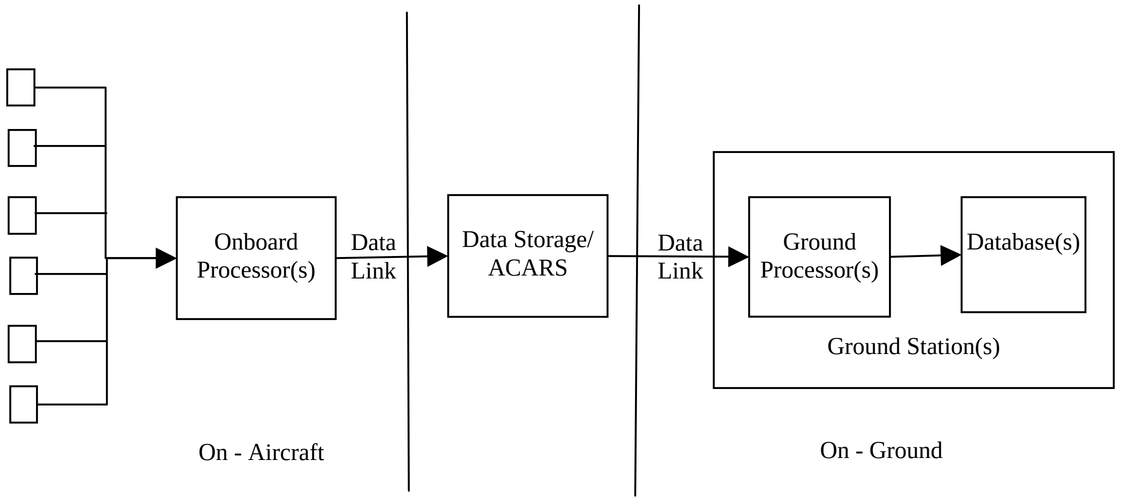


FIGURE 1. TOP-LEVEL ARCHITECTURE OF HUMS [8]

The HUMS can be thought of as three distinct parts, as depicted in figure 1. The aircraft contains a large number of onboard sensors that feed data to the onboard processor(s). The processor(s) then transmit this data through a channel, such as the Aircraft Communication Addressing and Reporting System (ACARS) to the ground stations. The ACARS channel may also have some data storage with the actual data transfer to ground stations occurring offline at a later stage. The ground stations receive data and store it for subsequent inspection and use.

To ensure that data integrity is maintained, one must investigate the safety and security issues in the HUMS. Figure 1 shows that data can be compromised at two points, during transmission through the ACARS and during storage in the databases on the ground. Discussions about these two areas of vulnerability follow.

The ACARS is the data link used by most commercial aircrafts today for two-way communication between the aircraft and ground control. It transmits data in text format over available radio frequency channels. Anyone can intercept these transmissions and gain information such as aircraft type, condition, position, projected track, cargo content, and operational details [9] from the transmission. There are three primary concerns with using such an unsecured network [10]:

- Privacy concerns. Anyone within range and having a very high frequency (VHF) or a high-frequency (HF) device can intercept ACARS signals. This raises an issue of privacy of the health and usage signals transmitted from the aircraft to the ground stations.

- Authenticity. Assurance cannot be obtained that the transmission is coming from the aircraft/authorized ground station. That is, the origin of the message cannot be guaranteed.

- Data Integrity. The integrity of the data during transmission cannot be guaranteed.

However, other nonsecurity related concerns exist as well:

- Saturation of radio frequencies. Since only a limited bandwidth is available with an ever-increasing number of flights, the number of new frequencies that can be allocated is very limited.

- Decrease in quality of data transmission. Because of the exponential increase in flight demands, the available bandwidth is continually shrinking, resulting in poor quality of data transmissions to and from ground control staff.

The ground-based processing systems are points where the data is vulnerable as well. The primary concerns and some techniques for consideration for data integrity during storage are:

- Access security. Only authorized personnel should be able to access the HUMS data. Sophisticated schemes (such as firewalls and encryption) need to be considered.

- Authentication and access control. Simple identification/password mechanisms do not provide the level of security desired. Modern techniques such as digital signatures should be adopted. Also, different users should be assigned different privilege levels to prevent malicious/involuntary corruption of data by authorized users.

- Intrusion detection. Intrusion detection techniques should be adopted to detect patterns of intrusive behavior (such as denial-of-service (DoS) attacks) and take the necessary corrective action.

- Information verification and accountability. Before the data can be used, it should be ensured that data has not been modified maliciously or due to other considerations such as noise bursts, power surges, and so on. Techniques such as checksum calculation, cyclic redundancy check and message authentication codes should be investigated. The use of audit trails should also be considered as a vehicle to maintain accountability. An audit trail is a series of records of events that take place in a system. The events may be related to the OS, application programs, or users of the system. An independent examination of these records can ensure compliance with established controls, policy, and operational procedures [11]. Audit trails thus enable a user to verify the activity of an information system and can help trace violations of security policies back to individuals.

4. OVERVIEW OF AVAILABLE GUIDANCE RELATING TO COTS AND GROUND-BASED SYSTEMS.

4.1 COVERAGE OF COTS IN DO-278.

The Federal Aviation Administration (FAA) provides guidance to developers of airborne software systems so that those systems meet the regulations and are safe and execute as expected when integrated into a new and complete system. The implementation phase of airborne software development is typically manifested by tight control of the entire development effort. Most aspects of the development of a software system (design, configuration control, quality assurance, life cycle management, etc.) are under the control of the developer. Therefore, the developer of a new airborne system, and the FAA, can have confidence that if the guidelines are followed during the development of the new system, then it will perform its intended functionality with a level of assurance that complies with airworthiness requirements. The software assurance guidelines are published as RTCA DO-178B.

In 2002, RTCA Inc. published guidelines for developers of ground-based systems. The ground-based systems are often CNS/ATM systems and typically include COTS software components. These guidelines are published as DO-278, Guidelines for Communication, Navigation, Surveillance, and Air Traffic Management (CNS/ATM) Systems Software Integrity Assurance [1]. DO-278 is based on DO-178B, and for software that is being written line by line, the guidance is very similar to DO-178B. DO-278 also addresses the incorporation of COTS software components, and Section 4 of DO-278 is devoted mainly to software development with COTS software components.

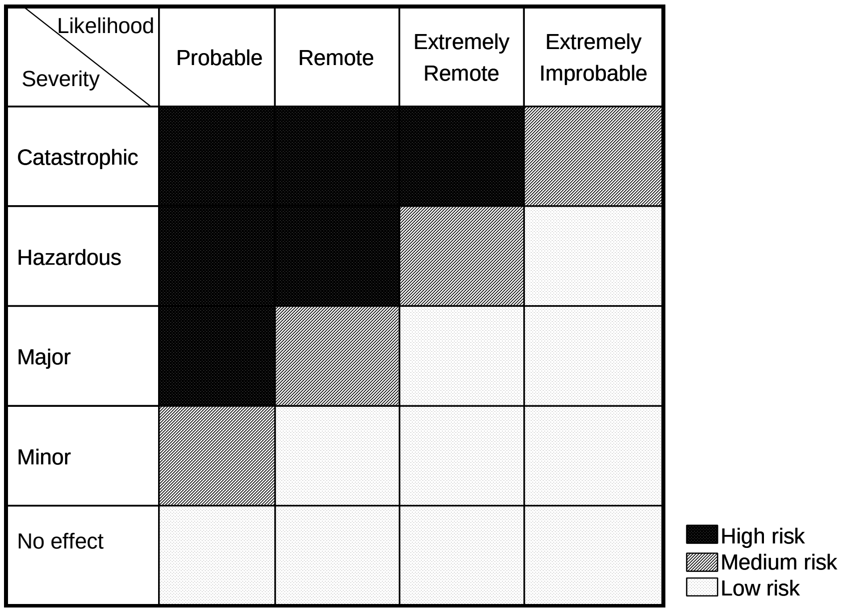
Section 4 of DO-278 specifically addresses the use of COTS software and addresses four processes (planning, acquisition, verification, and configuration management) that must be considered when COTS software is used. The extensive discussion of COTS software is new to DO-278 and was not included in DO-178B. Section 3 of DO-278 contains ten tables of objectives derived from ten tables of objectives in DO-178B. Some of the objectives in these ten tables are COTS-related. The COTS-specific objectives in Section 4 of DO-278 are an addition to, rather than a replacement of, the objectives in section 3.

This project analyzes all of the DO-278 COTS-related objectives from the perspective of a representative COTS software component in a representative ground-based system (see section 6 of this report).

4.2 THE FAA GROUND-BASED, COTS-RELATED PROCESS.

Many analyses are needed for an effective assessment of a ground-based system. Some of these include Operational Safety Assessment, Comparative Safety Assessment, Preliminary Hazard Analysis, Subsystem Hazard Analysis, System Hazard Analysis, Operating and Support Hazard Analysis, Test Safety Analysis, System Safety Program Plan, and a safety assessment. Some of these analyses and documents are used to accommodate FAA Order 8040.4, which states that the FAA should use a formal, disciplined, and documented decision-making process to address safety risks in relation to high consequence decisions impacting the complete product life cycle. The products resulting from this order are System Safety Handbook [12] and System Safety Management Program [13]. Precise definition of the assessment process of a ground-based system at this time is difficult because some of the acquisition process and guidelines are still undergoing change at this time with the National Airspace System (NAS) Modernization Program being the driving force.

The FAA System Safety Handbook for ground-based systems defines system safety as "aiming to optimize safety by the identification of safety-related risks, eliminating or controlling them by design and/or procedures, based on acceptable system safety precedence" [12]. System safety is risk management. At one end of the spectrum, it is not acceptable to have air transport systems that would generally be considered dangerous (catastrophic failures occurring at frequent intervals). At the other end of the spectrum, it is not practical or cost-effective to build systems that are 100% reliable (no failures at all, not even very minor failures at very infrequent intervals). Therefore, practicality and cost effectiveness lie in the middle of the spectrum, balancing the likelihood that a failure will occur, with the consequences if that failure actually does occur. Consequently, in reference 14, Stroup, et al. have proposed a Common Risk Index, which is a matrix that can be used to determine a qualitative measurement (high, medium, or low) of potential system risk by combining the severity of consequences with the likelihood of occurrence. For example, a hazard whose likelihood is classified as extremely remote, and severity as catastrophic is classified as having a high risk. The matrix can then be mapped onto the assurance levels specified in DO-278, thereby helping developers in defining an effective risk mitigation strategy. Section 5.3 of this report will describe how the likelihood and severity information obtained from the mapping between this matrix and assurance levels of DO-278 software can be used in the generation of a risk mitigation strategy. The matrix in figure 2 illustrates the proposed Common Risk Index. The definitions of likelihood and severity terms are given in tables 1 and 2.



Severity \ Likelihood	Probable	Remote	Extremely Remote	Extremely Improbable
Catastrophic	■	■	■	▨
Hazardous	■	■	▨	□
Major	■	▨	□	□
Minor	▨	□	□	□
No effect	□	□	□	□

■ High risk
▨ Medium risk
□ Low risk

FIGURE 2. PROPOSED COMMON RISK INDEX [13]

TABLE 1. LIKELIHOOD DEFINITIONS [13]

Likelihood	Definitions
Probable	Qualitative: Anticipated to occur one or more times during the entire system/operational life of an item. Quantitative: Probability of occurrence per operational hour is equal to or greater than 1 x 10^{-5}
Remote	Qualitative: Unlikely to occur to each item during its total life. May occur several times in the life of an entire system or fleet. Quantitative: Probability of occurrence per operational hour is less than 1 x 10^{-5}, but greater than 1 x 10^{-7}
Extremely remote	Qualitative: Not anticipated to occur to each item during its total life. May occur a few times in the life of an entire system or fleet. Quantitative: Probability of occurrence per operational hour is less than 1 x 10^{-7} but greater than 1 x 10^{-9}
Extremely improbable	Qualitative: So unlikely that it is not anticipated to occur during the entire operational life of an entire system or fleet. Quantitative: Probability of occurrence per operational hour is less than 1 x 10^{-9}

TABLE 2. SEVERITY DEFINITIONS

Severity	Definitions
Catastrophic	Safety: Unacceptable results in fatalities and/or system loss. Security: Loss of mission capability for extended period.
Hazardous	Safety: Large reduction in safety margin or functional capability. Security: Consistent severe impairment of mission capability.
Major	Safety: Significant reduction in safety margin or functional capability. Security: Noticeable impact on security.
Minor	Safety: Slight reduction in safety margin or functional capability. Security: No noticeable impact.
No Effect	Safety: No effect on safety. Security: No effect on security.

Section 5.3 investigates how the above matrix can be used in developing an effective risk mitigation strategy.

4.3 AIRBORNE SOFTWARE ASSURANCE LEVELS.

All airborne software could potentially have an impact on safety. Hence, it is subject to certification requirements as documented in Title 14 Code of Federal Regulations (CFR) [15]. DO-178B defines a set of objectives to establish assurance that airborne software has the integrity needed for use in safety-related applications [2]. DO-178B is implemented by AC 20-115B. AC 20-115B considers DO-178B to be a means, but not the only means, to secure FAA approval of digital computer software. The guidance of DO-178B is provided in the form of objectives for life cycle processes (this is defined according to software levels A through E), activities to be performed to achieve these objectives, and evidence indicating that these objectives have been met [16]. DO-178B acts as a guideline for determining that an acceptable level of confidence is present in the software aspects of airborne systems (in accordance with FAA airworthiness requirements). The primary part of the DO-178B definition of the different software level is as follows [1]:

- Level A. Software that could cause or contribute to the failure of the system resulting in a catastrophic failure condition.
- Level B. Software that could cause or contribute to the system resulting in a hazardous or severe failure condition.
- Level C. Software that could cause or contribute to the system resulting in a major failure condition.
- Level D. Software that could cause or contribute to the system resulting in a minor failure condition.
- Level E. Software that could cause or contribute to the system resulting in no effect on the system.

4.4 NONAIRBORNE SOFTWARE ASSURANCE LEVELS.

The levels in DO-278 are based on and relevant to DO-178B. In DO-278, these levels are called assurance levels and, when applied to a non-airborne-based system, are represented by the following [2]:

- Assurance Level 1 (AL1). Software that could cause or contribute to the failure of the ground-based system resulting in a catastrophic failure condition. Equivalent to DO-178B Level A (as mentioned in section 4.3 of this report).

- Assurance Level 2 (AL2). Software that could cause or contribute to the failure of the ground-based system resulting in a hazardous or severe failure condition. Equivalent to DO-178B Level B (as mentioned in section 4.3 of this report).

- Assurance Level 3 (AL3). Software that could cause or contribute to the failure of the ground-based system resulting in a major failure condition. Equivalent to DO-178B Level C (as mentioned in section 4.3 of this report).

- Assurance Level 4 (AL4). This level accounts for certain CNS/ATM (ground-based) systems where AL3 is too stringent and AL5 is too lenient. There is no DO-178B equivalent for this assurance level.

- Assurance Level 5 (AL5). Software that could cause or contribute to the failure of the ground-based system resulting in a minor failure condition. Equivalent to DO-178B Level D (as mentioned in section 4.3 of this report).

- Assurance Level 6 (AL6). Software that could cause or contribute to the failure of the ground-based system resulting in no effect on the system. Equivalent to DO-178B Level E (as mentioned in section 4.3 of this report).

The DO-278 assurance levels are correlated to risk, as illustrated by an enhanced Common Risk Index in figure 3.

The assigned assurance levels will depend on the function to be performed by the software and the impact of that function's failure. The specific process applied to the software will vary, depending on whether it is a COTS component or not.

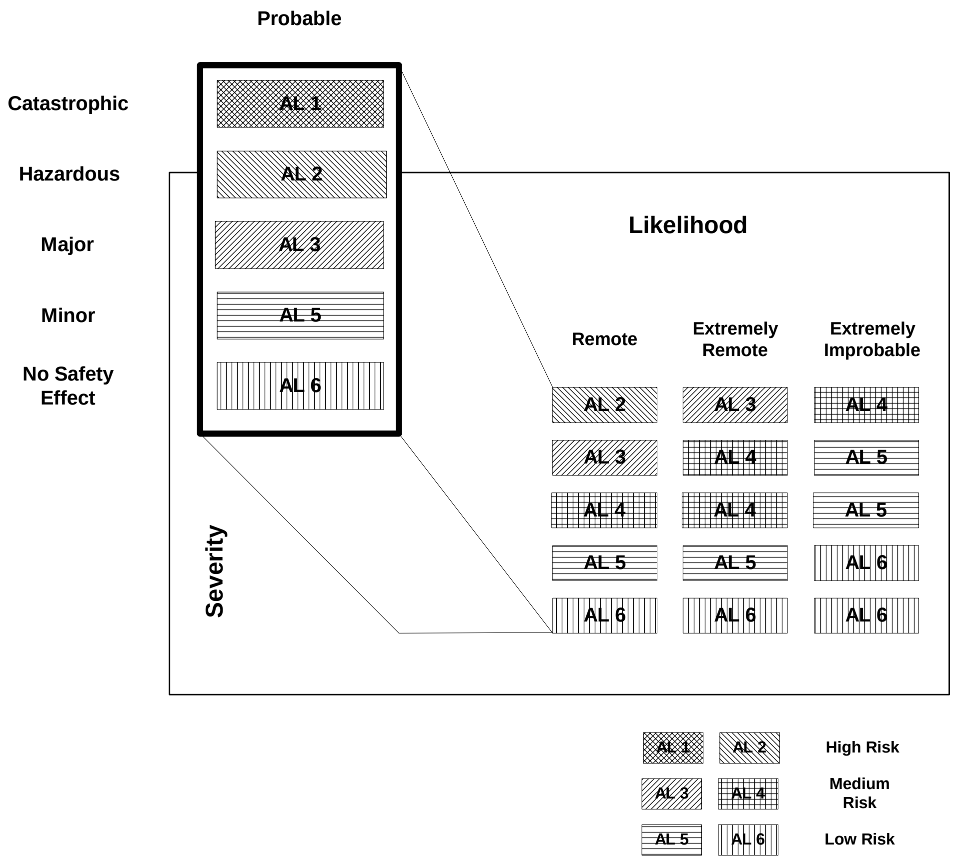


FIGURE 3. ENHANCED COMMON RISK INDEX [17]

4.5 CONSIDERATIONS OF THE HUMS ACs.

This section will review the current guidance listed in the Rotorcraft HUMS ACs' guidance material found in AC-27-1B Change 1 and AC-29-2C Change 1 for certifying HUMS ground-based systems that contain COTS components. The specific guidance of concern is found in Chapter 3 and Miscellaneous Guidance Section 15 (MG 15) of the HUMS ACs. This section will also consider other potential guidance such as DO-178B and DO-278B.

4.5.1 Background of HUMS.

Health and usage monitoring has become a focal point in the helicopter domain. Anticipated benefits include improved aircraft safety, increased availability and reliability, rapid determination of aircraft status, reduced overhaul and repair costs, and reduced scheduled component removal [18]. Many helicopter parts are dynamically loaded and are subject to develop fatigue cracks. As a result, many of these parts are life-limited and are assigned a safe-life limit in flight hours based on Miner's theory of linear damage rule. The process is to determine the aircraft's usage or flight maneuvers (flight regimes), the frequency of those maneuvers, or the percent usage of that regime in a particular time frame. This information is combined with material strength characteristics and part-testing results. These loads are then used to establish an acceptable calculated retirement time for these parts [19]. The development of a HUMS is rather unique in that the application software, perhaps level A, that is developed under DO-178B could be integrated with COTS software that may require a level D (or even higher) software assurance level, depending on the functional hazard assessment (FHA).

Recent efforts have explored substituting the assumed worst-case usage with the actual measured usage to calculate retirement times. This would permit safe part-life to be extended if the actual measured usage is less than then assumed usage. It would also permit early removal of parts to maintain safety margins if the actual usage was more severe than assumed. In reference 19, a design assessment is carried out for an application taking such an approach. The application under study in reference 19 was a Helicopter Structural Usage Monitor. The usage monitoring effort acquires helicopter sensor data, which is used to determine the aircraft's flight regime. The regimes are calculated in real time once per second and stored in a data file that is used on landing to calculate a damage fraction for the part. This data is packed and sealed with a cyclic redundancy check (CRC) algorithm. The damage data is then moved to a ground-based computer [19].

A business case point of view in the HUMS domain is presented in reference 8, but it is cited that in June 1999, at the time the business case was developed, the HUMS ACs (AC 27-1B Change 1 and AC 29-2C Change 1) were in draft form, with the intent to provide specific guidance for FAA approval of functions implemented on a PC-based ground station. Because of the guidance being in draft form, several HUMS applications chose not to include COTS software as part of their HUMS solution. As a result, there is little current data as to the effectiveness of the AC guidance related to COTS. However, since the official release of the HUMS ACs guidance material, to date, no HUMS applications for rotorcraft have been approved for maintenance credits. That is, the HUMS applications that were approved by the FAA for rotorcraft to date, do not replace the existing maintenance program requirements. These HUMS applications have been approved on a no maintenance credit basis. This means that maintenance actions are not predicated on the data collected and processed by the approved HUMS applications.

4.5.2 The HUMS Advisory Circulars.

Advisory circular material AC 27-1B and AC 29-1C for airworthiness approval of HUMS was developed by the Rotorcraft Health and Usage Monitoring System Advisory Guidance, which consists of the FAA, the Joint Aviation Authorities (JAA), the Aerospace Industries Association of America Inc. (AIA), and the Association Europeene des Constructeurs de Material Aerospatial [6 and 7].

The advisory circulars provide guidance to achieve airworthiness approval for three basic aspects to HUMS certification:

- Rotorcraft HUMS installation
- Credit validation
- Instructions for Continued Airworthiness (ICA)

Certification of HUMS must address all three aspects. These aspects are not independent of each other and have varying degrees of interaction among themselves. Installation includes all the equipment needed for the end-to-end application that is associated with acquiring, storing, processing, and displaying the HUMS application data, including airborne and ground-based equipment. Credit validation includes evidence of effectiveness for the developed algorithms, acceptance limits, trend-setting data, tests, etc., and for the demonstration methods employed. ICA includes the methods used to ensure continued airworthiness of those parts that could change with time or use.

4.5.3 Commercial Off-The-Shelf Approaches in the HUMS ACs.

AC 27-1B Change 1 and AC 29-2C Change 2 (referred to as HUMS ACs for the remainder of this report) have very specific terms when discussing the various approaches to using COTS in a HUMS. The HUMS ACs explain the following terms:

- COTS. This term defines equipment, hardware, and software that is not qualified to aircraft standards.

- Mitigating Action. It is an autonomous and continuing compensating factor that may modify the level of qualification associated with certification of a HUMS application. Mitigating actions are often performed as part of continued airworthiness considerations and are also an integral part of the certification.

- Independent Verification Means. An independent process to verify the correct functionality of a HUMS application on a ground station that uses COTS. The intent of independent verification is to gain some degree of confidence in the COTS operation reliability. Note: This process may be discontinued when sufficient confidence in the application has been achieved.

The COTS definition in the HUMS ACs is very different than traditional COTS definitions. The HUMS ACs state that COTS can be any software or hardware that has not been developed to DO-178B (or DO-254) standards. This definition means that for the FAA to approve a HUMS, the COTS software and hardware must both be developed and qualified to acceptable aircraft standards. However, the group that developed the HUMS ACs recognized that many existing ground stations used COTS components, like PCs and commercial operating systems, to process maintenance data collected through traditional means (i.e., manual recording of component life usage by humans). Hence, the developers of the HUMS ACs wanted to continue to allow this use of COTS components, but there needed to be some acceptable criteria that these COTS components needed to be evaluated against. The HUMS ACs do not relieve the HUMS application software from having to comply with DO-178B, even if this software resides in ground-based COTS station. An example of the HUMS application software is the software that is developed to recognize the various flight regimes and processes raw data to determine the extent of wear and tear on a specific aircraft part. This HUMS application software may reside in the ground-based COTS station.

DO-178B defines COTS in a different way. The DO-178B glossary defines COTS software as "Commercially available applications sold by vendors through public catalog listings" [1]. This definition states nothing about the pedigree of the COTS product relative to DO-178B. In fact, if one were to have a commercially available product that has been developed to the guidance of DO-178B, then it would be COTS under DO-178B, but not under the HUMS ACs. Correspondingly, if one were to have an in-house developed software module not developed to DO-178B, then the HUMS ACs state it is COTS, but under DO-178B it would not be COTS because it was not commercially available.

In the HUMS ACs, a mitigating action is very different from independent verification means (IVM), and they should not be confused. Mitigating action can be viewed as a compensation factor applied to the HUMS, which can result in an allowance in the reduction of the required level of qualification for the HUMS hardware and software. The compensating factor applied as a mitigating action is permanent. The IVM addresses the need to independently verify the results of COTS software or hardware components that have not been developed to DO-178B (or DO-254 for complex electronic hardware). The intent of the IVM process is to gain some degree of confidence in the operational reliability of the ground-based COTS station. This process may be discontinued when sufficient confidence in the application has been achieved.

<u>4.5.3.1 The HUMS ACs Intent and Other Considerations</u>.

Like airborne systems, the HUMS system must determine its end-to-end criticality by performing a FHA, including the ground components. From a hardware point of view, ground components can include not only a PC desktop, laptop, and server, but also subassembly components such as modems, Ethernet cards, or memory transfer media (such as memory sticks). On the software side, not only does the system under review need to consider the computing engine software (such as databases or spreadsheet software), but also the enabling software (such as communications driver software and the OS). Any potential effect on the data must be considered in the FHA. The HUMS ACs' intent is to have the FHA identify all system components, including COTS components, and determine the component's criticality level to find its corresponding software level. Any COTS component performing such actions (i.e., that determines component life time) must come under the guidance of DO-178B or an acceptable alternative.

Compliance to the determined FHA criticality level can be accomplished by qualification plus appropriate mitigating action. The final level of equipment qualification may not only be the result of technical consideration, but also of other mitigating actions, which can result in a reduction of qualification levels for equipment.

<u>4.5.3.1.1 The HUMS ACs and Service History</u>.

The HUMS ACs further state that for COTS components, the IVM must be accompanied with satisfactory service history. Recent service history findings have shown, however, that for higher criticality systems, service history may simply not be an appropriate or practical choice. Care must be taken when using service history since it is usually part of a system. Any use of service history for a COTS component is typically an extrapolation of the system's performance. This extrapolation must be justified [20]. Ferrell [20] reported at the FAA 2003 National

Software Conference that when trying to apply an effective service history argument to a component of a system, problematic data collection practices, problematic interpretations, and associated assumptions are moving targets. Such problems provide a challenge to effectively quantifying and justifying the data used to support the service history case.

Given Ferrell's [20] findings with regards to the difficulty of establishing a sound service history base, at least for the higher criticality systems, the HUMS AC statement of "satisfactory service history" would be very difficult to argue for.

Take, for example, the use of Miner's Cumulative Damage theory. Historically, the approach has been validated via service history. With the HUMS proposition of recalculating Miner's laws using data from actual usage, an effective database of service history is lacking for validation of any new directed action with regards to the part's continued performance ability.

4.5.3.1.2 The Required Guidance Objectives for Air- and Ground-Based Software.

The service history discussion above is one of several related to the state of accepting COTS components. The more important issue or question to debate is, given that DO-178B was developed for airborne systems, What are the safety attributes of COTS components that are now ground-based?

DO-178B requires structural coverage to identify any code not exercised through tests (so that any potential unintended functionality can be identified and addressed). DO-248B, section 4.2, provides a glimpse of the source of this requirement back to 14 CFR. 14 CFR 25.1309, Subpart F – Equipment, states that the equipment, systems, and installations "must be designed to ensure that they perform their intended function under any foreseeable operating conditions." It has been observed that the requirement for structural coverage on ground-based systems may financially preclude the use of COTS. Yet these COTS components may provide many benefits. Perhaps a reassessment of the structural coverage requirements for ground-based systems should be considered.

In June 2002, the CAST completed a position on an Automatic Code Generation (ACG) Tools [21]. The CAST paper recognizes that halts during execution, overflows, variations in time response, hardware and software incompatibilities, hardware failures, unbounded recursive algorithms, bad stack usage, resource contention, task conflicts, bad interaction with other systems, etc., are examples of issues that may jeopardize flight safety if they appear in aviation software. However, these types of errors may not have any influence on the flight safety if they occurred within a non-airborne-automated, code-generation tool.

The CAST paper continues to modify the basis of compliance for such a tool by considering two entities of many ACGs:

- Entity 1. The library of elementary symbols (e.g., code primitives or basic functions) that contain basic symbols that contain the source code associated with implementing the function of each elementary symbol.

- Entity 2. The architect (e.g., tool logic and program constructor) that reads the software specification and selects the association to the elementary symbol and may insert the corresponding source code for each selected symbol.

In the CAST paper, each entity has different applications of DO-178B-related guidance. A similar approach could be taken for software components in a HUMS. Similarly, two entities should be considered for HUMS ground-based COTS components: (1) COTS components not directly related to the manifestation of vehicle part action (discussed in the following section) and (2) COTS components directly related to such action (discussed in following section). For example, a COTS Visual Basic (VB) program written in an Excel spreadsheet that implements vehicle part action algorithms should have a different scope of verification than a COTS OS.

<u>4.5.3.1.3 Commercial Off-The-Shelf Not Related to Vehicle Part Action</u>.

Many COTS software components exist that do not have part-life, decision-making authority. Examples are OSs and communication software. Certainly, this class of COTS software is simply enabling software that permits the HUMS data to be operated on by other software. Section 6 provides a brief look at the objectives of DO-178B to determine which are pertinent.

<u>4.5.3.1.4 Commercial Off-The-Shelf Directly Related to Vehicle Part Action</u>.

A focus for ground-based COTS components in a HUMS is correctness and accuracy for the predicted vehicle part action. Compiler libraries, database software, and other computationally intensive COTS software must have many more applicable DO-178B objectives than not-directly-related COTS components. Arguably, one could state that all objectives must be applied according to a criticality level as determined by the FHA. But, as noted with the publication of DO-278, not all ground-based systems align well with the objectives for assurance levels. Hence, a new assurance level was created (AL4).

In fact, in many areas of ground-based software operations, the accountability of these objectives possibly should be on a case-by-case basis. A preferred method of determining the applicability of the DO-178B or DO-278 objectives may be to perform a detailed analysis of each system component for all the objectives of these guidance documents and provide a rationale for why or why not an objective needs to be met. A deeper research investigation into this technique is performed later in this document on two components of the demonstration portion of this study (see section 6).

<u>4.5.3.1.5 Independent Verification Means</u>.

Multiple-version redundancy has been offered in the HUMS ACs as a means for independent verification of ground-based COTS hardware and software equipment. The use of multiple versions for fault tolerance has been discussed in the past. Leveson [22] states that multiple-version redundancy has been shown to have no statistical basis and cites six separate sources supporting this claim. The issue of independence and redundancy of software needs more investigation.

4.5.3.2 Alternate Approaches for Acceptance of COTS Software.

Certainly, the guidance in DO-178B is a recognized means of compliance. DO-278, discussed elsewhere in this document, provides additional guidance recommendations with respect to COTS software. DO-278 also has specific guidance on COTS that goes beyond the scope of DO-178B.

DO-248B attempts to clarify portions of DO-178B including several discussions on COTS acceptability, but provides no new or additional guidance material beyond DO-178B. DO-178B is highly process-based, yet a trend appears to be leaning toward more product-based analysis. When one considers permitting modular certification or approval of components as Rushby suggests in reference 23, alternate approval methods will be needed. A variety of approaches can aid in the determination of the acceptability of a COTS component. An area of future study for this research would be to assess alternate means, as listed below for each objective, and determine the viability of making such an analysis process part of the suggested guidance of DO-178B or DO-278.

- Wrappers
- Calculation Reasonable Checks
- Data Input Reasonable Checks
- Full Up Part-life Models
- Partitioning
- Isolation
- Product Service Experience
- Prior Assurance
- Process Recognition
- Reverse Engineering
- Restriction of Functionality
- Audits
- Inspections
- Formal Methods
- Fault Detection and Accommodation
- Monitoring—Performance, Safety, Activity
- Fail-safe Architectures
- Adaptive Part Model Predictions—recent confidence index

5. CURRENT AND EMERGING INDUSTRY APPROACHES TO SECURITY AND SAFETY.

This section considers some of the current security and safety trends for COTS software and their relationship to HUMS.

5.1 SECURITY APPROACHES.

System security is closely related to system safety; both deal with threats or risks to the system and both involve protection against losses, although the types of losses involved may be different. Faults inherent to the system can be removed over a period of time in system testing, but system security is a factor that needs to be considered even when the system is in a functional stage, as negligence in security may compromise system safety.

Security focuses on malicious actions of outside or inside attackers. System security is different from safety in that it protects the system from the deliberate attacks of malicious attackers, while system safety deals with protecting the system in case of faults or failures that are unintentional (i.e., not deliberate). The primary emphasis in security has been on preventing unauthorized access to classified information. However, it also includes unauthorized disclosure, modification, and withholding of data.

The following sections introduce various security mechanisms (i.e., cryptography, access security, intrusion detection, information and data protection) and consider their potential use in the HUMS context.

5.1.1 Cryptography.

Cryptography is the art of creating and using cryptosystems. A cryptosystem or cipher system is a method of disguising messages so that only certain people (the intended recipients of the message) can see through the disguise. The original message that is disguised is called plaintext. The disguised message is called ciphertext. Encryption refers to the method used to convert plaintext into ciphertext. Decryption refers to any procedure to convert ciphertext into plaintext. A cryptosystem is usually a whole collection of algorithms. Cryptosystems come in two different versions: secret key cryptosystems and public key cryptosystems. This section describes the basics of cryptography and the applications of cryptography that are relevant to the HUMS.

5.1.1.1 Secret Key Cryptography.

Secret key cryptography is sometimes referred to as symmetric cryptography [24]. In this traditional form of cryptography, the same key is used for encryption and decryption, as shown in figure 4. Message Authentication Codes (MAC) use secret keys to provide authentication.

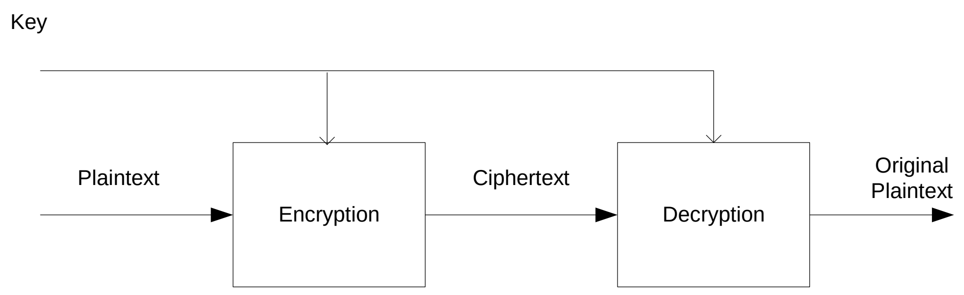


FIGURE 4. SYMMETRIC KEY CRYPTOSYSTEM

Secret key cryptography has certain advantages over public key cryptography, its higher speed being one. However, secret key cryptography requires that the sender and receiver of a message agree on a secret key, and this is often a difficult task due to eavesdropping.

The most common techniques in secret key cryptography are block ciphers, stream ciphers, and MACs. Each is discussed below.

5.1.1.1.1 Block Ciphers.

A block cipher is a type of secret key encryption algorithm that converts a block of unencrypted data of fixed length into a block of encrypted data of the same length. A secret key is used for this transformation. Decryption is performed using the same secret key. The fixed length of unencrypted data (plaintext) and encrypted data (ciphertext) is termed the block size.

5.1.1.1.2 Stream Ciphers.

A stream cipher is a type of secret key encryption algorithm. As opposed to block ciphers that operate on blocks of plaintext, stream ciphers transform smaller units of unencrypted data (generally bits into the ciphertext).

A stream cipher generates a key stream (i.e., a sequence of bits used as a key). Encryption generally involves combing the key stream and the plaintext, which is quite commonly a bitwise XOR operation.

5.1.1.1.3 Message Authentication Code.

A MAC involves the use of an authentication tag generated by applying a secret key and an authentication scheme to a plaintext message. Unlike digital signatures, MACs cannot be verified by receivers that do not possess the secret key used to create the MAC. As shown in figure 5, a sender uses G (key, message) to generate a tag. This tag is appended to the message and is transmitted. If an attacker succeeds in intercepting the message, he or she modifies the message and tag. The modified message and tag then reach the receiver, who uses G (message′, key) to determine whether or not the contents have been modified.

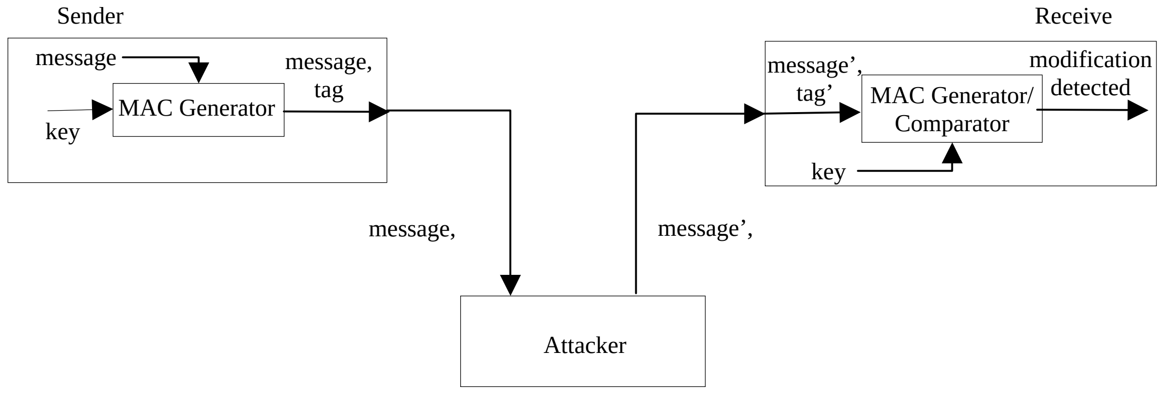


FIGURE 5. MESSAGE AUTHENTICATION CODE

5.1.1.2 Public Key Cryptosystems.

In secret key cryptography, the sender and receiver of a message use the same secret key; the sender uses the secret key to encrypt the message, and the receiver uses the same secret key to decrypt the message.

A key exchange scheme is required for the sender and receiver to agree on the same key. If an eavesdropper is able to intercept the key during exchange, all transmissions encrypted using this key will be prone to passive attacks on the confidentiality of the data.

To deal with key management issues, Whitfield Diffie and Martin Hellman introduced the concept of public key cryptography in 1976. Public key cryptosystems have two primary uses, encryption and digital signatures. The system involves the private key and the public key. Private keys are kept secret while public keys may be disclosed to anyone. Public key cryptography can be used not only for privacy (encryption), but also for authentication (digital signatures) and other various techniques. Figure 6 shows a public key cryptosystem that uses different keys for encryption and decryption.

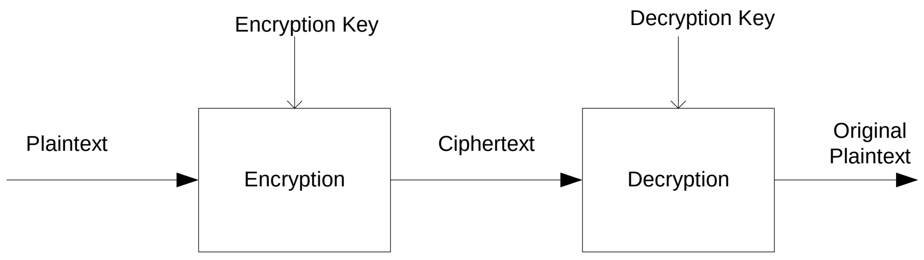


FIGURE 6. PUBLIC KEY CRYPTOSYSTEM

Public and private keys are linked mathematically. Since the public key is not kept secret, an attacker may try to derive the secret private key using the public key. However, in public key cryptosystems, the task of deriving the private key from the public key is made difficult. For instance, some public key cryptosystems like RSA (Rivest, Shamir, Adelman) are designed such that deriving the private key from the public key requires the attacker to factor a large number. This derivation is computationally infeasible. Encryption and digital signatures are further discussed below.

5.1.1.2.1 Encryption.

Assume *A* and *B* to be two entities that wish to communicate. Suppose *A* wishes to send a message. *A* looks up *B*'s public key, encrypts the plaintext with *B*'s public key, and sends the ciphertext to *B*. Confidentiality is achieved, as only *B*, who has the private key, can decrypt the message.

5.1.1.2.2 Digital Signatures.

Again, assume *A* and *B* to be two entities that wish to communicate. To sign a message, *A* does a computation using its private key and the message itself; i.e., *A* encrypts the message using its private key. The encrypted output is called a digital signature and is attached to the message. To verify the signature, *B* does a computation that involves the message, *A*'s signature, and *A*'s public key. Using a simple mathematical relation, *B* decrypts the digital signature using *A*'s public key. If the decrypted result matches the original message, then *A*'s signature is verified to be genuine. Otherwise, the signature is considered fraudulent, or the message may have been modified.

5.1.2 Cryptography in the HUMS Context.

Cryptography should be used in HUMS for achieving security objectives. Different forms of encryption techniques can be used to make ACARS secure, but this requires enhancements to the existing ACARS infrastructure.

If cryptography is used, the secure ACARS runs as an application; hence, it is independent of underlying hardware. The secure ACARS application runs over the existing infrastructure, which interprets the secure transmissions as shown in figure 7. Thus, the secure transmission is invisible to the user.

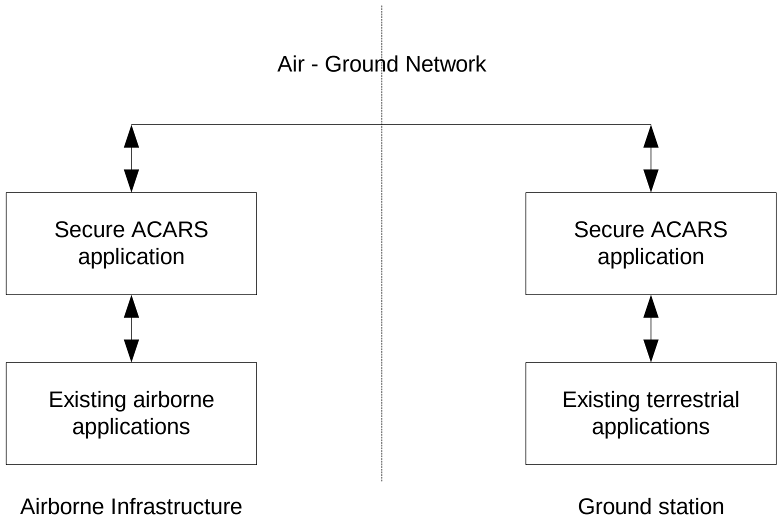


FIGURE 7. SECURE ACARS

The cryptography protocol works as follows, if integrated into HUMS (as shown in figure 7):

- Before data transfer can begin, a secure path must be established. That is, both sender and receiver must be unambiguously identified to each other. This is called the Handshake phase. The airborne system sends its identity, time of transfer, intended ground station, digital signature, etc., in a message to the ground station.

- The ground station, in turn, responds with another message comprised of the identity of the ground station and a random number used to generate a session key. This message is encrypted and can be decrypted only by the private key of the airborne system.

- Using this random number, a session key is established, which is then used to encrypt all data transfers. In this way, the system uses a combination of symmetric (private) as well as asymmetric (public) cryptographic techniques to ensure secure data delivery.

- If a breach is detected at any time (for example, if checksum validations on transmitted data fail or if authentication fails), either party can terminate the session, and transmission can revert back to voice-based communication. Once the failure is resolved, secure ACARS operations may resume.

5.1.3 Access Security.

Every system must support the ability to protect data and system resources from intrusions, modifications, theft, and unauthorized disclosures. As data in any database is related by semantic relationships, damage in a database environment can affect the entire information system.

Categories of security breaches include [25]:

- Unauthorized data observation. Unauthorized data observation results in a disclosure of information to users not entitled to gain access to such information.

- Incorrect data modification. Incorrect modifications of data, either intentional or unintentional, result in an inconsistent database state.

- Data unavailability. When data is unavailable, information crucial for proper functioning may not be readily accessible when needed.

Therefore, a complete solution to the data security problem requires addressing issues of secrecy, integrity, and availability.

Authentication deals with the problem, Who is the user? Access control deals with the problem, What can the user do to a certain resource? Authentication and access control together form an authorization mechanism; hence, the two techniques are discussed together below. For HUMS, all the mechanisms for authentication and access control can be deployed at the gateway, which separates the HUMS database subsystem from other subsystems.

5.1.3.1 Authentication.

An authentication mechanism attempts to provide assurance of the claimed identity of an entity. Whenever any entity tries to access HUMS, that entity should first be authenticated. Only if the entity can be authenticated properly should it be allowed to access and modify the data.

Authentication methods generally rely on one or a combination of the following principles:

- Something known: e.g., password, pin, mother's maiden name.
- Something possessed: e.g., smart card, key, ID badge.
- Some immutable characteristics: e.g., fingerprints, voice scan, retina scan (biometric identifier).

In the case of HUMS, entities that require access to the database are:

- Secure ACARS network. It transfers airborne data from the aircraft to the ground-based gateways in a secure manner. This data needs to be stored in the HUMS database without compromising data security and integrity.
- Applications that require HUMS data for use and maintenance purposes. The applications may access data over a secure/private network connection, a virtual/private network, or over an unsecured/public network (Internet).

HUMS database subsystems can assume that the encrypted data it receives via the secure ACARS is authenticated. But this encrypted data needs to be checked for its integrity, since it could be tampered with during transmission through the network.

5.1.3.2 Access Control.

Access control mechanisms ensure data secrecy. Whenever an entity tries to access data, the access control mechanism checks the rights of the entity against a set of authorizations. An authorization states which user can perform which action on which object. Data integrity is jointly ensured by an access control mechanism and by semantic integrity constraints. Whenever an entity tries to modify some data, the access control mechanism verifies that the user has the right to modify the data; the semantic integrity subsystem verifies that the updated data is semantically correct (i.e., the system is in a stable logical state).

Access control mechanisms generally rely on authentication mechanisms and encryption techniques for proper functioning. Access control policies can be broadly classified into two categories: Discretionary Access Control and Mandatory Access Control [25]. Each is discussed below.

5.1.3.2.1 Discretionary Access Control.

The discretionary access control policy governs the access of entities to data on the basis of entities' identity and authorization rules. Such mechanisms are discretionary in that they allow entities to grant other entities authorization to access data. An important aspect of discretionary access control is related to authorization administration policy, which refers to the function of granting and revoking authorization. Some common authorization administration policies are listed below [25]:

- Centralized administration. Some privileged users may grant or revoke authorizations.
- Owner-based administration. The creator of a data object issues grant or revoke operations. Due to this characteristic, this category allows decentralized administration.
- Joint-based administration. Several users are jointly responsible for authorization administration.

Authorization of each access hit depends exclusively on the existence or absence of an authorization rule, without taking into account the confidentiality level of the data or the level each entity can access. OSs generally implement these policies.

Thus, a high degree of flexibility characterizes discretionary access control policies. However, these policies do not impose any control on how information is propagated and used after authorized users have accessed information.

5.1.3.2.2 Mandatory Access Control.

The mandatory access control policy classifies entities and data into different security levels. The main characteristic of mandatory access control policy is that an access is authorized if a certain relation exists between the entity security level and the security level of the object to be accessed. Data has appropriate security levels, irrespective of the users who wish to access it. Multilevel database management systems can support mandatory access control through different security levels in the data and different accreditation levels for users.

Mandatory policies are rigid compared to discretionary policies because they require a strict classification of entities and data objects into security levels. However, they ensure a high degree of protection in that they prevent any illegal flow of information.

5.1.3.3 Variations in Access Control Policies.

Different variations of the above-mentioned access control policies exist that are suitable to different applications. Examples of such variations are:

- Negative authorization. Explicit denials can be expressed as authorization rules.
- Temporal duration of authorization. Permissions can be specified for specific time intervals.
- Role-based access control. In this policy, access permissions are associated with roles, and users are made members of the roles. The roles represent each functional group of the system; grouping in each one represents users with similar functions and

responsibilities. Thus, operations that a user can perform are based on the user's role. This policy facilitates administration and is a natural way to represent hierarchies in the system.

5.1.3.4 Authentication and Access Control in HUMS.

An authentication and access control module in HUMS should have the following characteristics [26]:

- High-assurance security. Applications should provide high confidence through strong security services for confidentiality, data integrity, user-based authentication, and nonrepudiation. System assurance should be limited only by the underlying COTS platform and applications. This implies that COTS components should be chosen such that from a security perspective, their limitations are within acceptable limits of the target HUMS application.

- Secure submissions and retrieval. Data must be protected while allowing authorized users to access and modify information.

- High usability. A uniform user interface should be used to increase user acceptance and efficiency while reducing the training costs of high-security solutions.

- Scalability. A secure solution must be extendable to meet ever-increasing throughput and storage requirements.

- COTS products use. Secure solutions should employ COTS-based hardware and software, where possible, to minimize redevelopment, testing, and support efforts, without compromising security.

5.1.4 Access Control in the HUMS Context (An Example).

A firewall is a set of mechanisms that can enforce a network security policy on communication traffic entering or leaving a network policy domain [27]. Current firewall technologies and low-cost, powerful COTS hardware and software computing platforms lend themselves to a solution using standards-based cryptographic algorithms and mechanisms. As discussed in the previous sections, this secure network architecture provides secure data access over trusted or untrusted networks and high-assurance access control in a highly usable, flexible, and scalable manner [26].

The architecture in figure 8 is an example of a HUMS that is built using a commercially available, application-level firewall and other COTS machines. The systems, which use standard OSs, act as gateways between the users and the servers. The firewall mediates access between multiple network domains with trusted HUMS servers. Only administrative user logins are permitted on this trusted server domain. Thus, the architecture provides strong protection for sensitive data by physically separating HUMS servers from other users and servers on the network.

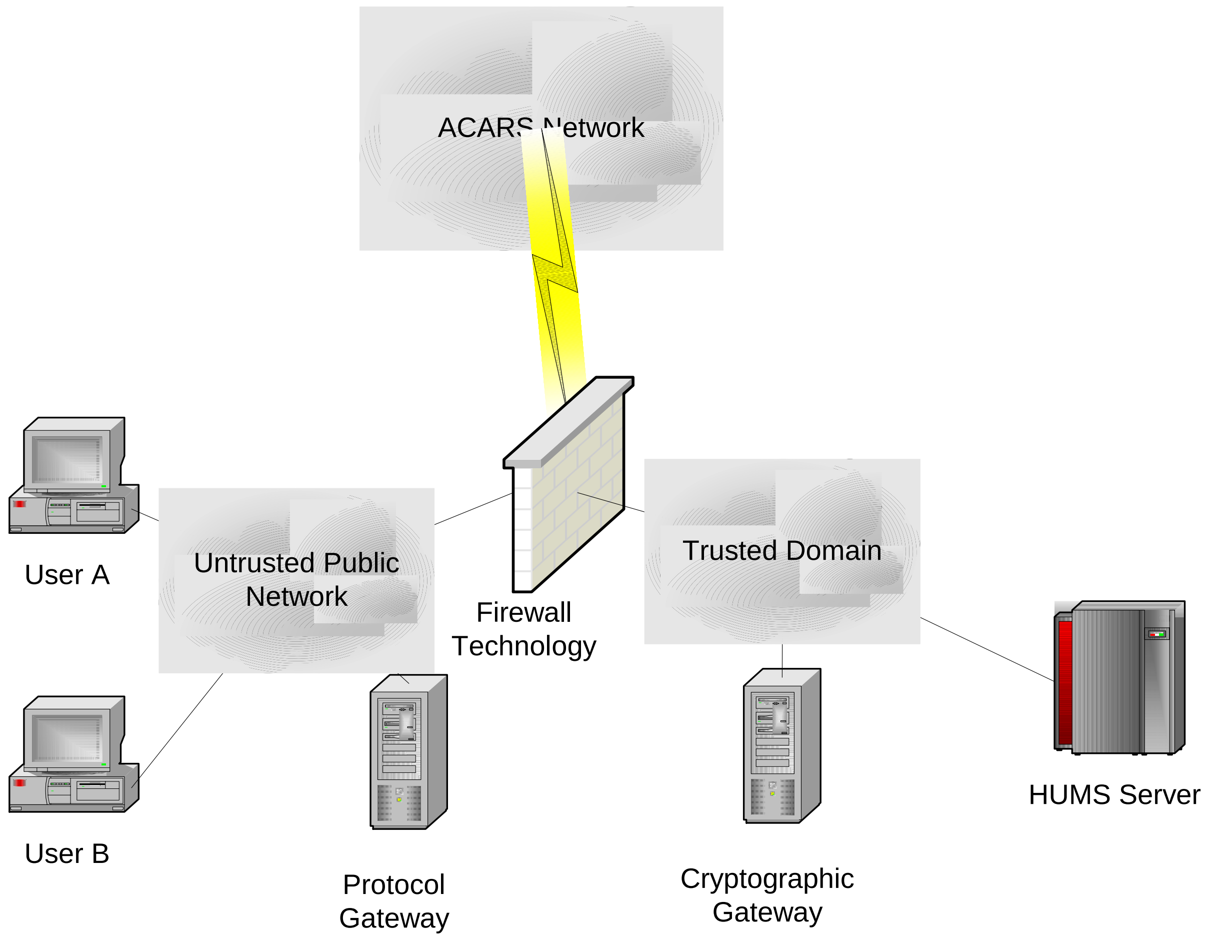


FIGURE 8. SECURE NETWORK ARCHITECTURE [25]

Data flow transmitted over the secure ACARS network can be assumed to be authenticated. This encrypted data consists of actual data sent by onboard sensors and message digests computed on the actual data using standard algorithms. The firewall allows data from the secure ACARS network to be passed to cryptographic gateways. Data is checked for its integrity using message digests. If the message digest computed by the cryptographic gateway is different from the message digest in the data, then the data is discarded. Otherwise, the gateway forwards the data to the appropriate HUMS server.

A protocol gateway, such as a web server, located on an untrusted network domain presents the user interface on client workstations. The user completes a secure data submission or retrieval request and encrypts and digitally signs the request using cryptographic standard algorithms. In addition to or instead of a digital signature, the user may provide other forms of authentication, such as biometrics.

A firewall relays the data submission or retrieval request to the HUMS server within a trusted domain through cryptographic gateways. When the request reaches the trusted domain, the cryptographic gateway verifies the digital signature and/or biometric template and decrypts the transaction. The architecture supports the Public Key Infrastructure mechanism.

After successful authentication, the cryptographic gateway can check the requested action and its originator against its access control list (ACL) and can determine if the user is authorized to perform the requested action. If the authentication fails, or if the user's request cannot be authorized, the cryptographic gateway would reject the request immediately and log this rejected request for subsequent monitoring.

Authenticated requests from authorized users are forwarded to HUMS servers for further processing. The server sends the result of the request to the cryptographic gateway, where it is encrypted for the originating user, digitally signed, and relayed back to the originator.

Network access to the trusted domain is strictly transaction-oriented and must be digitally signed by the individual user making the request. The security policy for this architecture permits no direct network logins to HUMS servers (except for administrative access). This policy is enforced by a firewall technology that mediates access between the different network domains, allowing only transaction-oriented communication addressed to the cryptographic gateway to pass through to trusted server domain. Because administrative logins are also protected by strong authentication, the firewall policy can be used to control the perimeter of the trusted server domain.

Authorizations are performed through the use of ACLs maintained on the cryptographic gateway located on a trusted server domain. ACLs are files that maintain information on protected HUMS resources and the individual users who may access them. Firewalls prevent direct connections to the machines on trusted domains. Also, firewalls route all transactions to the cryptographic gateway to be verified and authorized against ACLs before being processed by the server. Therefore, only the administrator can access ACLs; regular users cannot modify them.

5.1.5 Intrusion Detection.

An Intrusion Detection System (IDS) is defined as an application or process, which monitors an environment for the purpose of identifying activity that indicates system abuse, misuse, or malicious attack [28].

This section discusses popular approaches to IDSs and describes the role they may play in the security architecture used to protect HUMS.

Systems are vulnerable to inside and outside attackers. Inside attackers are users with access to the systems. Outside attackers are malicious attackers who try to get through the security mechanisms and gain access to information and resources. Attackers use several different tools and methods to break a system. A good description of different types of attacks can be found in reference 29. Several security mechanisms can be used to fortify the security of the HUMS, for example, firewalls and Virtual Private Network gateways. However, weaknesses and bugs in OSs and network protocols are continually exploited by system attackers. Thus, no system is unbreakable or 100 percent secure.

An IDS provides a second line of defense to aid the intrusion prevention techniques. If an attacker is successful in breaking into the system, the IDS can possibly detect the attack and signal an alarm. A good analogy would be home security systems. Home security systems signal an alarm when an intruder breaks in and acts as a second line of defense in addition to the preventive techniques (e.g., locks on doors and windows).

Because of their ability to detect attacks, IDSs are now an essential component of computer security. Some IDSs can detect an attack in real time and can actually take some action to stop a particular attack from inflicting further damage. Other IDSs can provide information about the attack that can be used to prevent those attacks from happening again. Advanced IDSs can detect never-before-seen attacks, but the more typical IDS detects attacks that are previously known [30].

The following sections discuss the most popular approaches used in the design of IDSs.

5.1.5.1 Signature-Based Approach.

The signature-based approach to intrusion detection involves looking at an invariant sequence of events that match a known type of attack. For example, a "Ping of Death" is a DoS attack. This attack involves sending an oversized ping packet that causes the target machine to reboot. A signature for this attack would have a rule: "A ping packet greater that 64 Kilobytes is an attack." Signatures can be created for the various types of attacks that are already known in a variety of ways (for example, hand translation of attack manifestations and automatic training and learning).

However, the signature-based approach has problems detecting truly novel attacks. In addition, false-alarm rates for these systems are very high because it is difficult to create signatures that successfully detect attacks while allowing normal traffic to flow through [31]. Finally, when new attacks are discovered, signatures must be created for the attacks and the system must be updated.

5.1.5.2 Anomaly-Based Approach.

Anomaly-based detectors have an advantage over signature-based detectors because they are able to detect novel attacks. This approach to intrusion detection involves creating statistical models for the behavior of the system. The IDS will send out an alarm whenever it observes an anomaly (that is, behavior that deviates from the normal behavior of the system).

The anomaly-based approach involves an initial training phase in which the system creates a model of the user or network. After this training phase, the IDS, upon observing behavior that deviates from the normal, generates an alarm.

Anomalous behavior does not always mean that an attack is underway, so anomaly detection systems should be carefully tuned to avoid high false-alarm rates. This requires that the activity of the system or user be stable over a period of time and not overlap with the activities of an intruder [32].

A user with very regular habits will be easy to model; therefore, the behavior of an attacker will likely deviate significantly from the usual behavior and an attack will be detected. However, a system administrator’s actions will vary considerably, and it will be more difficult to distinguish attacker from administrator. Also, a hacker may slightly deviate from what is perceived as normal behavior by the intrusion detection system for a long period of time, thus staying unnoticed. Another disadvantage is the large storage requirement for the statistical model that represents normal behavior.

5.1.5.3 Specification-Based Approach.

The specification-based approach to intrusion detection can detect new attacks that involve improper use of the system or application programs. This approach involves specifying normal behavior for the system and application programs. The audit records on a host are then examined to detect if there are any variations from normal behavior. Specification-based IDSs can provide very low false-alarm rates and detect a wide range of attacks, including many forms of malicious code such as Trojan horses and other viruses. These attacks exploit race conditions and take advantage of improper synchronization in distributed programs.

However, this approach is difficult to apply, because security specifications must be written for all programs to be monitored. Writing security specifications is difficult because system and application programs are constantly updated. The specification-based approach is best suited to certain critical programs (system or user programs) that are considered prime targets of the system.

5.1.5.4 Bottleneck Verification Approach.

The bottleneck verification approach to intrusion detection can be applied when only a few ways to transition from a certain set of states to another set of states are possible. The transition from user to superuser in a shell is one example. Whenever a transition from a user to superuser state happens, a bottleneck verification-based system will check whether or not the superuser command was indeed used to change from user state to superuser state. Thus, even if some novel method was used to carry out the transition, the attack will be detected. The bottleneck verification scheme therefore has the capability to detect new attacks.

5.1.5.5 Host-Based Approach.

The host-based approach to intrusion detection is deployed on a host. The OS on that host collects audit and log data, which is analyzed by the IDS to detect intrusions.

In cooperation with the host OS, the IDS can also monitor the interaction of various running applications to detect attacks.

The disadvantage of this approach is an attack that allows the attacker to gain control of the targeted machine may allow the attacker to disable the operations of the IDS. But the approach can help uncover attacks that do not create any externally observable behavior.

5.1.5.6 Network-Based Approach.

The network-based approach to intrusion detection looks at network traffic to detect attacks. The detection system in no way affects the performance of the protected hosts, and it allows simultaneous monitoring of a number of hosts. However, the approach suffers from performance problems with increasing network speeds [28].

5.1.6 Intrusion Detection Within a HUMS Context.

Of all the intrusion detection approaches mentioned above, there is no single best approach. For the HUMS, a combination of these approaches should be used and tailored to the needs of the system.

In selecting and deploying an IDS, certain selection criteria should be considered [33]. The following is a short summary of some selection criteria requirement categories:

- Detection. Detection requirements address various functional and security issues regarding detection of intrusions.

- Response. This set of requirements addresses various activities that are initiated when an intrusion is detected. The requirements address the degree to which the IDS can respond to intrusion related information.

- Deployment. Various practical issues, such as installation, platform support, and interoperability, are encompassed in this set of requirements.

Requirements in each category are associated with degrees of compliance. The criteria could be used as a basis for selecting an IDS that matches the security requirements of the HUMS system.

In addition to evaluating existing IDSs to determine which best fit the detection, response, and deployment criteria, an approach that incorporates anomaly detection and signature-based systems might best suit the security needs of the HUMS system. The detection system would be able to detect known and novel attacks. Host-based IDSs would be deployed at all the entities involved in the HUMS system, and network-based IDSs would be deployed at strategic network locations (i.e., gateway and access points). The host-based system would monitor audit logs generated by the OS to detect malicious activity by looking for the presence of known signatures and aberrations from normal profiles. The network-based IDS would look at network traffic for known and novel attacks.

Specification-based intrusion detection approaches should be employed for prime targets in the HUMS. Security specifications should be written and continuously updated to facilitate specification-based detection. Prime targets in HUMS include critical components of HUMS, which should be chosen based on hazard and vulnerability analysis.

5.1.7 Information and Data Protection.

This section describes information verification methods that can be used to ensure that the data has not been corrupted during storage. Data stored in the HUMS database may be modified either accidentally or by malicious attackers. Hash functions can verify correctness. Audit trails can trace violations of security policies back to individuals.

5.1.7.1 Hash Functions as Information Verification Methods.

Hash functions act on some input to produce new output. For example, in y = f (x), f () is a hash function that, on x as input, produces y as output. One-way hash functions are a special class of hash functions with the following properties [24]:

- Given x, it is easy to compute y.
- Given y, it is hard to compute x, that is, computing $x = f^{-1}(y)$ is difficult.
- Given x, it is hard to find another x′ such that f (x) = f (x′)

One-way hash functions are also called message digests, as they take the original message as input to produce output digest hash value. All modern hash algorithms produce hash values of 128 bits or higher. Hash algorithms also have the property that even a slight change in the input string causes the hash value to change drastically. This is called an avalanche effect. Thus, even if a single bit is flipped in the input string, at least half of the bits in the hash value will flip as a result.

Since it is computationally infeasible to produce a message that would hash to a given value or to find two messages that hash to the same value, the hash value of a message can serve as a cryptographic equivalent of the message. This makes a one-way hash function a central notion in public key cryptography, where it can be used to ensure data integrity.

Hash functions that do not use secret keys but are used for ensuring data integrity are called Modification Detection Codes (MDC). Another class of hash functions, which use a private key and provide both data integrity and origin authentication, are called MACs.

Examples of widely used MDC hash algorithms are Message Digest version 5 (MD5) and Secure Hash Algorithm (SHA). MD5 produces 128-bit hash value, while SHA can produce hash value of length 160 bits [24].

5.1.7.2 Audit Trails.

An audit trail is a series of records of events that take place in a system. The events may be related to the OS, application programs, or users of the system. Audit trails enable a user to identify and verify the activity of an information system.

Trusted systems must maintain audit trails of system activity to ensure that actions that violate the security policy of the system can be traced back to accountable individuals [34]. Even when a deliberate, malicious attack is not suspected, audit trails can be useful in restoring data integrity following unintentional mistakes or software failures.

A monitoring system generates audit trails. They have great importance in computer security for the following reasons:

- Individual Accountability. Audit trails can be used to detect insider attacks, which are malicious activities by authorized users of the system. They can be very difficult to detect because the attacker has a legitimate way to get into the system. Audit trails may deter inside attackers.

- Reconstructing Events. When a system is successfully attacked, it is a good idea to figure out exactly what happened. This includes determining the state the system was in before ascertaining how attackers compromised security measures. If it is not possible to determine who or what used the system, there is little chance of ascertaining what occurred to breach the security. Without audit trails, the exact same vulnerability would stay open, without any improvement in the system to prevent future attacks.

- Problem Monitoring. Audit trails may also be used to monitor the system to detect any abnormal activities. Disk failures and overuse of resources may be detected, for example.

- Intrusion Detection. IDSs identify attempts to penetrate a system and gain unauthorized access. Audit trails can help in detecting system misuse and can serve as a database for the anomaly detection engine of an IDS [34].

If an audit trail is to aid in tracing back an event after the fact, it must continuously record many system events at a fine level of detail. This results in a large volume of data, most of it useless in tracing attacks. However, data must be generated and stored in the hope that, if an abnormal event takes place, the few critical records needed to trace the event will be present.

A simple model of an auditing system is shown in figure 9 and consists of:

- Audit Data Collector. This is responsible for collecting the audit data.

- Audit Data Analyzer. This is responsible for analyzing the audit data. The format to be used to transfer data from the collector to the analyzer is an ongoing research effort in academia.

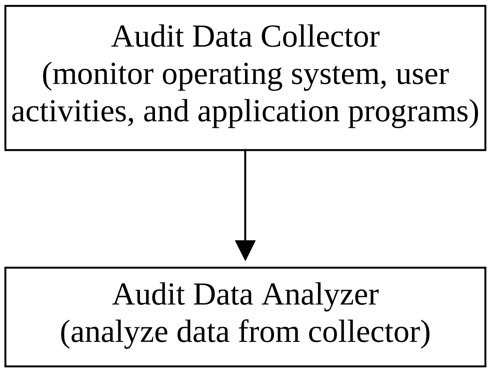

Audit Data Collector
(monitor operating system, user activities, and application programs)
Audit Data Analyzer
(analyze data from collector)

FIGURE 9. MODEL OF AN AUDITING SYSTEM

An audit trail should have a standard format, which helps overcome incompatibility and interoperability issues. Exchange of audit data from audit sources on different systems and collaborative analysis of data in a distributed environment is greatly facilitated by a standard format. Some of the proposed standards for the format of audit trails are Bishop's Standard Audit Trail Format, Normalized Audit Data Format, and Sun Solaris Common Audit Trail Interchange Format for UNIX®.

The content of the audit trails also needs to be standardized. Interoperability issues that arise from analyzing data from different sources, particularly in a distributed environment, can be solved by use of these standards. Some of the proposed standards are Trusted Computer System Evaluation Criteria and Security Criteria for Distributed Systems.

5.1.8 Data Integrity Assurance in HUMS Context.

The following considers data integrity of a HUMS.

In a HUMS, in which data is being transmitted over the ACARS network, all data to be transmitted is given as an input to the MDC hash algorithm. The message digest for that data is then calculated and encrypted along with the actual data. The whole encrypted message is sent over the network.

At the other end, a cryptographic gateway in a trusted domain decrypts the entire message to get the actual data and message digest. The gateway then computes the message digest on the actual data and verifies it with the message digest received over the network. The MDC hash algorithm property ensures that even if a single bit of the message is altered, the gateway will get a message digest different from the message digest of the original message.

Therefore, even if an intruder were able to alter the message, the gateway would detect it, and the message could be discarded. Hence, the gateway can alert the monitoring system and take appropriate action, like asking for retransmission of the message.

The same mechanism ensures data integrity for requests coming from users on an untrusted network and for responses to these requests by the HUMS server. A cryptographic gateway checks the message digest for requests from the users. Only if the message digest computed by the gateway matches the message digest in the message is the request considered further; otherwise, it is blocked. Also, when sending the response back to the user, the gateway calculates and appends the message digest to the response before encrypting it. The user can ensure that data integrity is maintained on his side by checking this message digest.

Audit trails should be used to log all activities, including source and number of messages being transmitted over the ACARS, number of retransmissions in case of failures, client requests, database read/write requests, and so on. This will aid in analyzing patterns of access to the HUMS data in case of a security breach and also during general maintenance activities.

5.2 SAFETY APPROACHES AND RISK MITIGATION.

In part, section 4.1.3 of DO-278 states: "Risk mitigation techniques may be used to reduce the CNS/ATM system's reliance on the COTS. The goal of these mitigation techniques is to accommodate the assigned failure condition classification by reducing the effect of anomalous behavior of COTS on the CNS/ATM system function. Risk mitigation techniques may be achieved through a combination of people, procedure, equipment, or architecture. For example, architectural means may involve partitioning, redundancy, safety monitoring, COTS safe subsets by the use of encapsulation or wrappers, and data integrity checking" [2].

Stroup, et al. [14] identified four activities associated with an effective risk mitigation strategy for incorporation into a system specification. These activities are hazard identification, assessing hazard effect and risk level, achieving requirements balance, and incorporating a balanced set of requirements into system specifications.

5.2.1 Hazard Analysis of the HUMS.

Figure 10 represents a model used to identify hazards from a safety perspective.

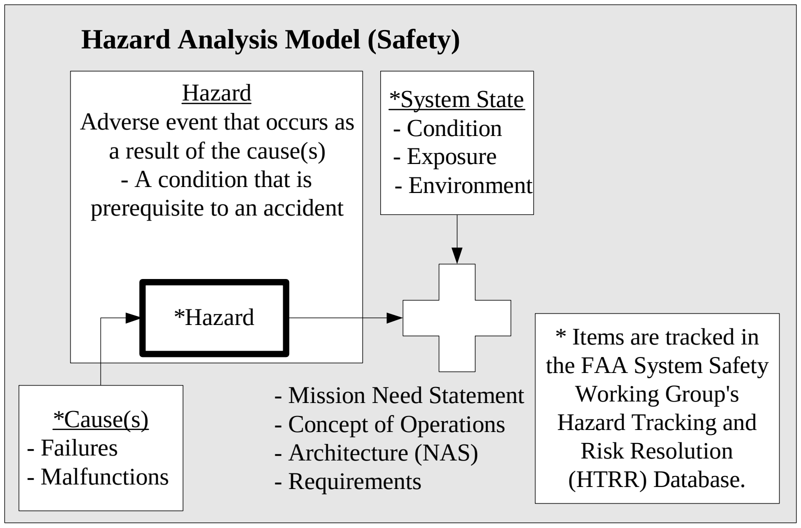

Hazard Analysis Model (Safety)
Hazard
Adverse event that occurs as a result of the cause(s)
- A condition that is prerequisite to an accident
*System State
- Condition
- Exposure
- Environment
*Hazard
*Cause(s)
- Failures
- Malfunctions
- Mission Need Statement
- Concept of Operations
- Architecture (NAS)
- Requirements
* Items are tracked in the FAA System Safety Working Group's Hazard Tracking and Risk Resolution (HTRR) Database.

FIGURE 10. HAZARD ANALYSIS MODE [13]

The mission need statement, concept of operations, requirements, and NAS architecture are considered while documenting a list of hazards. Also taken into account are the factors that cause the hazards and the different states in which that system is expected to perform, their environmental impact, and so on. Table 3 provides an example of a hazard analysis performed on a HUMS. Table 4 should be read as follows:

- Column 1 (Number): Numbers the hazard listed in the table.
- Column 2: Lists the particular subsystem being considered.
- Column 3 (Hazard): Describes the hazard under consideration.
- Column 4 (Cause Description): Provides a short description of the cause of the hazard.
- Column 5 (Effect Description): Provides a short description of the effect of the hazard.
- Column 6 (Remedial Technique): Provides a design technique that can detect and/or address the hazard under consideration.

TABLE 3. SAMPLE HAZARD ANALYSIS FOR A HUMS

No.	HUMS Subsystem Under Consideration	Hazard (Critical Event)	Causal Analysis		Remedial Technique
			Cause	Effect	
1			Noise on communica-tion channel.	Important data can be corrupted.	Message digests should be used to ensure data integrity.
2	ACARS	Data output from the ACARS network could be in an incorrect format.	Noise bursts during transmission could have corrupted the data.	Since the HUMS cannot understand the format, the packet will be rejected.	Format transformation mechanisms should be thoroughly tested. In case of dropped packets, retransmission should be requested.
3	ACARS	Data itself could be incorrect.	The onboard sensors may become faulty.	Erroneous data is transmitted.	Range checks or other forms of checks should be employed on all received data, e.g., a device that measures height or temperature cannot show a very steep change in a short interval of time.

TABLE 3. SAMPLE HAZARD ANALYSIS FOR A HUMS (Continued)

No.	HUMS Subsystem Under Consideration	Hazard (Critical Event)	Causal Analysis		Remedial Technique
			Cause	Effect	
4	ACARS	It can cease to function.	An onboard incident damages the device or the communication link.	No data transmitted until new/ backup unit takes its place.	Any lag/delay in messages received should be logged and flagged.
5	ACARS	Data sent to unintended recipient.	Recipient not authenticated.	Unauthorized access to important data.	Authentication and encryption technique should be used.
6	ACARS	Data transmission is interrupted.	Receiving substation may be out of service.	Data sent is not received by the receiving station.	Retransmission is tried once and then data is stored onboard.
7	ACARS	Either party claims not to be part of the communication.	A non-repudiation mechanism is not used.	Sender/ receiver may claim that they did not send/receive the data.	Digital signature is used for the communication.
8	ACARS	Receiving ground station goes down	The COTS software used in the ground systems crashes.	The receiving stations will not be able to receive messages, leading to lost messages.	The ground station software should be tested thoroughly. In case it is down, there should be a provision to store HUMS data on board.
9	HUMS Database	The database goes down.	Power failure, database fails during execution.	Data cannot be accessed.	A backup module/power supply should be present to take over in case of an emergency.

TABLE 3. SAMPLE HAZARD ANALYSIS FOR A HUMS (Continued)

No.	HUMS Subsystem Under Consideration	Hazard (Critical Event)	Causal Analysis		Remedial Technique
			Cause	Effect	
10	HUMS Database	The database goes down and when brought up again, it is in an inconsistent state.	Efficient check pointing mechanism not employed.	The data is in an inconsistent state.	Prior tested check pointing mechanism should be employed. Redundancy should be used for alternate forms of check pointing.
11	HUMS Database	Attempt can be made to make unauthorized access to the repository.	Malicious user trying to gain access.	Intruders can read/write data for which they have no authority.	Authorization through some form of encryption is required for authentication.
12	HUMS Database	Attempt can be made to access data with no permissions.	Malicious user trying to gain access.	Intruders can read/write data.	Check access rights for users before any read/write.
13	HUMS Database	HUMS incorrectly calculates the lifetime or maintenance decision related to one or more parts.	Human or system error	Incorrect data will be stored and/or transmitted.	Periodic manual checks should also be performed.
14	HUMS Database	Parts are incorrectly associated with the wrong vehicle.	Human or system error	Incorrect data and/or associations between data will be stored.	Periodic manual checks should also be performed.
15	HUMS Database	Data is corrupted during storage.	Noise bursts or power surges.	Incorrect data accessed by applications.	Validation mechanisms such as checksum should be present to ensure data integrity.

TABLE 3. SAMPLE HAZARD ANALYSIS FOR A HUMS (Continued)

No.	HUMS Subsystem Under Consideration	Hazard (Critical Event)	Causal Analysis		Remedial Technique
			Cause	Effect	
16	HUMS Database	The interface software to the HUMS and the repository software do not integrate very well.	The COTS software modules used do not integrate very well.	The repository cannot be accessed.	When choosing COTS software modules, integration issues between them must be considered.
17	HUMS Database	The repository is not designed to cope with the workload.	Too many requests generated.	The repository is not able to service all requests and may go down, or there may be a severe performance hit.	The approximate request load should be anticipated, and an appropriate repository must be selected.
18	HUMS Database	The primary and backup repositories are not synchronized properly.	Periodic synchroni-zation of repositories not enforced.	When the backup repository takes over, errors may arise.	The primary and backup repositories should be synchronized periodically.
19	HUMS Database	The data may be corrupted as it is being transferred from the repository to the requesting application.	System error	Erroneous data is transferred to the application that requested it.	The received data should be checked before use. If erroneous, it should be rejected and the request retransmitted.
20	HUMS Interface to Client Application	Client attempts to connect to the HUMS database but does not get a response.	The HUMS subsystem may be down.	Client will generate a request timeout error.	Appropriate messages should be returned so that the client knows that there is an error at the HUMS.
21	HUMS Interface to Client Application	Client software does not support SSL/Encryption.	Outdated software being used by client.	Client will not be able to connect to the HUMS.	Specifications for the client machine and software should be provided.

TABLE 3. SAMPLE HAZARD ANALYSIS FOR A HUMS (Continued)

No.	HUMS Subsystem Under Consideration	Hazard (Critical Event)	Causal Analysis		Remedial Technique
			Cause	Effect	
22	HUMS Interface to Client Application	Client session remains inactive for a long time without being terminated.	Incorrect logic being used for session timeout.	Long sessions pose a security risk, as they can be hacked into.	If no activity is registered for a period of time, the session should be terminated and have to be restarted.

*SSL = Security sockets layer

5.2.2 Vulnerability Analysis of the HUMS.

Figure 11 represents a model used to identify hazards from a security perspective.

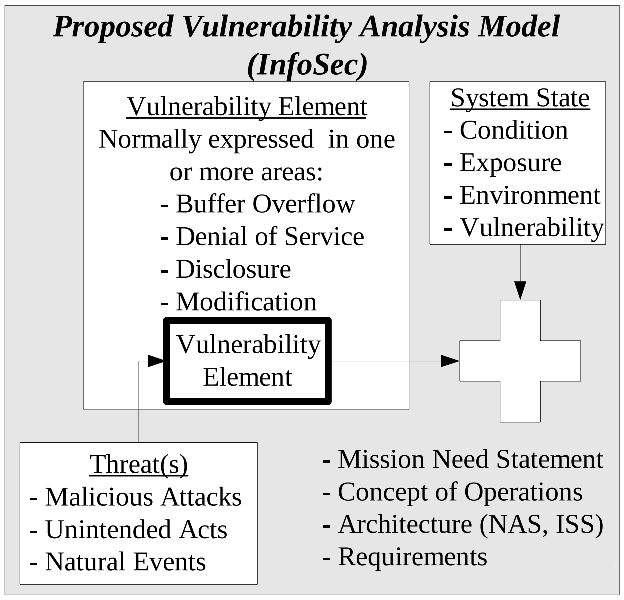


FIGURE 11. VULNERABILITY ANALYSIS MODEL [13]

The mission need statement, concept of operations, requirements, and NAS architecture are considered while documenting a list of hazards. Also taken into account are the factors that cause the vulnerabilities and the various states in which the system is expected to perform, their environmental impact and so on.

Table 4 provides an example vulnerability analysis performed on the various modules of the architecture proposed in figure 11 [13].

TABLE 4. SAMPLE VULNERABILITY ANALYSIS FOR A HUMS

No.	HUMS Subsystem Under Consideration	Threat	Effect	Remedial Technique
1	ACARS	Data can be intercepted during transmission by hackers.	Unauthorized access gained to important data.	Encryption should be used during transmission.
2	ACARS	Data can be corrupted during transmission by hackers.	Important data can be corrupted.	Message digests should be used to ensure data integrity.
3	ACARS	Encryption key is compromised	Unauthorized access to important data.	The session key is changed frequently.
4	ACARS	Denial of Service	Receiver is unable to handle incoming requests.	Intrusion detection mechanism is employed.
5	HUMS Gateway	Attacker launches a DoS attack on the HUMS gateway.	The HUMS database is incapable of handling data.	Host-based and network-based intrusion detection can be used to detect and respond to DoS attacks.
6	HUMS Database	Attacker compromises the secret key used for validating ACARS.	An attacker can now pretend to be the ACARS node that is sending data to the HUMS.	Efficient key storing techniques must be employed. Also, key lengths should be longer, and key setup techniques should be chosen to make the task of an attacker difficult.
7	HUMS Internal Network	An insider who has access to the private network can sniff the network or disrupt data transfer across it.	The confidentiality of transmissions between HUMS and other entities in the private network is compromised.	Network monitoring tools that can detect and thwart such attacks should be used.

TABLE 4. SAMPLE VULNERABILITY ANALYSIS FOR A HUMS (Continued)

No.	HUMS Subsystem Under Consideration	Threat	Effect	Remedial Technique
8	HUMS Gateway	The COTS software running on the HUMS gateway may be vulnerable to buffer overflow attacks.	An attacker can use a buffer overflow attack to run his code on the system, which will result in compromise of the gateway.	The software must be analyzed for vulnerability to buffer overflow attacks.
9	COTS Software	Attackers can exploit bugs and other vulnerabilities in software.	Trojans, backdoors, bugs, and other vulnerabilities in COTS software can result in compromise of sensitive information.	Bug reports and mailing lists of COTS components should be periodically checked to ensure that all patches are duly installed on the COTS software.
10	COTS Software	New updates/matches may cause a previous bug to resurface, resulting in a threat to security.	Previously corrected bugs continue to contribute to the vulnerability of the system.	Updates/patches should be installed carefully to avoid any undesirable side effects. Exhaustive testing should be done each time.
11	COTS Software	A user may install malicious software that may result in the transfer of data different from that requested.	Incorrect data will be supplied to requesting entities.	Inherent intrusion detection techniques employed in databases ensure that intrusions are detected and prevented.

5.3 CONSIDERATIONS FOR INTEGRATED SAFETY AND SECURITY APPROACHES.

Safety and security cannot be considered orthogonal to each other. For any effective risk mitigation strategy, both have to be taken into account. The advantages of this are twofold [14]:

- System deployment goals can be more effectively achieved when system safety and information security issues are resolved and risk mitigation requirements are implemented during the design phase.

- It is less expensive to accommodate risk mitigation strategies at the design phase, rather than during project completion.

Sections 5.2.1 and 5.2.2 provided example hazard and vulnerability analyses performed on the HUMS. The identified hazards and vulnerabilities are then viewed from the point of view of their hazard effect and likelihood and are assigned a risk level. (Refer to section 4.2 for details about hazard effect and likelihood.) A risk mitigation strategy is adopted for each hazard and vulnerability corresponding to its risk level. This strategy may in turn affect some of the original requirement definitions. The changes in requirements are then incorporated into the system specifications. This risk management strategy is illustrated in figure 12.

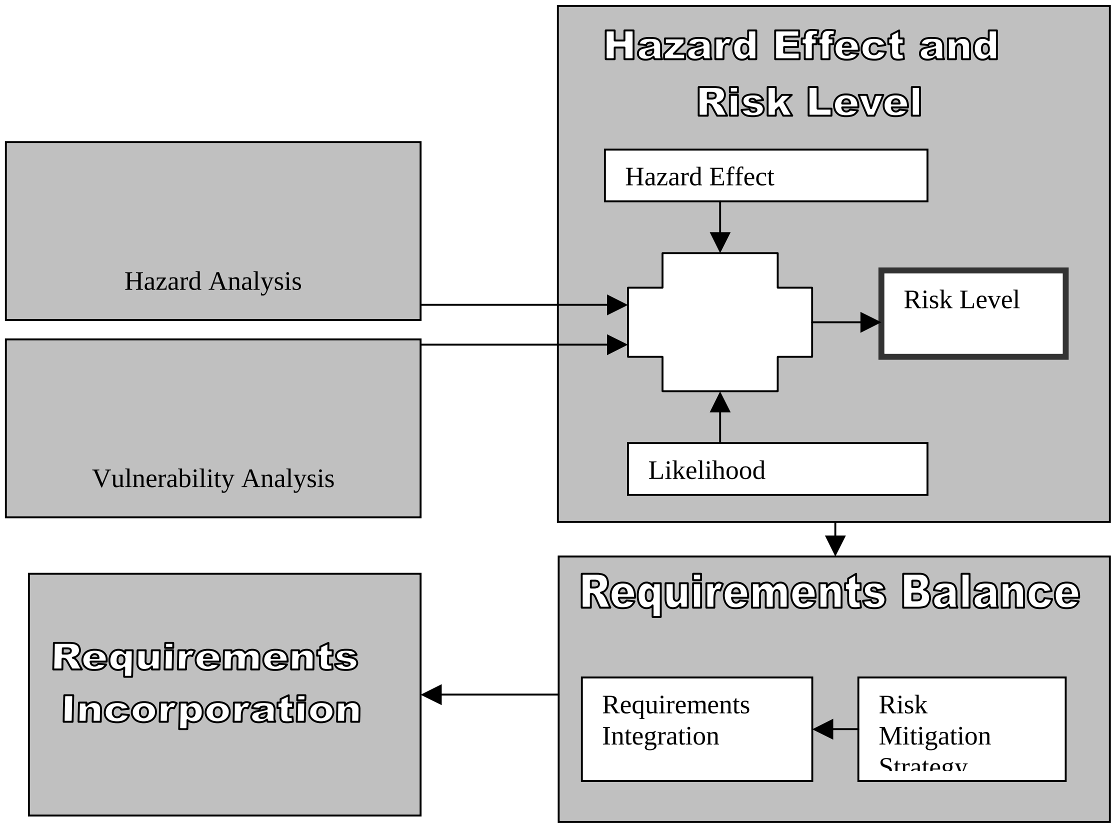


FIGURE 12. RISK MANAGEMENT STRATEGY [13]

6. STUDY OF DATA INTEGRITY IN GROUND-BASED SYSTEMS.

Two case studies were performed to analyze data integrity in ground-based systems. Each case study used a different approach. In case study 1, two COTS components were considered: one that did not participate in HUMS part-life determination (i.e., the Microsoft Windows OS) and one that did take part in HUMS part-life determination (i.e., Microsoft Access). Here, the term part-life determination refers to the information and maintenance policy used to determine the lifetime of parts. All objectives of DO-278 (Tables A-1 through A-10 and COTS Tables 4-1 through 4-3) were considered. In case study 2, a study was performed on a generic HUMS using the COTS-specific guidance of DO-278, specific to a COTS DBMS package.

6.1 CASE STUDY OF COTS HAVING DIFFERING HUMS PART ACTIONS (CASE STUDY 1).

DO-178B [1] provides guidance for software used in airborne systems, while DO-278 [2] provides guidance for ground-based systems. However, the HUMS AC only recognizes DO-178B, and not DO-278, as an acceptable standard for the HUMS airborne and ground-based system software. This report focuses on ground-based COTS systems, and therefore, uses DO-278 for software assurance guidance. In many areas of ground-based software operations, the accountability of DO-278 objectives should be made on a case-by-case basis. A preferred method of determining the applicability of DO-278 objectives would be to perform a detailed analysis of each system component for all the objectives of DO-278 guidance document and provide a rationale of compliance. A sample of the research investigation into this technique is presented below, specific for the HUMS system under study. Of key importance in this technique is the component's assigned assurance level from the system safety assessment.

Per the HUMS AC, the level of assurance for the HUMS application will be based on the results of the FHA. The FHA will identify what the effects will be on the aircraft if the HUMS application provides misleading information. The determined failure condition categories for the HUMS functions will depend on the functions provided by the HUMS. For example, a HUMS that is used to do rotor track and balance may be assessed to result in a major failure condition category should the HUMS fail to perform this function correctly. However, a HUMS that tracks effective usage hours for a critical structural component may result in either a catastrophic or a hazardous/severe-major failure condition category.

The method applied to both COTS case studies was targeted to DO-278's AL4, which is a new level that is not present in DO-178B. The primary reason to select AL4 was to exercise this new assurance level to analyze how applicable are its objectives given the information available for COTS components. Some assurance levels in DO-178B (airborne software guidance) map directly to DO-278 (ground-based software guidance). For instance, DO-178B Level C maps to AL3 and DO-178B level D maps to AL5. However, AL4 in DO-278 "was developed to account for certain CNS/ATM systems where AL3 was too stringent and AL5 was too lenient." From DO-178B, level C could cause or contribute to a major failure condition and level D could cause or contribute to a minor failure condition. Thus, AL4 is targeted somewhere between a major or minor failure condition. Only those DO-278 objectives for AL4 were considered in the research investigation below.

The two COTS products under study differ in that the Microsoft Windows OS does not make any determination of the part's suitability for remaining on-aircraft. The other COTS product, Microsoft Access, will make a part-life determination for the aircraft. Many of the objectives investigated for this assurance level resulted in identical activities for both COTS products needing to meet the objective. For instance, the planning objectives of DO-278 must be clearly met for both the Microsoft Windows OS and Microsoft Access. However, other objectives were not necessary for both COTS products. For example, the objective on software architecture is completely unimportant for the Microsoft Windows OS COTS product because it makes no part-life decisions. One the other hand, Microsoft Access does make those decisions and, therefore, an understanding of the software architecture is necessary. Given this disparity based on part-life, one could argue that the two components are essentially at two different assurance levels. However, with an integrated system component (such as an OS) there are operations that are performed at a variety of functional levels, which is problematic in the acceptance process.

The resultant objectives reported in table 5 from this research investigation are only those objectives that showed different activities of objective compliance between a non-part-life decision COTS (Microsoft Windows OS) and a part-life decision process (Microsoft Access).

The terms applicable, partial, and N/A used in table 5 are explained below.

- Applicable. The term applicable means that given the information available, the corresponding objective may be applied fully.

- Partial. The term partial means that given the information available, the corresponding objective may be applied only partially, and further methods of compliance would be needed to fully comply with the objective.

- N/A. The term N/A means that given the information available, the corresponding objective may not be applied.

The analysis in table 5 is only based on the demonstration HUMS for this study, and should not be used as a recommendation for similar systems.

TABLE 5. ANALYSIS OF DO-278, AL4, FOR TWO COTS COMPONENTS

DO-278 Annex A Reference	Objective	Paragraph Reference	Microsoft Windows OS (has no part-life effect)	Microsoft Access (affects part-life decisions)
2-3	Software architecture is developed. (A-2,2)	5.2.1a	Partial: Not applicable except any architectural issues arising during integration of the COTS OS into the HUMS system (e.g., a partitioned system or process management)	Applicable: for Microsoft Access setup and data structures.
2-8	Adaptation data and related processes are defined (when applicable). (A-2,4)	4.2	N/A: Adaptation data is HUMS specific information designed for future updates, such as part-life algorithm coefficients. In this case, the Microsoft Windows OS has no impact on this data.	Applicable

TABLE 5. ANALYSIS OF DO-278, AL4, FOR TWO COTS COMPONENTS (Continued)

DO-278 Annex A Reference	Objective	Paragraph Reference	Microsoft Windows OS (has no part-life effect)	Microsoft Access (affects part-life decisions)
3-7	Algorithms are accurate. (A-3,2)	6.3.1g	Partial: Algorithms used in the COTS components are proprietary. However, for certain special cases, this objective needs to be met. For example, error correction codes performed in OS drivers and similar related functions must meet this objective. Proof of accuracy needs to be provided by vendor, or applicant must verify the function correctness.	Applicable: Although there is no access to the actual algorithm used, it can be verified via functional correctness.
4-7	Algorithms are accurate. (A-4,1 and A-4,3)	6.3.2g	Partial: Algorithms used in the COTS components are proprietary. However, for certain special cases, this objective needs to be met. For example, error correction codes performed in OS drivers and similar related functions must meet this objective. Proof of accuracy needs to be provided by vendor, or applicant must verify the function correctness.	Applicable: Although typically there is no access to the actual algorithm, it can be verified via functional correctness.
4-8	Software architecture is compatible with high-level requirements.	6.3.3a	Partial: The need to accommodate this objective is very project specific. While the details of the exact software architecture used are proprietary, there may be alternate means to demonstrate that the architecture meets the high-level requirements. Some requirements may need an understanding of the OS architecture, e.g., task queuing, which must be demonstrated to comply with the high- level requirements. There are also examples of how the architecture of the OS is not needed, e.g., logging into the OS.	Applicable: Need verification via Microsoft Access API.
4-9	Software architecture is consistent. (A-4,4)	6.3.3b	Partial: same as 4-8	Applicable: need verification via Microsoft Access API.

TABLE 5. ANALYSIS OF DO-278, AL4, FOR TWO COTS COMPONENTS (Continued)

DO-278 Annex A Reference	Objective	Paragraph Reference	Microsoft Windows OS (has no part-life effect)	Microsoft Access (affects part-life decisions)
4-10	Software architecture is compatible with target computer. (A-4,2;A-4,5;A-4,6)	6.3.3c	Partial: same as 4-8.	Applicable: need verification via Microsoft Access API.
4-13	Software partitioning integrity is confirmed. (A-4,8)	6.3.3f	Applicable: If the OS supplies partitioning, then it should be confirmed. Other system components such as Microsoft Access would require confirmation that the OS supplies effective partitioning such that Microsoft Access does not interfere with other component operation.	N/A: However, one exception is that confirmation is needed to assure that Microsoft Access does not violate any partition mechanism, should such a system have partition mechanisms.
7-1	Test procedures are correct.	6.3.6b	Partial: Test procedure correctness is not needed for Microsoft Windows OS. However, procedures for testing of the integrated COTS component in the HUMS system would need to be verified as correct.	Applicable for HUMS specific requirements at this level 4, but as one goes higher in the assurance levels, a question arises about needing the COTS vendor test procedures and results.
7-2	Test results are correct and discrepancies explained.	6.3.6c	Partial: Typically test results are not available. However, procedures for testing of the integrated COTS component in the HUMS system would have to be verified as correct.	Applicable for HUMS specific requirements at this level 4, but as one goes higher in the assurance levels, a question arises about needing the COTS vendor test procedures and results.
8-6	Software life cycle environment control is established.	7.2.9	N/A: No control over the development process of the COTS product.	N/A for AL4, but as one goes higher in the AL it may be a factor.

TABLE 5. ANALYSIS OF DO-278, AL4, FOR TWO COTS COMPONENTS (Continued)

DO-278 Annex A Reference	Objective	Paragraph Reference	Microsoft Windows OS (has no part-life effect)	Microsoft Access (affects part-life decisions)
Adaptation Data				
4.2.1a	The data that may be adapted for a particular location is defined.	4.2.1.a	N/A for a Microsoft Windows OS, except for isolated cases such as an IP address.	Applicable
4.2.1b	The mechanisms for generating and modifying adaptation data are defined for each location.	4.2.1.b	N/A for a Microsoft Windows OS, except for isolated cases such as an IP address.	Applicable
4.2.1c	The objectives for verification, software quality assurance, software configuration management, and approval processes are satisfied for each location.	4.2.1.c	N/A for a Microsoft Windows OS, except for isolated cases such as an IP address.	Applicable

API = Application Programming Interface
IP = Internet Protocol
Note: DO-278 Annex A sections 4.2.1a, 4.2.1b, and 4.2.1c are related to adaptation data.

6.2 CASE STUDY OF A WIDELY USED COTS DBMS WITHIN A HUMS CONTEXT (CASE STUDY 2).

Case study 2 was also performed on an in-service, generic HUMS using the COTS-specific guidance of DO-278. A generic HUMS was selected for the case study because it is a ground-based system and uses COTS components, specifically, a database management component.

The case study in this section makes no judgment about the applicability of the objective to this particular COTS component. This study, rather, assumes each objective must be met and discusses the issues that arise when considering the objectives for such a DBMS component.

Although the FAA HUMS ACs' guidance does not call out DO-278, it is used for this case study because it provides relevant information for ground-based systems using COTS components.

The lessons learned from this case study might also apply to other ground-based systems, such as CNS/ATM systems.

This section describes the case study, using a particular DBMS as a representative COTS product and applying the COTS-specific objectives and activities described in Section 4 of DO-278.

Figure 13 illustrates the generic HUMS. The COTS components, which are used in the construction of this HUMS, are:

- ACARS
- COTS installation software
- COTS Common Object Request Broker Architecture (CORBA)—compliant communications software
- COTS Relational DBMS

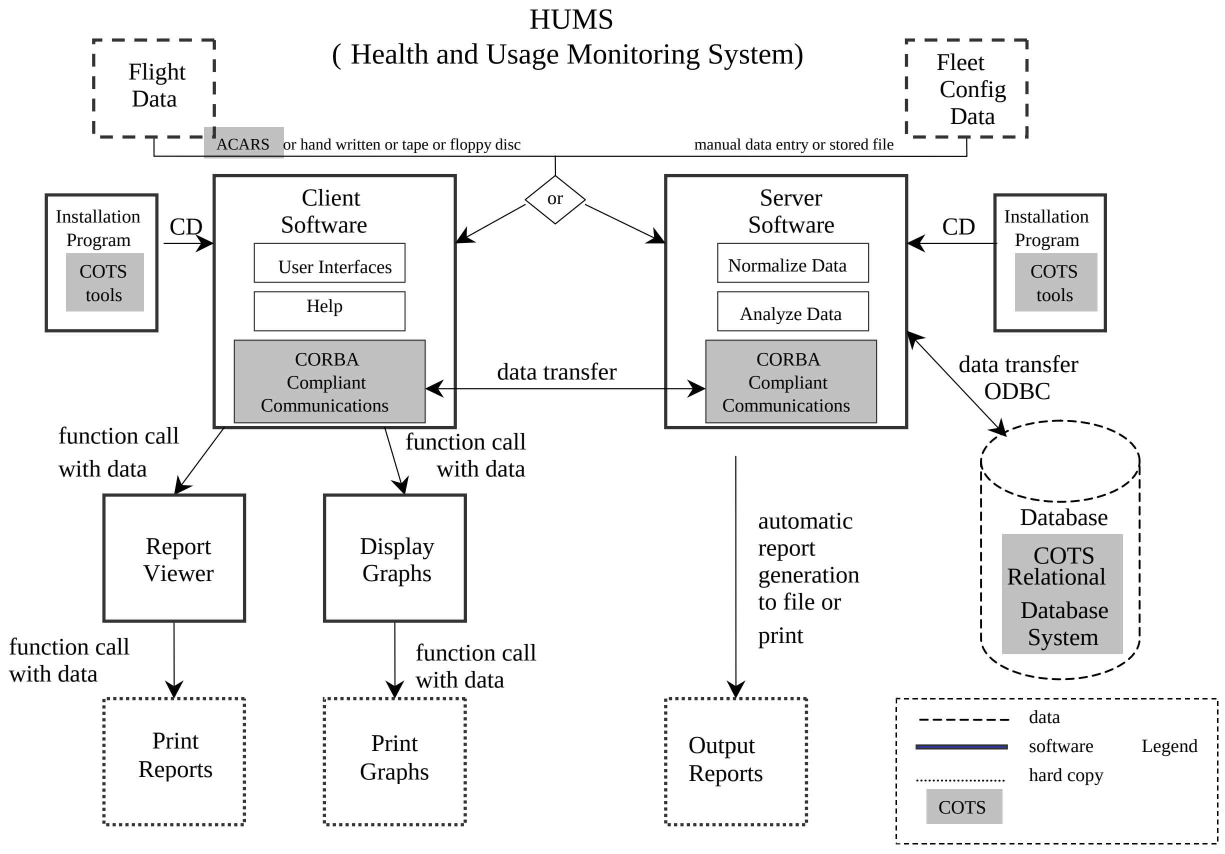


FIGURE 13. REPRESENTATIVE GROUND-BASED SYSTEM

Database management systems are used in a wide variety of applications. A very commonly used, extensive DBMS is selected for this case study. The particular DBMS is a representative

COTS product because of its complexity, which provides a sample to exercise DO-278, and because of its widespread use, adds to the applicability and value to this study.

Like many vendors of sophisticated software, the vendor of the example COTS database maintains an extensive web site that contains a great deal of information about their products. The COTS database vendor also encourages communication within their user community. Table 6 summarizes how the information available regarding the COTS database might be used to exercise DO-278 coverage of COTS-related objectives. Table 6 describes the DO-278 objective and associated lessons learned. For a clearer understanding of the lessons learned, refer to DO-278 activities supporting the objective.

TABLE 6. DO-278 COTS-RELATED OBJECTIVES—CASE STUDY 2

DO-278 Objective Reference	DO-278 Objective Description	Lessons Learned
4.1.4.1.a	Activities for acquisition and integral processes, including additional considerations, integration, and maintenance, are defined.	• The database life cycle data has limited availability, and the life cycle description is not apparent. • The integration of this highly commercialized database was not difficult; however, a COTS product more tightly coupled with the ground-based system may increase integration-specific activities. • Historical information was difficult to obtain. User discussion forums may offer some insight, but this may provide little justification toward product acceptance. • Guidance in DO-278 with respect to COTS vendor qualifications was difficult to interpret for the activity associated with this objective, as the level of qualification requested is not quantitatively understood. This area needs improvement in DO-278. • In general, it would not be suggested that the developers of the ground-based system attempt to modify the COTS database beyond the normal customizing capabilities of the COTS product. In particular, if modifications are necessary, vendor support might not be available and associated verification would be difficult. • In the COTS acquisition planning process, consideration should be given to acquiring access to discussion and other related forums to assist in determining the maintenance and health of particular versions.

TABLE 6. DO-278 COTS-RELATED OBJECTIVES—CASE STUDY 2 (Continued)

DO-278 Objective Reference	DO-278 Objective Description	Lessons Learned
4.1.4.1.a (Continued)		• Summary—There may not presently be any COTS products that were developed to explicitly support DO-278 compliance with respect to applications that incorporate the COTS. As such, the customer would be required to analyze and discuss with the vendor its ability to meet these objectives.
4.1.4.1.b	Transition criteria for these processes and transition criteria with respect to CNS/ATM software life cycle processes are defined.	• No COTS integral process was found in DO-278, yet it asks for the integral process to be defined. The assumption made was that the term integral process is used from DO-178B and represents the planning, verification, quality assurance, and configuration management activities. However, DO-278 should include a reference to the DO-178B integral process description for first time users. • COTS vendor transition criteria were not available from this database developer.
4.1.4.1.c	Plans for COTS processes, including COTS transition criteria, are consistent with the supplier's software plans.	• Plans were not available from the COTS vendor. Plan consistency per DO-278 was not possible for this particular COTS product. • In reviewing the transition criteria guidance in DO-278 and DO-178B as applied to COTS products, it was difficult to assess the real guidance being offered. This portion of DO-278 was found to be weak; it needed better definition for proper understanding of the objective's true intent.
4.1.5.1.a	The degree to which of the CNS/ATM software requirements are satisfied by the COTS capabilities is determined.	• The supplier's target system offered two different database packages. The flexibility of having two different database packages to choose from may require two separate requirement traceability matrices. • Technical consulting from the COTS vendor may be one mechanism for assisting the effort to have the COTS database software meet DO-278 requirements.

TABLE 6. DO-278 COTS-RELATED OBJECTIVES—CASE STUDY 2 (Continued)

DO-278 Objective Reference	DO-278 Objective Description	Lessons Learned
4.1.5.1.b	The adequacy of life cycle data available for assurance purposes is determined.	• The life cycle data for this database was not readily available, and no evaluation as to the adequacy of the COTS life cycle data has been made. • This highly commercialized product leaned toward lack of support for older versions. If the COTS components are upgraded to be current in the ground-based application, then the vendors who provide them will more likely support the COTS components.
4.1.5.1.c(1)	The derived requirements are identified. Derived requirements consist of (1) Requirements imposed on the CNS/ATM system due to the usage of COTS and (2) Requirements to prevent the unneeded capabilities of the COTS from adversely affecting the CNS/ATM system.	• The subject database included undesirable capabilities such as automatic database calculation update. Automatic functions such as this may require additional requirements by the application in both development and installation.
4.1.5.1.d	The compatibility of COTS with target hardware and other ground-based software is assured.	• The COTS database would be compatible with the target environment. However, the vendor's information does not, and cannot reasonably, cover every possible configuration of the target environment. Careful functional testing by the supplier is required.

TABLE 6. DO-278 COTS-RELATED OBJECTIVES—CASE STUDY 2 (Continued)

DO-278 Objective Reference	DO-278 Objective Description	Lessons Learned
4.1.6.1	Section 4.1.6.1 of DO-278 states "There are no additional verification objectives imposed upon the CNS/ATM system [HUMS being developed] because of the use of COTS." However, it is still necessary to meet the verification objectives of the 10 tables in section 3 of DO-278. (Note: Brackets added for clarity.)	• High-level requirements coverage is possible; low-level requirements are problematic. Other areas of difficulty included requirement traceability and conformity to standards. • Supplemental software (glue code, etc.) is mainly developed by the applicant/supplier and as such can fully follow DO-278 in both development and verification. However, if the COTS vendor supplies this code, then alternate means may be required to comply with this objective. • Because the application under study here is of a very low assurance level (perhaps AL5 or AL6), few of these objectives are required to be met. It is clear, however, that arguing compliance to AL1 or AL2 would be very difficult with a COTS component unless the supplier had access to details and data regarding the development of the COTS component. One would surmise that the COTS component would need to have been developed under DO-178B or DO-278 guidance to accomplish these assurance level goals.
4.1.7.1.a	The COTS specific configuration and data items (for example, software, documentation, adaptation data) are uniquely identified in the CNS/ATM software configuration management system.	• The vendor's COTS software identification was apparent, but the developer of the ground-based system is responsible for coordinating the precise hardware and OS configurations to be used with the COTS database configuration. Anecdotal data also suggest that some COTS vendors may not have proper control of their delivered product; subsequent deliveries may not have been based on previous deliveries.

TABLE 6. DO-278 COTS-RELATED OBJECTIVES—CASE STUDY 2 (Continued)

DO-278 Objective Reference	DO-278 Objective Description	Lessons Learned
4.1.7.1.c	The CNS/ATM change control process ensures that the incorporation of COTS releases is controlled.	• The COTS database vendor provides information regarding updating releases and versions of the database; however, the information is typically relevant to updating from the last release or version of the database to the current one. Updating from releases or versions in the more distant past could present a problem from the perspective of obtaining adequate information. However, the developer of the ground-based system should test the ground-based system with the upgraded COTS database to confirm the expected impact or lack of impact the upgrade will cause.

6.3 THE COTS INSTALLATION SOFTWARE.

Additionally, COTS installation software must be considered. Although not explicitly mentioned, objectives for correct installation of software target computer systems are lacking in DO-278. This would include verification that the installation package software for the ground-based system has correctly installed the system using all selected installation target environments. A COTS installation software package used on the subject system was studied to determine the information that would be available to help developers of the ground-based system verify that the installations of the ground-based systems would be correct. Information regarding internal configuration control, testing, or source code for the COTS installation software does not appear to be readily available.

7. SUMMARY AND RECOMMENDATIONS.

7.1 SUMMARY.

Ground-based processing systems are likely to use COTS components for maintaining flight-critical data. Hence, the ground-based COTS processing systems must be trustworthy and secure to maintain the integrity of the data. This report presents an investigation of the issues that arise from the use of COTS components in safety-critical systems and the approaches required to ensure flight-critical data integrity. The research was focused on information and data protection, access security, and the processes and objectives for ensuring data integrity.

The existing guidance governing the use of COTS components for safety-critical applications and the objectives of current guidance from the point of view of applicability and shortcomings were investigated. The use of hazard analysis and vulnerability analysis as a means for developing an effective risk mitigation strategy was also investigated. Individual COTS components such as the Microsoft Windows operating system and the Microsoft® Access®

DBMS were analyzed from a safety and security perspective. An investigation of the relevance of the rotorcraft AC 27-1B Change 1 and AC 29-2C Change 1 to a COTS components scenario was presented. To address security and vulnerability concerns, several technologies were investigated, such as encryption/decryption, authentication, access control, and intrusion detection, particularly in relation to application within a HUMS context.

Two case studies involving COTS products were done to determine if the products can be certified by following existing certification guidance, such as DO-178B or DO-278. This report presented a possible process using COTS components that can readily be analyzed. In addition, feasibility of the presented process is demonstrated by the implementation of a proof-of-concept (POC) and scaled-down version of the HUMS system. The architecture of this demonstration project shows that various security mechanisms can be incorporated in the design of this scaled-down HUMS system. Finally, a case study of COTS used in a different domain was completed to determine if lessons could be learned in the design process.

7.2 RESULTS.

The main result of the investigation was the analysis of the existing certification guidance for COTS component-based ground processing systems, including DO-178B, DO-278, and the HUMS AC, from the point of view of applicability and shortcomings. It was also demonstrated how hazard analysis and vulnerability analysis can be used as a means for developing an effective risk mitigation strategy. Using a commercial database in a COTS-based flight-information system as a study case, the investigation led to several useful lessons, for instance, the life cycle data for the example database was not readily available, no evaluation as to the adequacy of the COTS life cycle data has been made, and the support for older versions was not available once the COTS components were upgraded.

7.3 RECOMMENDATIONS.

A future research task may be to investigate the issue of independence and redundancy of software systems. Multiple-version redundancy has been offered in the HUMS AC 27-1B Change 1 and AC 29-2C Change 1 as a means for independent verification of ground-based COTS hardware and software equipment. For example, if two software systems can justify that they do not have any common failures, it can be claimed that each may be used for an independent verification of the other. Yet, improperly controlled or incorrect data submitted to redundant processing by a second dissimilar PC with different COTS components compared to the primary processor may in all probability still yield incorrect predictions.

Another immediate recommended extension of this study is to investigate the effectiveness of various alternate approaches as mitigating actions in ground-based COTS systems. Due to the nature of COTS components, it is not always possible to meet the required objectives. In case these objectives are not met by the COTS components using a normal system design process, mitigating actions can be considered to increase the reliability, trustworthiness, and failsafe properties of the COTS systems. It is recommended that these mitigating actions need to be taken at all three logical stages of flight data (data communication, processing, and storage) and should be based on a hazard assessment. In addition, it is recommended to analyze the effectiveness of mitigating actions based on application characteristics, such as non-time-critical and possible human intervention, and to determine the viability of making these mitigating actions a part of DO-178B or DO-278 guidance.

Although not explicitly mentioned, objectives for correct installation of software target computer systems are lacking in DO-278. It is recommended that COTS installation software must also be considered as an objective. This would include verification that the installation package software for the ground-based system has correctly installed the system using all selected installation target environments.

It is recommended that cryptography should be used in HUMS to make the ACARS signals secure with techniques such as secret key cryptography and public key cryptography, since anyone having a very high frequency or a high-frequency communication device that is within range of the transmissions can intercept these ACARS signals. In addition, IDSs should be used in HUMS to enable detection and subsequent prevention of intruders into the network.

8. REFERENCES.

1. RTCA DO-178B, prepared by RTCA SC-167/EUROCAE WG-12, "Software Considerations in Airborne Systems and Equipment Certification," Washington, DC, December 1, 1992.

2. RTCA DO-278, prepared by SC-190, "Guidelines for Communication, Navigation, Surveillance and Air Traffic Management (CNS/ATM) System Software Integrity Assurance," March 5, 2002.

3. Maurizio Morisio, "Commercial-Off-The-Shelf (COTS): A Survey. A DACS State-of-the-Art Report," Contract Number SP0700-98-D-4000, Data and Analysis Center for Software, December 2000.

4. CAST paper, "Use of a Level D Commercial Off-the-Shelf Operating System in Systems With Other Software of Levels C and/or D," CAST-14, http://www.faa.gov/aircraft/air_cert/ design_approvals/air_software/CAST/cast_papers/.

5. James E. Land, "HUMS – The Benefits – Past, Present and Future," *IEEE Proceedings of the Aerospace Conference*, 2001, Vol. 6, CA, USA, 2001.

6. Advisory Circular 27-1B, Change 1, "Certification of Normal Category Rotorcraft," April 25, 2003; Miscellaneous Guidance (MG) 15, February 12, 2003, Chapter 3, "Airworthiness Standards Normal Category Rotorcraft;" AC 27 MG 15, "Airworthiness Approval of Rotorcraft Health Usage Monitoring System (HUMS)," February 12, 2003.

7. Advisory Circular 29-2C, Change 1, "Certification of Transport Category Rotorcraft," April 25, 2003; AC 27-1B, Change 3, Chapter 3, "Airworthiness Standards Transport Category Rotorcraft;" AC 29 MG 15, "Airworthiness Approval of Rotorcraft Health Usage Monitoring System (HUMS)," February 12, 2003.

8. Richard Sewersky, "System Design Assessment for a Helicopter Structural Use Monitor," MS Thesis, MIT, June 1999.

9. Aloke Roy, "Secure Aircraft Communications Addressing and Reporting System (ACARS)," *20th Digital Avionics Systems Conference (DASC)*, Vol. 2, pp. 124-134, Honeywell International, Inc., Columbia, Maryland, October 2001.

10. Curtis Risley, James McMath, and Captain Brian Payne, "Experimental Encryption of Aircraft Communications Addressing and Reporting System (ACARS) Aeronautical Operational Control (AOC) Messages," *20th Digital Avionics Systems Conference (DASC)*, Vol. 2, pp. 320-327, October 2001.

11. SANS Institute Resources, "NSA Glossary of Terms Used in Security and Intrusion Detection," http://www.sans.org/resources/glossary.php

12. The FAA System Safety Handbook, http://www.faa.gov/library/manuals/aviation/risk_management/ss-handbook/

13. National Airspace Modernization, "System Safety Management Program," FAA Acquisition Management System, January 2004.

14. Ronald Stroup, Warren Naylor, and Michael LeBeau, "The Leveraging of Safety and Information Security Engineering Principles: Establishing an Effective Level of Integrated Risk Management," *FAA National Software Conference*, May 2002.

15. Code of Federal Regulations, Title14 - Aeronautics and Space, National Archives and Records Administration, http://www.access.gpo.gov/nara/cfr/cfr-table-search.html

16. Kelly Hayhurst, "The Guidance and Control Software Project: A Software Engineering Case Study," Assessment Technology Branch, NASA Langley Research Center, http://shemesh.larc.nasa.gov/people/kjh/short-GCS-talk/

17. Ron Stroup, "Implementation of RTCA DO-278/ED-109," *FAA National Software Conference*, May 2002.

18. "Integrated Mechanical Diagnostics (IMD) Health & Usage Monitoring System (HUMS)," DSMC Lecture, June 24, 1999.

19. Richard A. Sewersky, "System Design Assessment for a Helicopter Structural Usage Monitor," Paper for a Masters of Science in Engineering and Management at MIT, June 1999.

20. Uma D. Ferrell and Thomas K. Ferrell, "Software Service History Handbook," FAA report DOT/FAA/AR-01/116, January 2002.

21. CAST-13, "Automatic Code Generation Tools Development Assurance," June 2002, http://www.faa.gov/aircraft/air_cert/design_approvals/air_software/cast/cast_papers/

22. Nancy G. Leveson, *Safeware, System Safety and Computers*, Addison-Wesley Publishing Company, Chapter 12, pp. 436, September 1995.

23. "Modular Certification," NASA Report December 2002, NASA/CR – 2002-212130, SRI International, Menlo Park, CA.

24. Bruce Schneier, *Applied Cryptography*, Second Edition, John Wiley and Sons, Inc., 1996.

25. E. Bertino and E. Ferrari, "Data Security," *Proceedings of the Twenty-Second Annual International Computer Software and Applications Conference (COMPSAC)*, pp. 228-237, 1998.

26. P.C. Clark, M.C. Meissner, and K.O. Vance, "Secure Compartmented Data Access Over an Un-Trusted Network Using a COTS-Based Architecture," *16th Annual Conference Computer Security Applications (ACSAC)*, 2000.

27. C.L. Schuba and E.H. Spafford, "A Reference Model for Firewall Technology," *Proceedings of the 13th Annual Computer Security Applications Conference*, 1997.

28. B. Mukherjee, L.T. Heberlein, and K.N. Levitt, "Network Intrusion Detection," *IEEE Network*, Vol. 8, Issue 3, pp. 26-41, May-June 1994.

29. R.P. Lippmann, D.J. Fried, I. Graf, J.W. Haines, K.R. Kendall, D. McClung, D. Weber, S.E. Webster, D. Wyschogrod, R.K. Cunningham, and M.A. Zissman, "Evaluating Intrusion Detection Systems: The 1998 DARPA Off-Line Intrusion Detection Evaluation," *DARPA Information Survivability Conference and Exposition (DISCEX)*, Vol. 2, pp. 12-26, 2000.

30. Stefan Axelsson, "Research in Intrusion Detection Systems: A Survey," Department of Computer Science and Engineering, Chalmers Institute of Technology, Goteborg, Sweden, TR:98-17, December 15, 1998, Revised August 19, 1999.

31. M. Bishop, S. Cheung, et al., "The Threat From the Net," *IEEE Spectrum*, 1997.

32. J. McHugh, A. Christie, and J. Allen, "Defending Yourself: The Role of Intrusion Detection Systems," *IEEE Software*, Vol. 17, Issue 5, pp. 42-51, September-October 2000.

33. E. Amoroso and R. Kwapniewski, "A Selection Criteria for Intrusion Detection Systems," *Proceedings of the 14th Annual Computer Security Applications Conference*, pp. 280-288, 1998.

34. B. Schneier and J. Kelsey, "Secure Audit Logs to Support Computer Forensics," *ACM Transactions on Information and System Security*, Vol. 2, No. 2, pp. 159-176, 1999.

9. GLOSSARY.

Anomaly Detection Mode—A model in which intrusions are detected by looking for activity that is different from the user's or system's normal behavior.

Assets—Information or resources to be protected by the countermeasures of a target of evaluation (TOE).

Assurance—Grounds for confidence that an entity meets its security objectives.

Attack—An attempt to bypass security controls on a computer. The attack may alter, release, or deny data. Whether an attack will succeed depends on the vulnerability of the computer system and the effectiveness of existing countermeasures.

Audit—The independent examination of records and activities to ensure compliance with established controls, policy, and operational procedures, and to recommend any indicated changes in controls, policy, or procedures.

Audit Trail—A chronological record of the system resource used in computer security systems. This includes user login, file access, various other activities, and whether any actual or attempted security violations occurred, legitimate and unauthorized.

Buffer Overflow—This happens when more data is put into a buffer or holding area than the buffer can handle. It results from a mismatch in processing rates between the producing and consuming processes. This can result in a system crash or the creation of a back door leading to system access.

Class—A grouping of families that share a common focus.

Component—The smallest selectable set of elements that may be included in a protection profile (PP), a security target (ST), or a package.

Compromise—An intrusion into a computer system where unauthorized disclosure, modification, or destruction of sensitive information may have occurred.

Computer Abuse—The willful or negligent unauthorized activity that affects the availability, confidentiality, or integrity of computer resources. Computer abuse includes fraud, embezzlement, theft, malicious damage, unauthorized use, denial-of-service (DoS), and misappropriation.

Computer Security—Technological and managerial procedures applied to computer systems to ensure the availability, integrity, and confidentiality of managed information.

COTS Installation Software—The software used to manually or automatically install some other COTS software, e.g., InstallShield®, WinZip™, etc.

Countermeasure—Action, device, procedure, technique, or other measure that reduces the vulnerability of an automated information system. Countermeasures aimed at specific threats and vulnerabilities involve more sophisticated techniques, as well as activities traditionally perceived as security.

Cryptography—The science concerning the principles, means, and methods for rendering plaintext unintelligible and for converting encrypted messages into intelligible form.

Denial-of-Service—Action(s) that prevent any part of an automated information system (AIS) from functioning in accordance with its intended purpose.

Dependency—A relationship between requirements where the requirement that is depended upon must normally be satisfied for the other requirements to be able to meet their objectives.

Element —An indivisible security requirement.

Evaluation—Assessment of a PP, ST, or TOE against defined criteria.

Evaluation Assurance Level—A package consisting of assurance components from section 3 of DO-278 that represents a point on the common criteria (CC) predefined assurance scale.

Evaluation Authority—A body that implements the CC for a specific community by means of an evaluation scheme and thereby sets the standards and monitors the quality of evaluations conducted by bodies within that community.

Family—A group of components that share security objectives but may differ in emphasis or rigor.

Firewall—A system or combination of systems that enforces a boundary between two or more networks. It limits access between networks in accordance with local security policy. The typical firewall is an inexpensive microbased UNIX® box, clean of critical data, that includes many modems and public network ports, but just one carefully watched connection back to the rest of the cluster.

Hacker—A person who enjoys exploring the details of computers and how to stretch their capabilities, or a malicious or inquisitive meddler who tries to discover information by poking around.

Hacking—An unauthorized use, or attempt to circumvent or bypass the security mechanisms, of an information system or network.

Hazard—A state or set of conditions of a system (or an object) that, together with other conditions in the environment of the system (or object), will lead inevitably to an accident (loss event).

Health—A measure of the overall flightworthiness of the aircraft. Health is assessed by examining the instantaneous indicators of the well being of vital components on the aircraft as well as trend analysis of these indicators.

Information Security—The result of any system of policies or procedures for identifying, controlling, and protecting from unauthorized disclosure, information whose protection is authorized by executive order or statute.

Integrity—Assuring information will not be accidentally or maliciously altered or destroyed.

Intrusion Detection—Pertaining to techniques that attempt to detect intrusion into a computer or network by observation of actions, security logs, or audit data. Detection of break-ins or attempts, either manually or via software expert systems, that operate on logs or other information available on the network.

Misuse Detection Model—The system detects intrusions by looking for activity that corresponds to known intrusion techniques or system vulnerabilities. Also known as Rules Based Detection.

Monitoring—The means by which information can be gathered on the aircraft's vital systems.

Network Based—Network traffic data along with audit data from the hosts used to detect intrusions.

Network Security—Protection of networks and their services from unauthorized modification, destruction, or disclosure, and provision of assurance that the network performs its critical functions correctly, without harmful side effects. Network security includes providing for data integrity.

Package—A reusable set of either functional or assurance components (e.g., an evaluation of assurance level) combined together to satisfy a set of identified security objectives.

Product—A package of information technology (IT) software, firmware or hardware that provides functionality designed for use or incorporation within a multiplicity of systems.

Protection Profile—An implementation-independent set of security requirements for a category of TOEs that meet specific consumer needs.

Risk—The hazard level combined with (1) the likelihood of the hazard leading to an accident (sometimes called danger) or compromise and (2) hazard exposure or duration (sometimes called latency).

Role—A predefined set of rules establishing the allowed interactions between a user and the TOE.

Safety—Freedom from unintentional accidents or losses.

Security—A condition that results from the establishment and maintenance of protective measures that ensure a state of inviolability from hostile acts or influences.

Security Architecture—A detailed description of all aspects of the system that relate to security, along with a set of principles to guide the design. Security architecture describes how the system is put together to satisfy the security requirements.

Security Objective—A statement of intent to counter identified threats and/or satisfy identified organization security policies and assumptions.

Security Target—A set of security requirements and specifications to be used as the basis for evaluation of an identified TOE.

System—A specific IT installation with a particular purpose and operational environment.

Target of Evaluation—An IT product or system and its associated administrator and user guidance documentation that is the subject of an evaluation.

Use—A measure of how the life of components is being expended on the life-limited parts of the aircraft as well as determining the time to overhaul of major components.

User—Any entity (human user or external IT entity) outside the TOE that interacts with the TOE.

Vulnerability—Hardware, firmware, or software flow that leaves an AIS open for potential exploitation. A weakness in automated system security procedures, administrative controls, physical layout, internal controls, etc., that could be exploited by a threat to gain unauthorized access to information or disrupt critical processing.

APPENDIX A—A PROOF-OF-CONCEPT DEMONSTRATION ARCHITECTURE

This section outlines the architecture of the demonstration project. The purpose of this appendix is to demonstrate the feasibility of the process described in section 6 of this report. This is carried out by implementing a proof-of-concept (POC) and demonstrating a scaled-down version of the system, as described in section 6. The requirements of this POC are

- The system needs to be distributed to correctly reflect the real-life scenario of the Aircraft Communication Addressing and Reporting Systems (ACARS), the health and usage monitoring system (HUMS) engine, and client applications that run in physically different locations.
- Follow an architecture similar (but scaled down) to the one proposed.
- Demonstrate the use of technologies such as encryption, access control, and replication required for HUMS.
- Simulate the different problems that could occur when using commercial off-the-shelf (COTS) components and a public network.
- Have a graphical user interface (GUI) display to depict the various components such as the ACARS simulator, client, monitoring module, and so on.

A.1 PROOF-OF-CONCEPT ARCHITECTURE.

The HUMS demonstration project consists of three applications using client server and sockets technology for communication. The three applications are the ACARS application, the HUMS application, and the client application.

Figure A-1 shows a block diagram of the architecture to be used for the POC. The HUMS and the data repository are assumed to be within a private network. The data repository contains a replicated database of the health and usage data of aircraft components to mitigate failure or error conditions in one of the databases.

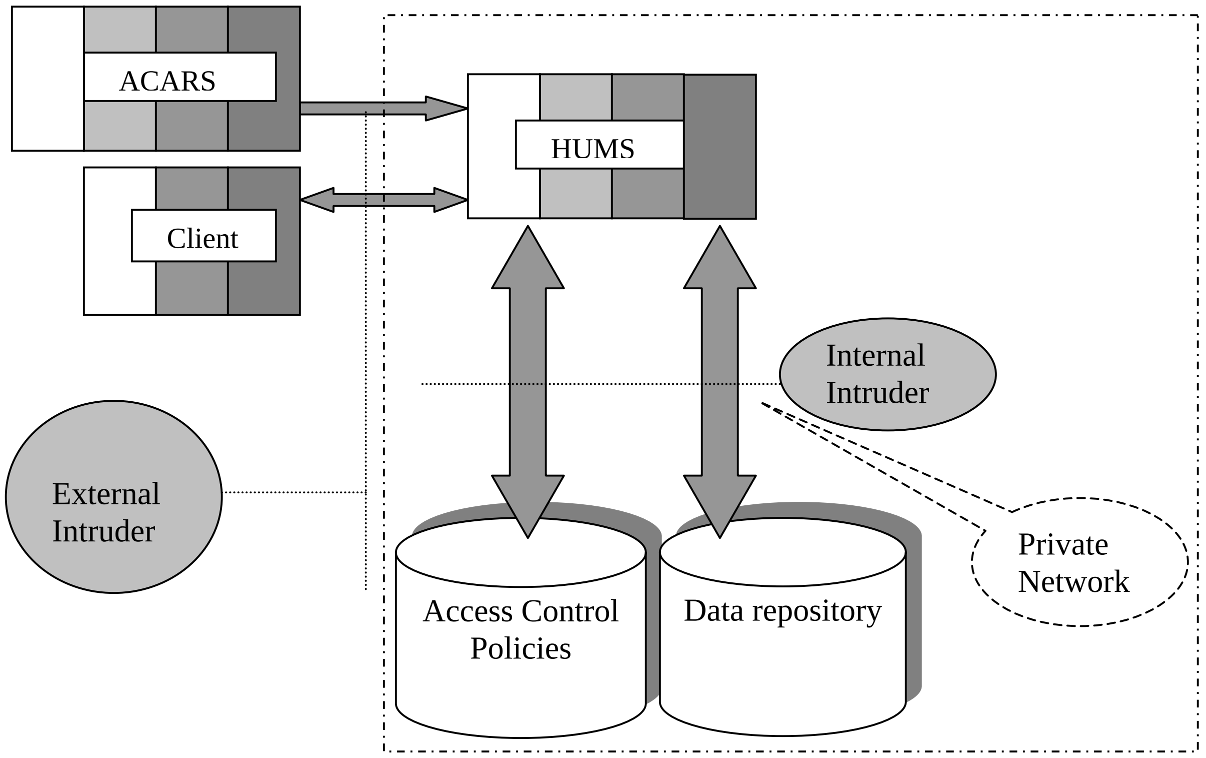


FIGURE A-1. DEMONSTRATION PROJECT

A brief description of the purpose of each application is given below:

- ACARS Simulator. The ACARS application generates simulated ACARS messages and sends them to the HUMS application over the network after using appropriate encryption. This is a Microsoft® Visual Basic® (VB) application with an appropriate GUI to allow the user to simulate different conditions in the network, such as failed authentication and buffer overflow attacks. Then, an appropriate response from the HUMS system can be checked. The ACARS application also provides a safety checklist that should be checked before a message is sent to the HUMS application.

- HUMS Application. The HUMS engine can be considered to be the heart of the system. It receives messages from the ACARS simulator and from the external clients. The messages received from the ACARS simulator are reports about health and usage of aircraft parts, and are stored in the data repository. The data repository itself employs a rudimentary replication scheme to illustrate error detection and remedial techniques in a COTS database. The HUMS database also uses access control rules to authorize and authenticate all communication that takes place within the HUMS. The external clients play the role of any third party application wishing to access the HUMS database, e.g., a maintenance crew. The HUMS application is implemented in Microsoft VB.

- Client. The client application simulates the working of external entities wishing to access the data present in the HUMS repository. Each client must authenticate itself with the HUMS application before accessing the aircraft parts database.

A.2 USING THE POC.

The POC was used to learn lessons in the following areas:

- Safety features to be used in ground-based systems that use COTS components
- Security of ground-based systems that use COTS components
- DO-278's applicability, weaknesses, and strengths, as relevant to COTS software
- HUMS guidance related to COTS
- General ground-based systems implementation

APPENDIX B—SYSTEM SECURITY EVALUATION CRITERIA

This section describes the Common Criteria for Information Technology Security Evaluation Model [B-1 through B-3]. The common criteria (CC) is used for evaluating the security properties of Information Technology (IT) systems and products. The CC is to be used to compare and choose commercial off-the-shelf (COTS) components for ground-based systems.

Deciding on which COTS components should be used in a ground-based system poses challenges:

- Determining if a particular piece of software, hardware, or firmware exactly matches security needs of the health and usage monitoring system (HUMS), without leaving the system either exposed or expensively overprotected.

- Plowing through the vendor type and obscure specifications of the COTS products that are available so as to choose components that will best protect the HUMS system.

The CC Information Technology Security Evaluation, usually referred to as the CC, was designed to meet these challenges [B-4].

The CC provides a framework to rate products by Evaluation Assurance Level (EAL). Each EAL embodies a recommended set of assurance requirements: the higher the EAL, the more secure the product. The CC helps determine if a given product meets the security needs of a system. By rating different products according to their EALs, the CC allows system engineers to comparison-shop and select appropriately secure products for the systems that use integrated COTS products.

B.1 THE COMMON CRITERIA AND BUILDING BLOCKS.

The CC for IT security evaluation is an international standard that supports the evaluation and development of security products [B-5].

The CC resulted from a combined effort by six countries: Canada, France, Germany, United Kingdom, the Netherlands, and the United States. The aim was to provide a set of criteria that could be used globally to assess the effectiveness of security products [B-6].

The CC builds upon the following standards:

- Europe’s Information Technology Security Evaluation Criteria
- The U.S.’s Trusted Computer System Evaluation Criteria
- The Canadian Trusted Computer Product Evaluation Criteria

The CC includes a combination of the best features of all these previous standards. The culmination of this whole process is CC 2.1, which is formally recognized as International Organization for Standardization 15408.

The foundation of the CC is formed by the functional requirements and the assurance security requirements [B-1 through B-3]. The functional requirements state the security functionalities that a particular product intends to provide to the consumer. Satisfying the functional requirements means that the assurance requirements have been met.

Requirement constructs include package, protection profile (PP), and security target (ST).

- Package. A combination of requirements components that can help express a set of functional and assurance requirements that meet a subset of security objectives. An example of a package would be functional requirements required for discretionary access controls.

- Protection Profile. A protection profile is an implementation-independent statement of security requirements that is shown to address threats that exist in a specified environment. It contains a set of security requirements that may be drawn from the CC or stated explicitly. The PP is needed to set up a standard for a particular product type.

- Security Target. It contains a set of security requirements that may be derived from a PP, stated explicitly, or drawn from a reference to CC functional or assurance components. The security target is the basis against which an evaluation is performed. The security target contains the target of evaluation (TOE) security threats, objectives, requirements, and summary specification of security functions and assurance measures. An ST is needed when a product is submitted for evaluation. An ST is also used when a product is submitted to a consumer as a statement of the TOEs security functionality and evaluated configuration.

B.1.1 SECURITY FUNCTIONAL REQUIREMENTS.

Consumers of a product use security functional requirements for the purpose of guidance and reference when formulating statements of requirements for security functions. Product developers can use security functional requirements for reference as well as for interpreting statements of functional requirements and then formulating the specifications for TOEs. Finally, the evaluators of a product can use the security functional requirements as a mandatory statement of evaluation criteria when determining if a TOE meets the claimed security functions.

Security functionality requirements are grouped into classes; all members of a class share a common focus. The various security functionality classes contained in part 2 of the CC are [B-2]:

- Audit
- Identification and authentication
- Cryptographic support
- Security management
- Communications
- Privacy

- User data protection
- Resource utilization
- TOE access
- Trusted path/channels
- Protection of TOE security functions

Each of these classes contains a number of families. The requirements in each family share security objectives but differ in rigor or emphasis. Each family contains one or more components, which may or may not be in a hierarchy.

Consider, for example, the audit class that has six different families that deal with various aspects of auditing. One of the families is Audit Generation. The audit data generation family has two components that are nonhierarchical; one component deals with the generation of data and the other deals with the association of a user with an auditable event.

B.1.2 SECURITY ASSURANCE REQUIREMENTS.

Security assurance requirements aid the consumer in determining the level of assurance that is required. The developer of the product uses the security assurance requirements when interpreting statements of assurance requirements and determining TOE assurance approaches. Product evaluators can use security assurance requirements as a mandatory statement of evaluation criteria when determining the assurance of TOE and also when evaluating PPs and STs. The security assurance requirements are grouped into classes as follows [B-3]:

- Configuration management
- Guidance documents
- Vulnerability assessment
- Delivery and operation
- Life cycle support
- Assurance maintenance
- Development
- Tests

Each of the classes contains a number of families, and the requirements within each family share a common objective. As an example, the development class contains seven families that deal with various aspects of design documentation (e.g., functional specification).

Each family includes one or more components that are in a strict hierarchy when more than one component exists in the family. Consider, for example, the functional specification family. The family contains four hierarchical components that deal with increasing completeness and formality in the presentation of the functional specification.

The CC has provided seven predefined assurance packages, known as EALs. These are explained in the next section.

B.1.3 COMMON CRITERIA EALs.

The CC provides seven assurance packages on a rising scale of assurance levels known as EALs. These provide balanced groupings of assurance components that are intended to be generally applicable.

The CC's seven EALs define a scale for the criteria used in evaluating targets of evaluation. Ordered hierarchically by degree of assurance, the EALs balance the assurance level obtained with the cost and feasibility of acquiring it [B-3].

- EAL1: Functionally Tested. Applies when confidence in a product's correct operation is needed, but security threats are not considered serious.

  An evaluation of the product at this level should prove that the TOE functions in a manner consistent with the documentation of the product, and that it provides useful protection against identified threats.

- EAL2: Structurally Tested. Applies when either the developers or users require low-to-moderate, independently assured security.

  This situation may occur in securing legacy systems or in cases of limited developer access.

- EAL3: Methodically Tested and Checked. Applies when developers or users need a moderate level of independently assured security and a complete investigation of the TOE and its development, without substantial reengineering.

- EAL4: Methodically Designed, Tested, and Reviewed. Applies when developers or users require moderate-to-high, independently assured security in conventional commodity products and are prepared to bear additional security-specific engineering costs.

- EAL5: Semiformally Designed and Tested. Applies when developers or users require high, independently assured security in a planned development and require a rigorous development approach that does not incur unreasonable costs from specialist security engineering techniques.

- EAL6: Semiformally Verified Design and Tested. Applies when developing security TOEs for application in situations involving high risk, where the value of the assets being protected justifies the additional costs.

- EAL7: Formally Verified Design and Tested. Applies to the development of security TOE for application in extremely high-risk situations, as well as when the high value of the assets involved justifies the higher costs.

B.2 REFERENCES.

B-1. "Common Criteria for Information Technology Security Evaluation Part 1: Introduction and General Model," Version 2.1 CCIMB -99-031, August 1999.

B-2. "Common Criteria for Information Technology Security Evaluation, Part 2: Security Functional Requirements," Version 2.1 CCIMB -99-032, August 1999.

B-3. "Common Criteria for Information Technology Security Evaluation, Part 3: Security Assurance Requirements," Version 2.1 CCIMB -99-033, August 1999.

B-4. Wenchang Shi; Yufang Sun "An Investigation of CC's Contribution to Confidence in Security," *Proceedings of the International Conference on Computer Networks and Mobile Computing*, pp. 333-338, 2001.

B-5. S. Lipner, "Twenty Years of Evaluation Criteria and Commercial Technology," *Proceedings of the 1999 IEEE Symposium on Security and Privacy*, pp. 111-112, 1999.

B-6. K. Caplan and J.L. Sanders, "Building an International Security Standard," *IT Professional*, Vol. 1, No. 2, pp. 29-34, March-April 1999.

APPENDIX C—A CASE STUDY OF COTS USED IN A DIFFERENT DOMAIN

To study the use of commercial off-the-shelf (COTS) in a different domain area for the purpose of safety and security, a nuclear certifiable tester for the Minuteman III Inter Continental Ballistic Missile (ICBM) was considered [C-1]. The purpose of the case study was to see if some lessons could be learned from the experience of designing the tester. The application of the lessons learned from this case study was considered for the health and usage monitoring system (HUMS) case. The entire case study is presented in the form of a series of questions relating to the safety and security issues.

- What is the goal of a nuclear certifiable tester?

  The goal of a nuclear certification is to ensure that no system hardware or software can impact the safety of the weapons system.

- Why did the old version of testing require changes?

  - The previous testing system was developed during the 1960s.
  - The method was outdated and difficult to maintain.
  - Criteria for safety had changed considerably.

- Why was COTS chosen for any improvement initiative?

  - To reduce the design cycle and material costs.

- What were the steps followed for the development of the system?

  - Requirements analysis
  - Requirements approval
  - Design
  - Design verification and validation, i.e. detailed analysis
  - Approval
  - Certification

- What safety analysis steps were carried out?

  Safety requirements were captured in a Requirements Traceability Matrix. This took inputs from the Prime Item Development Specification, Software Requirement Specification, and so on. Effects of component failures were captured in the Failure Modes Effects and Criticality Analysis Report, which describes the effect of each possible component failure of custom designs and COTS components. It also assigns a criticality factor to each failure (based on the severity of potential damage), provides a failure rate, and so on. The Testability Design Analysis Report detailed where each failure effect is detected (e.g., during testing) and the system response to that failure. The

Nuclear Surety Evaluation Report provides analysis and test results demonstrating that the system complies with the nuclear safety requirements.

- What were the safety related design implementations used?

  To meet all stringent constraints using COTS components, a fence concept was developed. The fence acts as a wrapper around COTS components. Each signal passes through this fence, which monitors the signals for any faults. If any fault is detected, the system is placed into a known safe state that disconnects all outputs and removes all stimuli sources by opening guard relays. The inbuilt monitoring mechanisms of the COTS components were also used, as was redundancy. For example, to use a COTS controller, dual-controller architecture was used. If the primary controller fails, the secondary controller takes over.

- What were the verification and validation steps taken?

  This was an iterative process. Hardware and software were tested separately and then together. Hardware was verified using discreet circuit simulations. Software was verified by unit testing, function testing, and so on. In addition, all software components were reviewed by software quality assurance to ensure that they complied with all stated requirements. An independent software validation team carried out validation.

- What were the software constraints of the project?

  – The software had to be developed under the DoD-STD-2167A methodology in a top-down design.

  – Use of a high-order language, like Ada or C++.

  – Ensuring memory integrity.

  – Prioritized interrupts.

  – Multitasking.

- What were the main criteria in selecting the desired COTS components?

  – The operating system (OS) had to allow easy integration of COTS and custom equipment.

  – The OS must be mature and have sufficient documentation.

  – The selection of COTS instruments in the tester was driven by the requirements of functional tests.

  – Twenty-year service goal to ensure long product life, minimal spares, and budgets.

As can be seen from the above discussion, the justification for using COTS products is the same as that for HUMS applications (i.e., to reduce development time and costs). Some of the lessons learned from this that can be applied to the COTS HUMS application are

- The primary advantage gained in adopting a COTS approach is the reduction in design cycle time and costs. This is a big motivation factor for the use of COTS.
- The absolute necessity of a planning, requirement, design approval and certification process.
- Sometimes COTS capabilities may have to be augmented by the use of a wrapper to incorporate safety and security guarantees.

C.1 REFERENCES.

C-1. J. Satterfield and D. Douthit, “How to Design Nuclear Certifiable COTS-Based ATE,” *AUTOTESTCON IEEE Proceedings*, pp. 525-533, 2000.